PUBLIC

SAP Data Services

Document Version: 4.2 Support Package 14 (14.2.14.0) – 2022-09-17

Reference Guide

THE BEST RUN SAP

Content

1 About this guide

The Data Services Reference Guide provides detailed information about the objects, data types, transforms, and functions in the Designer.

For source-specific information, such as information pertaining to a particular back-office application, refer to the supplement for that application.

1.1 Who should read this guide

We expect that readers of this Reference Guide have advanced experience and knowledge of the SAP Data Services application and the database systems with which they work.

We've written the Reference Guide, and other Data Services documentation for software developers, consultants, or database administrators who work on data extraction, data warehousing, data integration, or data quality. We assume that you're experienced with the following technology:

- Your source and target data systems, database management system, legacy systems, business intelligence, and messaging concepts
- Your organization's data needs
- SQL (Structured Query Language)
- Installation environments: Microsoft Windows or UNIX.

If you plan to use Data Services to design real-time processing, you are familiar with:

- DTD and XML schema formats for XML files
- Publishing Web Services, such as WSDL, HTTP/S, and SOAP protocols.

1.2 Naming conventions and variables

This documentation uses specific terminology, location variables, and environment variables that describe various features, processes, and locations in SAP Business Objects and SAP Data Services.

Terminology

SAP Data Services documentation uses the following terminology:

- The terms **Data Services system** and **SAP Data Services** mean the same thing.
- The term **BI platform** refers to **SAP BusinessObjects Business Intelligence platform**.
- The term **IPS** refers to **SAP BusinessObjects Information platform services**.

> **Note**
>
> Data Services requires BI platform components. However, when you don't use other SAP applications, IPS, a scaled back version of BI, also provides these components for Data Services.

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

Variables

The following table describes the location variables and environment variables that are necessary when you install and configure Data Services and required components.

Variables	Description
`INSTALL_DIR`	The installation directory for SAP applications such as Data Services. Default location: • For Windows: `C:\Program Files (x86)\SAP BusinessObjects` • For UNIX: `$HOME/sap businessobjects` **Note** `INSTALL_DIR` isn't an environment variable. The installation location of SAP software can be different than what we list for `INSTALL_DIR` based on the location that your administrator sets during installation.

Variables	Description
`BIP_INSTALL_DIR`	The directory for the BI or IPS platform. Default location: • For Windows: `<INSTALL_DIR>\SAP BusinessObjects Enterprise XI 4.0` **Example** `C:\Program Files (x86)\SAP BusinessObjects\SAP BusinessObjects Enterprise XI 4.0` • For UNIX: `<INSTALL_DIR>/enterprise_xi40` **Note** These paths are the same for both BI and IPS. **Note** `BIP_INSTALL_DIR` isn't an environment variable. The installation location of SAP software can be different than what we list for `BIP_INSTALL_DIR` based on the location that your administrator sets during installation.
`<LINK_DIR>`	An environment variable for the root directory of the Data Services system. Default location: • All platforms `<INSTALL_DIR>\Data Services` **Example** `C:\Program Files (x86)\SAP BusinessObjects\Data Services`

Variables	Description
<DS_COMMON_DIR>	An environment variable for the common configuration directory for the Data Services system. Default location: • If your system is on Windows (Vista and newer): <AllUsersProfile>\SAP BusinessObjects\Data Services **i Note** The default value of <AllUsersProfile> environment variable for Windows Vista and newer is C:\ProgramData. **Example** C:\ProgramData\SAP BusinessObjects\Data Services • If your system is on Windows (Older versions such as XP) <AllUsersProfile>\Application Data\SAP BusinessObjects\Data Services **i Note** The default value of <AllUsersProfile> environment variable for Windows older versions is C:\Documents and Settings\All Users. **Example** C:\Documents and Settings\All Users\Application Data\SAP BusinessObjects\Data Services • UNIX systems (for compatibility) <LINK_DIR> The installer automatically creates this system environment variable during installation. **i Note** Starting with Data Services 4.2 SP6, users can designate a different default location for <DS_COMMON_DIR> during installation. If you can't find the <DS_COMMON_DIR> in the listed default location, ask

Variables	Description
	your System Administrator to find out where your default location is for `<DS_COMMON_DIR>`.
`<DS_USER_DIR>`	The environment variable for the user-specific configuration directory for the Data Services system. Default location: • If you're on Windows (Vista and newer): `<UserProfile>\AppData\Local\SAP BusinessObjects\Data Services` **i Note** The default value of `<UserProfile>` environment variable for Windows Vista and newer versions is `C:\Users\{username}`. • If you're on Windows (Older versions such as XP): `<UserProfile>\Local Settings\Application Data\SAP BusinessObjects\Data Services` **i Note** The default value of `<UserProfile>` environment variable for Windows older versions is `C:\Documents and Settings\{username}`. **i Note** The system uses `<DS_USER_DIR>` only for Data Services client applications on Windows. UNIX platforms don't use `<DS_USER_DIR>`. The installer automatically creates this system environment variable during installation.

2 Objects in Data Services

An object is anything that you define, edit, or use in SAP Data Services Designer.

Each Data Services object falls into one of the following classes:

- Single use
- Reusable

The object class determines how you create and retrieve the object.

> **i Note**
>
> For information about source-specific objects, consult the applicable supplement document for that source. For example, for information about SAP applications as a source, consult the *Supplement for SAP*.

The following table describes all objects in Data Services in alphabetical order, and includes the object class. For more information about each object, see the *Reference Guide*.

Objects in Data Services

Object	Object class	Description
Annotation	Single-use	Contains information about a flow in the workspace, such as a work flow, data flow, or a combination of 2 work flows. Create the annotation and attach it to the applicable item in the workspace.

Object	Object class	Description
Batch job	Reusable	Defines processes that Data Services runs on demand or at a scheduled time. Generates error, monitor, and trace messages. Create a job in the object library or in an opened project. After you save a batch job, Data Services saves it to the local repository and displays it in the local object library. The batch job in the object library is a direct reference to the batch job in the repository. Add multiple unique batch jobs to a project. **i Note** A project can't contain multiple instances of the same job.
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.
Data flow	Reusable	Specifies the requirements for extracting, transforming, and loading data from sources to targets. A data flow can be a part of a batch job or a real-time job.
Datastore	Reusable	Specifies the connection information to import metadata for a database or other data sources. Add the imported source metadata to the workspace, and not the datastore.
Document	Reusable	Describes a data structure for complicated nested schemas. Available only for certain adapter datastore objects.

Object	Object class	Description
DTD	Reusable	Describes the format that an XML file or message reads or writes. "DTD" stands for Document Type Definition.
Excel workbook format	Reusable	Defines the format for an Excel workbook source.
File format	Reusable	Describes the location and file name of a flat file. Also describes the arrangement of flat file data in a source or target file.
File location	Reusable	Defines the file transfer protocol for transferring data files from 1 location to another location, such as a remote and local server. Includes information about the remote and local servers. Associate a file location object to a file format to use as a source or target in a data flow: • **As a source:** Transfers source data from a remote server to a local server. • **As a target:** Transfers the output file from the local server to the remote server.
Function	Reusable	A procedure that returns a value.
HDFS file format	Reusable	Describes the structure of a Hadoop distributed file system.
JSON file	Single-use	A source or target that you can use in a batch or real-time job. • **As a source:**Translates incoming JSON-formatted data into data that Data Services can process. • **As a target:** Translates the data produced by a data flow, including nested data, into a JSON-formatted file.

Object	Object class	Description
JSON message	Single-use	A source or target that you can use in a real-time job. • **As a source:** Translate incoming JSON-formatted requests into data that a real-time job can process. • **As a target:** Translate the result of the real-time job, including hierarchical data, into a JSON-formatted response, and sends the messages to the Access Server.
Log	Single-use	Records information from a single job run.
Message function	Reusable	Accommodates XML messages when properly configured. Available only for certain adapter datastores.
Nested Schemas template	Single-use	A target that creates a JSON file or an XML file that matches a particular input schema. Doesn't require a DTD, JSON Schema, or XML Schema.
Outbound message	Reusable	XML-based, hierarchical communication that real-time jobs publish to adapters. Available only for certain adapter datastores.
Project	Single-use	Contains groups of one or more jobs and applicable child objects, such as a data flow, for convenient access.
Query transform	Single-use	Defines conditions to retrieve a specified data set.

Object	Object class	Description
Real-time job	Reusable	Defines activities that Data Services executes on demand in real time. Create real-time jobs in the Designer. Then configure and run as services associated with an Access Server in the Administrator: • Design real-time jobs according to the model rules of the data flow. • Run real-time jobs as a request-response system.
Script	Single-use	Evaluates expressions, calls functions, and assigns values to variables.
Source	Single-use	Data that the Data Services reads and processes in a data flow.
Table	Reusable	Defines an external DBMS table for which metadata has been imported. Or, defines a target table into which data is or has been placed. A table: • Is associated with a specific datastore, and doesn't exist independent of a datastore connection. • Retrieves or stores data based on the schema of the table definition from which it was created.
Target	Single-use	Accepts extracted and transformed data in a data flow.
Template table	Reusable	Represents a new table that you add to a database. All datastores, except SAP application datastores, have a default template to create tables in the datastore. Data Services creates the schema for each instance of a template table at runtime. Data Services bases the schema on the data loaded into the template table.

Object	Object class	Description
Transform	Reusable	Configured to perform specific operations on data sets. Requires 0 or more data sets; produces 0 or 1 data set, which you can split into more that 1 data set.
Try/Catch	Single-use	Contains two objects: Try and Catch. Try/Catch block specifies the steps to execute if an error occurs in a given exception group while a job is running. A Try can have more than 1 Catch.
While loop	Single-use	A sequence of steps that Data Services repeats as long as a set condition results in **True**.
Work flow	Reusable	Defines a specific order for data flows and the operations that support a data flow.
XML file	Single-use	Translates XML data into applicable formats, based on whether it's a source or target, for batch and real-time jobs: • **As a source:** Translates incoming XML-formatted data into data that Data Services can process. • **As a target:** Translates data produced by a data flow, including nested data, into an XML-formatted file.
XML message	Single-use	Translates XML messages based on whether it's a source or target in real-time jobs: • **As a source:** Translates incoming XML-formatted requests into data that a real-time job can process. • **As a target:** Translate the result of the real-time job, including hierarchical data, into an XML-formatted response, and sends the messages to the Access Server.

Object	Object class	Description
XML Schema	Reusable	Describes an XML file or message data structure so an XML document can read or write the data.

Object metadata [page 24]
Object metadata consists of the options, properties, and attributes of the object.

Object descriptions [page 25]
Data Services operates using objects, and there are numerous objects that combine for Data Services processes.

2.1 Object metadata

Object metadata consists of the options, properties, and attributes of the object.

The following table describes the metadata that Data Services saves to the options, properties, and attributes of an object.

Object information	Description
Options	Defines the object operations such as database connection information, actions, passwords, and usernames. View and change object options by opening the object editor.
Properties	Describes the object, such as object name and description. View and change object properties by right-clicking the object name and selecting *Properties* and viewing the *General* tab.
Attributes	Provides other information about an object, such as date created, date modified, and if the object is enabled for Web service use. Attribute values can affect the behavior of an object. View object attributes by right-clicking the object name and selecting *Properties*. Then open the *Attributes* tab.

Parent topic: Objects in Data Services [page 18]

Related Information

Object descriptions [page 25]

2.2 Object descriptions

Data Services operates using objects, and there are numerous objects that combine for Data Services processes.

The following table lists all objects in Data Services, the class of the object, and a description of the object. For more information about each object, see the individual object topics.

Object descriptions

Object	Object class	Description
Annotation	Single-use	Note that contains information about aspects of a flow, part of a flow, or a diagram in the workspace. You create the annotation and attach it to the item in the workspace.
Batch job	Reusable	Defines activities that the software executes at a given time, and includes error, monitor, and trace messages. Add jobs to projects only. After you save a batch job, the software places it in the object library. The batch job in the object library is a direct reference to the batch job object. Projects can contain references to multiple batch job objects, but only one reference per batch job.
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.
Data flow	Reusable	Specifies the requirements for extracting, transforming, and loading data from sources to targets. A data flow can be a part of a batch job or a real-time job.
Datastore	Reusable	Specifies the connection information to access a database or other data source. You cannot add the datastore object itself to the workspace.
Document	Reusable	Describes a data structure for complicated nested schemas. Available only for certain adapter datastore objects.
DTD	Reusable	Describes the format that an XML file or message reads or writes. DTD stands for Document Type Definition.
Excel workbook format	Reusable	Defines the format for an Excel workbook source.
File format	Reusable	Describes the location and file name of a flat file and the arrangement of flat file data in a source or target file.

Object	Object class	Description
File location	Reusable	Defines the file transfer protocol to use for transferring data files. Includes information about the remote and local servers. Associate a file location object to a file format to use as a source or target in a data flow. • As a source, the software uses the file location object information to transfer source data from a remote server to a local server. • As a target, the software uses the file location object information to transfer the output file from the local server to the remote server.
Function	Reusable	Returns a value.
HDFS file format	Reusable	Describes the structure of a Hadoop distributed file system.
JSON file	Single-use	A batch or real-time source or target. • As a source, a JSON file translates incoming JSON-formatted data into data that the software can process. • As a target, a JSON file translates the data produced by a data flow, including nested data, into a JSON-formatted file.
JSON message	Single-use	A real-time source or target. • As a source, JSON messages translate incoming JSON-formatted requests into data that a real-time job can process. • As a target, JSON messages translate the result of the real-time job, including hierarchical data, into a JSON-formatted response and sends the messages to the Access Server.
Log	Single-use	Records information about a particular execution of a single job.
Message function	Reusable	Accommodates XML messages when properly configured. Available only for certain adapter datastores.
Nested Schemas template	Single-use	A target that creates a JSON file or an XML file that matches a particular input schema. Does not require a DTD, JSON Schema, or XML Schema.
Outbound message	Reusable	XML-based, hierarchical communications that real-time jobs can publish to adapters. Available only for certain adapter datastores.
Project	Single-use	Contains groups of one or more jobs for convenient access.
Query transform	Single-use	Defines conditions to retrieve a specified data set.

Object	Object class	Description
Real time job	Reusable	Defines activities that the software executes on demand in real time. Create real time jobs in the Designer. Then configure and run as services associated with an Access Server in the Administrator. • Design real time jobs according to data flow model rules. • Run real time jobs as a request-response system.
Script	Single-use	Evaluates expressions, calls functions, and assigns values to variables.
Source	Single-use	Contains source data that the software reads and processes in a data flow.
Table	Reusable	Defines an external DBMS table for which metadata has been imported, or a target table into which data is or has been placed. A table is associated with a specific datastore. A table does not exist independently of a datastore connection. A table retrieves or stores data based on the schema of the table definition from which it was created.
Target	Single-use	Accepts extracted and transformed data in a data flow.
Template table	Reusable	Represents a new table that you add to a database. All datastores except SAP datastores have a default template that you use to create tables in the datastore. The software creates the schema for each instance of a template table at runtime. The software bases the created schema on the data loaded into the template table.
Transform	Reusable	Contains settings that perform specific operations on data sets. Requires zero or more data sets; produces zero or one data set, which can be split into more that one data set.
Try	Single-use	Introduces a Try/Catch block in a data flow. You can have more than one catch with a single try.
While loop	Single-use	Causes a sequence of steps to repeat as long as a set condition results in True.
Work flow	Reusable	Contains a specific order of data flows and operations that support a data flow.

Object	Object class	Description
XML file	Single-use	Translates XML data into applicable formats based on whether it is a source or target for batch and real time jobs. • As a source, translates incoming XML-formatted data into data that the software can process. • As a target, translates the data produced by a data flow, including nested data, into an XML-formatted file.
XML message	Single-use	Translates XML messages based on whether it is a source or target in real time jobs. • As a source, translates incoming XML-formatted requests into data that a real-time job can process. • As a target, 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	Describes an XML file or message data structure so an XML document can read or write the data.

Parent topic: Objects in Data Services [page 18]

Related Information

Object metadata [page 24]

2.2.1 Annotation

Use an annotation like a stick-on note to add custom information about objects in your workspace.

Annotation characteristics

Characteristic	Description
	Annotation icon
Class	Single-use

Characteristic	Description
Access	Click the annotation icon in the tool palette, then click in the workspace. The software adds a stick-on note text box that you type your note into. Add notes about the flow, portions of the flow, or a diagram in your workspace. Drag the annotation to any location in your workspace as applicable. When you select the annotation, nodes appear on all sides for resizing.
Description	Annotations contain customized notes regarding aspects of 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. Annotations do not have options or properties.

Related Information

Creating and deleting annotations

2.2.2 Batch Job

Batch jobs process an entire dataset or multiple datasets.

Batch job characteristics

Characteristic	Description
	Batch job icon.
Class	Reusable
Access	Access batch jobs in the following locations: • In the object library, click the *Jobs* tab to open an existing batch job in the workspace. • In the project area, right-click the applicable project and select *New Batch Job* to create a new batch job and add it to the selected project.

Characteristic	Description
Description	A batch job contains a set of objects including work flows and data flows. You can schedule a batch job to run at a specific time, or you can execute the job in Designer after you save it. Before you can execute a job, it must contain objects in a specific hierarchical order. For example, add a work flow as a child of the job, and then add a data flow as a child of the work flow. For complete information about batch jobs, see the *Designer Guide*. A job is the only executable object in Data Services.

Batch job objects [page 30]
Objects that you add to batch jobs determine the source, target, and the types of Data Services processes to perform.

Batch job attributes and properties [page 31]
The software assigns built-in attributes to an object. You can view and sometimes change an object property through the job properties.

Batch job trace properties [page 32]
Trace properties are types of information that the software monitors during job execution, and writes to trace files.

Related Information

Projects and jobs

2.2.2.1 Batch job objects

Objects that you add to batch jobs determine the source, target, and the types of Data Services processes to perform.

A batch job can contain the following objects:

- Data flows, which include the following objects: Sources, transforms, targets
- Work flows
- Scripts
- Conditionals
- Try/catch blocks
- While Loops

Parent topic: Batch Job [page 29]

Related Information

Batch job attributes and properties [page 31]
Batch job trace properties [page 32]

2.2.2.2 Batch job attributes and properties

The software assigns built-in attributes to an object. You can view and sometimes change an object property through the job properties.

Attributes

Batch jobs have the following built-in attributes:

Attribute	Description
Name	The name that the job creator provided for the job. This name appears on the object in the object library and in the calls to the object.
Description	The description that the job creator provided for the job.
Date created	The creation date of the object that the software assigns using the system date.

Properties

Batch job properties determine the information collected and logged when running the job. Set the default properties that apply each time you run the job or set execution or runtime properties that apply for a particular run. Execution properties override default properties.

To view the job properties, right-click the name of the job and select *Properties* from the dropdown menu.

The *Properties* dialog box contains some view-only information and editable information in several tabs. The available tabs are based on the object type.

Batch job property tabs

Tab	Content
General	The name and description of the job.
Attributes	The attributes and values of the job.
Class Attributes	The attribute types for the selected object.

Tab	Content
Execution Options	The default execution options. Override these settings by making settings in the *Execute* dialog box when you execute the job.
Trace	The default trace properties. You can set default trace properties in the *Trace* tab.
Global Variables	The global variable values connected to the batch job. For complete information about global variables, see the *Designer Guide*.

Parent topic: Batch Job [page 29]

Related Information

Batch job objects [page 30]
Batch job trace properties [page 32]

2.2.2.3 Batch job trace properties

Trace properties are types of information that the software monitors during job execution, and writes to trace files.

Use the *Trace* tab of the *Properties* window to select the information that SAP Data Services monitors and writes to the trace log file during a job. The software writes trace messages to the trace log associated with the current Job Server and writes error messages to the error log associated with the current Job Server.

To enable or disable a trace property, open the*Trace* tab and select *Yes* or *No* in the *Value* list for the applicable property.

The following table contains descriptions for each trace property in alphabetical order.

Trace property	Description
Access Server Communication	Writes messages exchanged between the Access Server and a service provider, including the following message types: • A registration message, which tells the Access Server that the service provider is ready. • A request to execute that the Access Server sends to the service provider. • A response from the service provider to the Access Server. • All requests from the Access Server to shut down.

Trace property	Description
Assemblers	Writes messages for Substitution Parameter and SDK Transforms: • Substitution parameters: Writes trace messages such as the substitution configuration used and the value substituted for each substitution parameter. • SDK transforms: Writes transform-specific information specified in XML format. The transform-specific information can be hierarchical. The Assemblers trace message writes the XML extracted or assembled at runtime.
Audit Data	Writes a message when auditing: • Collects a statistic at an audit point • Determines if an audit rule passes or fails
Data Flow	Writes a message when the data flow starts, when the data flow successfully finishes, or when the data flow terminates due to error. This trace also reports when the bulk loader starts, any bulk loader warnings occur, and when the bulk loader successfully completes.
Debug Package	Creates a package containing the trace, monitor, and log files and the ATL for the job.
IDoc file reader	Writes a message when reading IDoc files including: • start reading the IDoc file • stop reading the IDoc file • result of the IDoc file validation
Memory Source	Writes a message for every row retrieved from the memory table.
Memory Target	Writes a message for every row inserted into the memory table.
Optimized data flow	For consulting and customer assurance use.
RFC Function	Writes a message related to RFC calls including: • start of RFC call • end of RFC call • a message for each record received from the software for the RFC call
Row	Writes a message when a transform imports or exports a row.
SAP Table Reader	Writes messages from readers of SAP system tables including: • start reading from table • stop reading from table • start of connection to SAP system where table is present • end of connection to SAP system

Trace property	Description
Scripts and Script Functions	Writes a message when the software runs a script or calls a script function. Specifically, this trace links a message when: • The script is called. Scripts can be started any level from the job level down to the data flow level. Additional (and separate) notation is made when a script is called from within another script. • A function is called by the script. • The script successfully completes.
Session	Writes a message when the job description is read from the repository, when the job is optimized, and when the job runs.
SQL Functions	Writes data retrieved before SQL functions: • Every row retrieved by the named query before the SQL is submitted in the key_generation function • Every row retrieved by the named query before the SQL is submitted in the lookup function (but only if PRE_LOAD_CACHE is not specified). • When mail is sent using the mail_to function.
SQL Loaders	Writes a message when the bulk loader: • Starts • Submits a warning message • Completes successfully • Completes unsuccessfully, if you select the *Clean up bulk loader directory after load* option Additionally, for Microsoft SQL Server and SAP ASE, writes when the SQL Server bulk loader: • Completes a successful row submission • Encounters an error This instance reports all SQL that the software submits to the target database, including: • When a `truncate table` command executes if the *Delete data from table before loading* option is selected. • Any parameters included in PRE-LOAD SQL commands • Before a batch of SQL statements is submitted • When a template table is created (and also dropped, if you turn on the *Drop/Create* option) • When a `delete from table` command executes if you turn on auto correct (Informix environment only). This trace also writes all rows that the software loads into the target.
SQL Only	Use 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.

Trace property	Description
SQL Transforms	Writes a message (using the Table_Comparison transform) about whether a row exists in the target table that corresponds to an input row from the source table. The trace message occurs before submitting the query against the target and for every row retrieved when the named query is submitted (but only if caching is not turned on).
Stored Procedure	Writes a message when the software calls a stored procedure. The report includes: • When the stored procedure starts • The SQL query submitted for the stored procedure call • The value (or values) of the input parameter (or parameters) • The value (or values) of the output parameter (or parameters) • The return value (if the stored procedure is a stored function) • When the stored procedure finishes
Tables	Writes a message when a table is created or dropped. The message indicates the datastore to which the created table belongs and the SQL statement used to create the table.
Trace ABAP	Writes a message when an ABAP data flow starts or stops, and to report the ABAP job status.
Trace Parallel Execution	Writes messages describing how data in a data flow is parallel processed.
Transform	Writes a message when a transform starts, completes, or terminates.
Work Flow	Writes a message when the work flow description is read from the repository, when the work flow is optimized, when the work flow runs, and when the work flow ends.

Parent topic: Batch Job [page 29]

Related Information

Batch job objects [page 30]
Batch job attributes and properties [page 31]

2.2.3 Catch

If one or more errors occur in a job execution, a catch object is the second part of a try/catch block, which provides an alternative workflow.

Catch characteristics

Characteristic	Description
	Catch icon
Class	Single-use
Access	With a work flow in the workspace, click the catch icon in the tool palette and then click in the workspace.
Description	A catch object is part of a serial sequence called a try/catch block. Try/catch blocks "catch" exception groups of errors, apply solutions that you provide, and continue execution.

For details about the Catch object, and about try/catch blocks, see the *Designer Guide*.

Catch attributes and exceptions [page 36]
A catch object has one attribute but many exception groups that you can catch in a try/catch block.

Catch error functions [page 37]
Use catch functions inside the catch object to identify details about the error.

Catch scripts [page 38]
Catch scripts define actions for specific errors using catch functions.

2.2.3.1 Catch attributes and exceptions

A catch object has one attribute but many exception groups that you can catch in a try/catch block.

Catch objects have a name attribute. The object name appears on the object in the data flow diagram.

Each try/catch block exception error group corresponds with an error group number. The Catch editor lists the exceptions. Select the checkbox next to each exception that you want the object to catch. The following table describes each exception group.

Try/catch block exception groups

Exception group	Group number	Description
Catch All Exceptions	All	All errors. Select by clicking the checkbox next to the *Exception* table heading in the editor.
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

Exception group	Group number	Description
Flat file processing errors	1004	Errors processing flat files
File access errors	1005	Errors accessing local and FTP files
Repository access errors	1006	Errors accessing the repository
SAP Execution errors	1007	Errors from the SAP system
System resource exception	1008	Errors accessing operating system resources
SAP BW execution errors	1009	Errors from the SAP BW system
XML processing errors	1010	Errors processing XML files and messages
COBOL copybook errors	1011	Errors processing COBOL copybook files
Excel book errors	1012	Errors processing Excel books
Data Quality transform errors	1013	Errors processing Data Quality transforms

For more information about try/catch blocks, see the *Designer Guide*.

Parent topic: Catch [page 36]

Related Information

Catch error functions [page 37]
Catch scripts [page 38]

2.2.3.2 Catch error functions

Use catch functions inside the catch object to identify details about the error.

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

> **i Note**
>
> You can call these error functions only 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, the software issues a validation error.

Catch object error functions and descriptions

Catch error function	Return data type and size	Description
`error_timestamp()`	timestamp	Returns the timestamp of the caught exception.

Catch error function	Return data type and size	Description
`error_context()`	varchar 512	Returns the context of the caught exception. For example: `|Session datapreview_job|Dataflow debug_DataFlow|Transform Debug`
`error_message()`	varchar 512	Returns the error message of the caught exception.
`error_number()`	int	Returns the error number of the caught exception.

Parent topic: Catch [page 36]

Related Information

Catch attributes and exceptions [page 36]
Catch scripts [page 38]
Example of error functions in catch block
Descriptions of Data Services built-in functions [page 1082]

2.2.3.3 Catch scripts

Catch scripts define actions for specific errors using catch functions.

A script is the most common action that a catch executes for a thrown exception. The catch script can contain the following:

- Catch error functions and other function calls
- Nested try/catch blocks
- `IF` statements to perform different actions for different exceptions

Syntax

The following sample code shows the syntax for a try/catch block within a script:

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

Where

`<steps>`	Contains catch error functions, other function calls, `IF` statements, or other statements that you want to perform for an error in the specified exception group number.

`<integer_constants >`	Contains one or more exception group numbers that you want to catch.
	Use a comma to separate exception group numbers. For example,
	`catch(1002,1003)`
	Specify `ALL` to catch all exceptions:
	`catch(ALL)`

Parent topic: Catch [page 36]

Related Information

Catch attributes and exceptions [page 36]
Catch error functions [page 37]
Example of error functions in catch block
Descriptions of Data Services built-in functions [page 1082]

2.2.4 COBOL copybook file format

Create a COBOL copybook file format to describe the structure of a COBOL copybook file.

COBOL copybook characteristics

Characteristic	Description
	Icon for a COBOL copybook file format.
Class	Reusable
Access	In the object library, click the *Formats* tab.

Characteristic	Description
Description	A COBOL copybook file format describes the structure defined in a COBOL copybook file (denoted with a .cpy extension). Store templates for file formats in the object library. Use the templates to define the file format of a particular source in a data flow. **i Note** You cannot use a COBOL copybook file format template as a target in a data flow.

For details about working with COBOL copybook file formats, see the *Designer Guide*.

COBOL copybook Format tab [page 40]
Define format options for the COBOL copybook file format when you create a new file format.

COBOL copybook Data File tab [page 42]
To define the data file for the COBOL copybook file format, complete options in the *Data File* tab.

COBOL copybook Data Access tab [page 45]
Specify the data transfer protocol so Data Services can access the COBOL copybook data file.

COBOL copybook Field ID tab [page 46]
Create rules for associating records to schemas when the imported COBOL copybook has multiple schemas.

COBOL copybook Record Length Field tab [page 47]
Identify the field that contains the length of the schema record and set an offset value in the *Record Length Field* tab.

2.2.4.1 COBOL copybook Format tab

Define format options for the COBOL copybook file format when you create a new file format.

Set format options only when you create a new COBOL copybook file format. You cannot edit the options when you edit the COBOL copybook file format.

COBOL copybook option descriptions in the Format tab

Format tab option	Description
File name	Specifies the COBOL copybook file name. Type the file name or click *Browse* to browse for the file. The COBOL copybook file contains the schema definition. For added flexibility, use a variable for this option.

Format tab option	Description
Expand OCCURS	Specifies whether the software imports OCCURS groups with each field. Select to import OCCURS groups with each field. OCCURS groups appear in one of the following views: • **Expanded view:** Get sequential suffixes, one for each repetition. For example, fieldname_1, fieldname_2, and so on. • **Collapsed view:** Appears only once in the copybook schema. For a collapsed view, the output schema matches the OCCURS group definition. For each input record there are several output records. If a copybook contains more than one OCCURS group, select the Expand OCCURS option. The default setting is with the *Expand OCCURS* option checked.
Ignore redefines	Specifies whether the software ignores REDEFINES clauses in the COBOL copybook file.

Source format	Specifies the format of the copybook source code.
	
	• *Free*: All characters on the line can contain COBOL source code.
	• *Smart mode*: The software attempts to determine whether the source code is in standard or free format. If this attempt does not produce the desired result, choose the appropriate source format, standard or free, manually for reimporting.
	• *Standard*: The traditional IBM mainframe COBOL source format, where each line of code is divided into the following five areas:
	  • sequence number (1-6)
	  • indicator area (7)
	  • area A (8-11)
	  • area B (12-72)
	  • comments (73-80)

Source codes [start]	Specifies the start column number of the copybook source file to use during the import. Typical value for IBM mainframe copybook: • 7 for standard source format • 0 for free source format
Source codes [end]	Specifies the end column number of the copybook source file to use during the import. Typical value for IBM mainframe copybook: • 72 for standard source format • 9999 for free source format

Format tab option	Description
Generate record number field	Specifies whether the transform creates a new field at the beginning of the schema. The software populates the new field at runtime with the record number.

Parent topic: COBOL copybook file format [page 39]

Related Information

COBOL copybook Data File tab [page 42]
COBOL copybook Data Access tab [page 45]
COBOL copybook Field ID tab [page 46]
COBOL copybook Record Length Field tab [page 47]

2.2.4.2 COBOL copybook Data File tab

To define the data file for the COBOL copybook file format, complete options in the *Data File* tab.

COBOL copybook option descriptions in the Data File tab

Data File tab option	Description
File Location	Optional. Specifies the name of an existing file location object. Select the name of an existing file location object from the dropdown list. A file location object contains file transfer protocol information, such as FTP or SFTP, and local and remote server information. The software uses this information to copy the COBOL copybook data file from a remote server to a local server. Then it accesses the local file to use the data as a source in a data flow. **i Note** When you choose a file location object for this option, the software disables the *Directory* option and the *Data Access* tab.

Data File tab option	Description
Delete file after transfer	Specifies whether the software deletes the local copy of the file after it loads as a source in the data flow. Only available when you set a file for *File Location*. • Check to delete the local copy of the data file after the software loads it as a source in a data flow. • Uncheck to save the local copy of the data file after the software loads it as a source in the data flow.
Directory	Specifies 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 *File name* field. During design, specify a file in one of the following ways: • For a file located on the computer where the Designer runs, you can use the *Browse* button. • For a file located on the computer where the Job Server runs, Type the path to the file. Type an absolute path or a relative path. Ensure that the Job Server can access the location. **i Note** The software disables this option if you enter a file location object name in *File Location*. For added flexibility, you can select a variable for this option.
File name	Specifies the COBOL copybook data file. If you leave *Directory* blank, then browse for or type the full path and file name here. For added flexibility, you can use variables or wild cards (* or ?). When you use a Cobol Copybook as a source, and your source is in cloud storage, specify the full remote path and file name. Include all subfolders in the full remote path. Supported cloud storages include Amazon S3, Azure Blob Storage, Azure Data Lake Store, Hadoop File System, and Google Cloud Storage.
Type	Specifies the record format type. • *Fixed(F)* • *Variable(V)*

Data File tab option	Description
Has record length	Specifies record length if the record format type is *Variable(V)* and the data file contains information about the length of each record. Available only when you select *Variable(V)* for *Type*. • *2-byte* integer • *2-byte followed by 0x0000* (integer followed by two 0 bytes) • *4-byte* integer • *None*—No length information at the beginning of each record
Record size	If the record format type is *Fixed(F)*, specifies the record length in bytes. All records in the file have this length, padded if necessary.
Record trailer length	Specifies the length of extra character padding in bytes at the end of each record.
Has record mark	Specifies that there is an extra byte in the beginning of each record data.
Integer format	Specifies the method that the existing data file uses to store binary data: • *Big endian*: The most significant byte comes first • *Little endian*: The least significant byte comes first
Code page	Specifies the character encoding of character data in the data file.
Skip first	Specifies the number of data records to skip before starting to process the file. The default is *{none}*. For added flexibility, you can enter a variable for this option.
Read total	Specifies the number of records the transform should read and process. The default is *{no limit}*. For added flexibility, you can enter a variable for this option.
Low Value *High Value*	Specifies the low and high value in hexadecimal. You can also use a variable to define a different value at runtime. • *Low Value*: Default is 0x40 • *High Value*: Default is 0xFF **Example** If the source field is binary 0x40, enter a *Low Value* of 0x40 and select the action *Convert to NULL*. The result would be as follows for these data types: • Char—Character represented by 0x40 • Packed decimal—NULL • Binary—0x40

Data File tab option	Description
Action	Specifies an action for *Low Value* and or *High Value*. • *No conversion*: Reads the value as an ASCII character (default). • *Convert to NULL*: Converts the given value to NULL. • *Convert to 0*: Converts the given value to 0.

Parent topic: COBOL copybook file format [page 39]

Related Information

COBOL copybook Format tab [page 40]
COBOL copybook Data Access tab [page 45]
COBOL copybook Field ID tab [page 46]
COBOL copybook Record Length Field tab [page 47]
File location object [page 98]

2.2.4.3 COBOL copybook Data Access tab

Specify the data transfer protocol so Data Services can access the COBOL copybook data file.

Select to use either FTP or a custom file transfer protocol method.

> → Tip
>
> Consider using a file location object instead of defining data access in the *Data Access* tab. A file location object allows greater variety of file transfer methods, and provides for editing outside of the data flow.

If you do not select the *FTP* or *Custom* checkboxes, the software assumes that the data file is on the same computer as the Job Server.

COBOL copybook option descriptions in the Data Access tab

Data Access tab option	Description
FTP	Specifies to use FTP to access the data file.
Host	Specifies the computer host name, fully qualified domain name, or IP address of the computer where the data file resides.
User	Specifies the FTP user name.
Password	Specifies the FTP user password.

Data Access tab option	Description
Directory	Specifies the directory that contains the COBOL copybook data file to import. If you include a directory path here, then enter only the file name in the *Name* field.
File name	Specifies the COBOL copybook data file name. You can use variables or wild cards (* or ?). If you leave *Directory* blank, then type the full path and file name, or browse to it.
Custom	Specifies to use a custom executable to access the data file.
Executable	Specifies the name of the program to read data file.
User	Specifies the user name.
Password	Specifies the password.
Arguments	Specifies custom program arguments, if applicable.

Parent topic: COBOL copybook file format [page 39]

Related Information

COBOL copybook Format tab [page 40]
COBOL copybook Data File tab [page 42]
COBOL copybook Field ID tab [page 46]
COBOL copybook Record Length Field tab [page 47]
File location object [page 98]

2.2.4.4 COBOL copybook Field ID tab

Create rules for associating records to schemas when the imported COBOL copybook has multiple schemas.

The *Field ID* tab appears only in the *Edit COBOL copybook* dialog box when the imported COBOL copybook has multiple schemas.

COBOL copybook option descriptions in the Field ID tab

Field ID tab option	Description
Use field <FIELD NAME> as ID	Specifies whether to set a value for the field selected in the top pane.
Edit	Changes the selected value in the *Values* pane to editable text.
Delete	Deletes the selected value in the *Values* pane.

Field ID tab option	Description
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.

Parent topic: COBOL copybook file format [page 39]

Related Information

COBOL copybook Format tab [page 40]
COBOL copybook Data File tab [page 42]
COBOL copybook Data Access tab [page 45]
COBOL copybook Record Length Field tab [page 47]

2.2.4.5 COBOL copybook Record Length Field tab

Identify the field that contains the length of the schema record and set an offset value in the *Record Length Field* tab.

COBOL copybook option descriptions in the Record Length Field tab

Record Length Field tab column	Description
Schema	Identifies the data schema in the copybook.
Record length field	Specifies a field (one per schema) that contains the record length from the dropdown list. The software populates the value based on the field you choose.
Offset	Specifies a value that, when added to the value in the *Record length field* option, results in the total record length. The default value for the offset is 4.

Parent topic: COBOL copybook file format [page 39]

Related Information

COBOL copybook Format tab [page 40]
COBOL copybook Data File tab [page 42]
COBOL copybook Data Access tab [page 45]

COBOL copybook Field ID tab [page 46]

2.2.5 Conditional

Conditionals are single-use objects that you use to define if/then/else logic in your work flow.

Define a conditional by specifying a condition and two logical branches. Place conditionals in a work flow. For complete information about conditionals, see the *Designer Guide*.

Conditional characteristics

Characteristic	Description
	Conditional icon.
Class	Single-use
Access	With a work flow diagram in the workspace, click the conditional icon in the tool palette and click in the workspace.
Description	A conditional implements if/then/else logic in a work flow.

Create a conditional [page 48]

To create a conditional, configure an if/then/else statement in the conditional editor.

2.2.5.1 Create a conditional

To create a conditional, configure an if/then/else statement in the conditional editor.

The Conditional editor contains the If, Then, and Else statement elements for completing a conditional statement.

Conditional elements

Conditional element	Description
If	Specifies the condition to evaluate. The IF statement is a Boolean expression that results in either true or false. Enter the IF statement directly in the IF text box, or click the *Ellipses* button to open the Smart Editor. You can use constants, functions, variables, parameters, and standard operators to construct the expression. **i Note** Don't put a semicolon (;) at the end of your expression in the *If* text box.
Then	Specifies the action to take when the If statement results in True. The action is a work flow to execute. To complete the *Then* pane, select a work flow from the *Work Flows* tab in the object library and drag it onto the *Then* pane.
Else	Specifies the action to take when the If statement results in False. The action is a work flow to execute. To complete the *Else* pane, select a work flow from the *Work Flows* tab in the object library and drag it onto the *Else* pane.

Conditional Name attribute

You name the conditional after you add it to your work space. Add the conditional to your workspace by right-clicking the conditional icon and selecting *Rename*.

To view the properties of the conditional, right-click the conditional icon and select *Properties*. Additionally, add a description for the conditional in the *Properties* dialog box.

For complete information about creating a conditional, see the *Designer Guide*.

Parent topic: Conditional [page 48]

Related Information

Smart Editor and the function wizard [page 1061]

2.2.6 Data flow

A data flow contains the objects that instruct the software to access, process, and load data.

Data flow characteristics

Characteristic	Description
	Data flow icon
Class	Reusable
Access	• In the object library, click the *Data Flows* tab. • With a work flow diagram in the workspace, click the data flow icon in the tool palette.
Description	A data flow extracts, transforms, and loads data. You can define parameters to pass values into the data flow. You can also define variables for use inside the data flow. When SAP Data Services executes data flows, it optimizes the extract, transform, and load requirements into commands to the DBMS and commands executed internally. Where it can, the software runs these operations in parallel.

For complete information about data flows, see the *Designer Guide*.

Data flow objects and attributes [page 50]
A data flow is an object that extracts, transforms, and loads data in SAP Data Services Designer.

Data flow properties, General options [page 52]
To control how the software executes the data flow, set options in the *General* tab of the Data Flow *Properties* dialog box.

2.2.6.1 Data flow objects and attributes

A data flow is an object that extracts, transforms, and loads data in SAP Data Services Designer.

Data flows contain a variety of object types based on the applicable data operations. A data flow has three categories of objects. Each category is further comprised of several object types.

Data flow object categories and object types

Data flow category	Object types
Source	A source can contain the following object types: • Files, including JSON and XML • Tables • Messages, such as JSON and XML (real-time jobs only) • Documents • Predefined template tables
Target	A target can contain the following object types: • Files, including JSON and XML • Tables • Messages, including JSON and XML (real-time jobs only) • Outbound messages • Documents • Nested schemas • Templates • Template tables
Transform	One or several Data Services transforms based on the desired outcome of the data flow. The Query transform is the most commonly used transform

→ Tip

The software generates SQL code for table sources in data flows. View the SQL code to find areas to improve your data flow design.

i Note

If you delete a data flow from the object library, you delete all calls to the data flow, even when it is in a different project or work flow. The software replaces all calls to the deleted data flow with the following icon that indicates the calls are no longer valid in the workspace:

Parent topic: Data flow [page 50]

Related Information

Data flow properties, General options [page 52]

2.2.6.2 Data flow properties, General options

To control how the software executes the data flow, set options in the *General* tab of the Data Flow *Properties* dialog box.

Access the *Properties* dialog box by right-clicking the data flow and selecting *Properties*. There are three tabs in the *Properties* dialog box: *General*, *Attributes*, *Class Attributes*. The following table describes the options in the *General* tab.

Option	Description
Name	Specifies the data flow name. Name the data flow when you create it. Edit the name in the *Properties* dialog box.
Execute only once	Specifies to run the data flow only once. Data Services executes only the first occurrence of the data flow and skips subsequent occurrences of the data flow in the job. The default is not selected. Under certain circumstances, the *Execute only once* option does not function as expected: • Parallel processing • Other options that override these settings such as *Recover as a unit* work flow option and *Enable recover* job option.
Use database links	Specifies whether the software uses database linking when processing the data flow. A database link is a one-way communication path from one database server to another. The option is selected by default.
Degree of Parallelism	Specifies how many times each transform in the data flow replicates for use on a parallel subset of data. The default value is *Default*, which is 0. If you leave the default setting, you can set a global degree of parallelism setting to affect all data flows for a given job server. If you set this option to something other than the default, the setting overrides any global setting.

Option	Description
Cache type	Specifies the type of caching to use. Caching data improves performance when the data flow contains joins, groupings, sorting, lookup tables, comparisons, and so on. • *In Memory*: Uses available memory for storing data. Select when your data flow processes a small amount of data that can fit into the available memory. • *Pageable*: Uses pageable cache when memory-intensive operations such as Group By and Order By exceed available memory.
Description	Optional. Enter a description for the data flow.
Bypass	Specifies that the software bypasses the data flow during execution. Appears only when there are two or more data flows in a work flow. • *{No Bypass}*: Disables bypassing this data flow. *{No Bypass}* is the default. • Substitution parameter Selecting to bypass a dataflow saves time when you test a job. Run specific data flows instead of all of them. Create a substitution parameter to use with the *Bypass* option. For example, create a substitution parameter $$BYPASSEnable with a value of Yes.

For more information about enhancing data flow processes through the property settings, see the *Performance Optimization Guide*.

Parent topic: Data flow [page 50]

Related Information

Data flow objects and attributes [page 50]
Degree of parallelism
Cache data sources

2.2.7 Datastore

Datastores contain connection information to a data source such as a database, application, Web service, or an adapter.

Datastore characteristics

Characteristic	Description
	Datastore icon.
Class	Reusable
Access	In the object library, click the *Datastores* tab.
Description	A datastore provides a connection to a data source such as a database. Through the datastore connection, SAP Data Services can import descriptions of the data source such as its metadata. When you specify tables as sources or targets in a data flow, the software uses the datastore to determine how to read data from or load data to those tables. In addition, some transforms and functions require a datastore name to qualify the tables they access.

There are several types of datastores. For example, create a datastore to connect to a supported database, create a datastore to connect to a specific application such as SAP applications, or create a datastore to connect to an application using an adapter. The *Designer Guide* contains descriptions for all datastore types. Also see specific supplement for details about specific types of datastores. For example, see the *Supplement for Adapters* to read about using adapter datastores.

Datastore properties, General options [page 54]
Datastore properties contain information such as name and description, attributes, and class attributes.

Datastore types [page 55]
When you create a datastore, you select a datastore type, which determines the remaining options that you complete.

Database types [page 56]
The dropdown list of *Database types* contains different values based on the option you choose from the *Datastore type* dropdown list.

2.2.7.1 Datastore properties, General options

Datastore properties contain information such as name and description, attributes, and class attributes.

The following table describes the options in the *General* tab of the datastore *Properties* dialog box.

Option	Description
Name	Specifies the name of the datastore. This name appears on the datastore icon in the object library and in the calls to the object. You cannot change the name of a datastore in the *Properties* dialog box..

Option	Description
Description	Optional. Describes the datastore. Enter text to describe the datastore.

> **i Note**
>
> If you delete a datastore from the object library, also remove references to the datastore from the following locations:
>
> - Source or target tables in data flows in which you used the datastore.
> - The lookup and key_generation functions and Key_Generation, History_Preserving, Table_Comparison, and SQL transform references.

Parent topic: Datastore [page 54]

Related Information

Datastore types [page 55]
Database types [page 56]

2.2.7.2 Datastore types

When you create a datastore, you select a datastore type, which determines the remaining options that you complete.

Select a datastore type in the *Datastore types* dropdown list in the Datastore editor. Access the datastore editor when you create a new or edit an existing datastore. There are many datastore types, therefore, there isn't one source for learning how to create a datastore. Therefore, the following table contains the datastore types and where you can find details about creating and configuring the datastore type.

Datastore type	More information
Adapter	*Supplement for Adapters*
Database	*Designer Guide* and other supplements, like the *Data Services Supplement for Big Data* and *Supplement for Hadoop*, based on type
Google BigQuery	*Supplement for Google BigQuery*
JDE One World JDE World	*Supplement for J.D. Edwards*
Oracle Applications	*Supplement for Oracle Applications*
PeopleSoft	*Supplement for PeopleSoft*

Datastore type	More information
Replication Server CDC	*Reference Guide*, SAP Sybase ASE
SAP Applications	*Supplement for SAP*
SAP BW Source SAP BW Target	*Supplement for SAP*
SAP DQM Microservices	*Supplement for SAP*
Siebel	*Supplement for Siebel*
Web Service REST Web Service SOAP	*Integrator Guide*

Parent topic: Datastore [page 54]

Related Information

Datastore properties, General options [page 54]
Database types [page 56]

2.2.7.3 Database types

The dropdown list of *Database types* contains different values based on the option you choose from the *Datastore type* dropdown list.

Database type descriptions

Database type	Description
Database	Enables Data Services to access, read from, and write to supported database types.
Adapter	Enables Data Services to access data from applications for which you have an adapter.
Memory	Enhances processing performance of data flows executing in real time jobs. Memory datastores are database datastore that use *Memory* as the *Database type*.
Persistent cache	Enables Data Services to store a large amount of data and create cache tables that multiple data flows can share. Persistent cache datastores are database datastores that use *Persistent cache* as the *Database type*.

Database type	Description
Linked	Enables Data Services to use linked databases to enhance performance by pushing down operations to a target database using a target datastore.
Web service	Enables Data Services to access external web service based data sources.
Mainframe interface	Attunity Connector datastore enables Data Services to access mainframe data sources.

Find more information about creating datastores in the *Designer Guide*.

Parent topic: Datastore [page 54]

Related Information

Datastore properties, General options [page 54]
Datastore types [page 55]

2.2.8 Document

Documents describe a data schema for some adapter datastores.

Document characteristics

Characteristic	Description
	Document icon.
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 documentation in the *Supplement for Adapters* for specific information about the options available for documents.

2.2.9 DTD

A DTD (document type definition) contains a description of a data schema for an XML message or file.

DTD characteristics

Characteristic	Description
	DTD icon.
Class	Reusable
Access	In the object library, click the *Formats* tab, then open the Nested Schemas category.
Description	A DTD (document type definition) describes the data schema of an XML message or file. **i Note** XML Schemas can be used for the same purpose. Data flows can read and write data to messages or files based on a specified DTD format. You can use the same DTD to describe multiple XML sources or targets. To use DTDs, import metadata into SAP Data Services. You can import a DTD directly, or you can import an XML document that contains or references a DTD. During import, the software converts the structure defined in the DTD into the nested-relational data model (NRDM).

DTD Properties [page 58]
DTD properties include values that you cannot change, and values that you can enable, disable, or edit.

Attributes for DTDs [page 61]
DTDs have two groups of attributes: Column and nested table.

Supported DTD components [page 62]
The software supports declarations in a DTD component based on the content model.

Related Information

XML schema [page 231]

2.2.9.1 DTD Properties

DTD properties include values that you cannot change, and values that you can enable, disable, or edit.

Context-click a DTD file under Nested Schemas in the *Formats* tab of the object library and select *Properties*. There are two tabs in the DTD properties dialog: *General* and *Format*.

General tab

Property	Description
Name	Read only. Specifies the name of the DTD file format. The DTD name appears in the object library under the Nested Schemas category of the *Formats* tab. It appears on sources and targets for XML files or messages that reference this format in data flows.
Description	Describes the DTD. You enter a description when you create the DTD format file.

Format tab

Property	Description
File location	Optional. Specifies the name of an existing file location object. The file location object contains file transfer protocol information. It specifies the local and remote servers that the transform uses to safely transfer data: • from remote to local server as a source • from local to remote server as a target

Property	Description
Delete file after transfer	Specifies to delete the source or target file on the server after the software has transferred the data. **As a source:** • Selected: Deletes the local file copy after the software loads it as a source in the data flow. > **i Note** > Not applicable for SCP file transfer protocol. • Not selected: Saves the local file copy after the software loads it as a source in the data flow. **As a target:** • Selected: Deletes the local file copy after the software transfers the generated output data from the local server to the remote server. > **i Note** > Not applicable for SCP file transfer protocol. • Not selected: Saves the local file copy after the software transfers the generated output data from the local server to the remote server. Available only when you select specify a file for *File location*.
File name	Specifies the DTD file name. Select the DTD file name from the drop list or click `<select file>` to browse for the DTD file. For added flexibility, select a variable for this option.
Imported from	Read only. Specifies the full path to the DTD file.
DTD file	Read only. Specifies whether the DTD file was originally imported from a DTD file. • Selected: DTD format was originally imported from a DTD file. • Not selected: DTD format was originally imported from an XML file with an associated DTD.
Root element name	Read only. Specifies the name of the primary node of the XML that the DTD defines. SAP Data Services imports only elements of the format that belong to this node or any sub nodes.

Parent topic: DTD [page 58]

Related Information

Attributes for DTDs [page 61]
Supported DTD components [page 62]

2.2.9.2 Attributes for DTDs

DTDs have two groups of attributes: Column and nested table.

View the DTD attributes tab in the *Properties* dialog. Data Services supports the DTD attributes described in the following table:

Supported column attribute	Description
Enumeration	Contains a list of all possible values separated by vertical bars. For example: "Red | White | Blue Green | Magenta". The software cuts off a string display at 256 characters.
Fixed Value	Indicates the only value the column can have.
Native Type	A string that specifies the original data type of the element or attribute in the DTD.
Required	Indicates whether mapping is required for this column. • *Yes*: Mapping is required for this column. • *No*: Mapping is not required because the column is optional. If you do not map the column, validation allows missing mapping expressions for the column at runtime. The software substitutes NULL for missing values.
XML Type	Track whether the column was an element or attribute in the original DTD.

Supported nested table attribute	Description
Any One Column	If choice (for example, "white | black | almond"), then SAP Data Services sets the value of Any One Column to YES. If sequence (for example, "first, last, street, city, state") then the software sets the value to NO. If both are present in the DTD, the value is set to NO.
Minimum Occurrence	If ()* then minimum occurrence is set to zero. If ()+, then minimum occurrence is set to 1. Indicates minimum number of rows that can be in the table.

Parent topic: DTD [page 58]

Related Information

DTD Properties [page 58]
Supported DTD components [page 62]

2.2.9.3 Supported DTD components

The software supports declarations in a DTD component based on the content model.

SAP Data Services reads DTD components. To process the data it reads in an XML file or message, the software translates the DTD into its internal nested relational database model.

Each component in the DTD is defined by its `<content model>`. The software supports the declarations in XML content models as follows:

DTD declaration		Supported
DOCTYPE	SYSTEM	Supported.
	PUBLIC	No support.
Declarations	ELEMENT	Supported. The XML Type attribute of the corresponding column is set to Element.
	ATTRIBUTE	Supported. The XML Type attribute of the corresponding column is set to Attribute.
	ENTITY	Supported. All entity references that can be expanded are expanded. Any that cannot be expanded cause an error at the time that you import the DTD.
	NOTATION	No support. Elements defined with NOTATION cause an error at the time that you import the DTD.
Content model	ANY	No support. Elements defined with ANY cause an error at the time that you import the DTD.
	EMPTY	Supported.
	#PCDATA	Supported. Converts to varchar(1024).
	MIXED	Supported.
Attribute declarations	CDATA	Supported. Converts to varchar(1024).
	ID	Supported. Converts to varchar(1024). When producing XML output, the software cannot ensure that ID values are unique throughout the schema.
	IDREF	Supported. Converts to varchar(1024).
	IDREFS	Supported. Converts to varchar(1024).

DTD declaration		Supported
	NMTOKEN	Supported. Converts to varchar(1024).
	NMTOKENS	Supported. The software treats multiple tokens as a single token with more than one space-separated values.
	Enumerated value	Supported. The software saves the enumerated values in the Enumeration attribute of the column. When producing XML output, the software checks to ensure that the value generated by the real-time job for the corresponding column is from the list; if no value is generated, the software uses the provided default value. If you validate XML messages against the DTD in a real-time job and the message includes a value that is not allowed based on the DTD, the XML source produces an error.
Attribute declaration defaults	#REQUIRED	Supported. The software saves this as the Required attribute with a value of `YES` and as data type `varchar(1024)`. When 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 attribute declaration as the Required attribute with a value of `NO` and as the data type `varchar(1024)`. When producing XML output, the software provides whatever value is generated in the data flow for the corresponding column, including a NULL value (`''`).
	#FIXED (default value)	Supported. The software saves this attribute declaration as the Fixed Value attribute and the data type `varchar(1024)`. When producing XML output, the software checks to ensure that the value generated by the real-time job for the corresponding column is from the list; if no value is generated, the software uses the provided default value.
	Default values	Supported. Converts to data type `varchar(1024)`. When producing XML output, the software uses the default value when 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.

Operator	Description	Supported
Comma (,)	Sequence	Supported. The software uses the ordering given in the DTD as the column ordering in the internal data set. Also the Any One Column attribute is set to a value of `NO`.
Vertical bar (|)	Choice (either or)	Supported. The software uses the ordering given in the DTD as the column ordering in the internal data set. Also the Any One Column attribute is set to a value of `YES`. The internal data set must include both options.
Plus (+)	One or more	Supported. Saved as nested table attribute Minimum Occurrence with a value of "1". The internal data set must include options for one or more elements.
Asterisk (*)	Zero or more	Supported. 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 ()*.

Parent topic: DTD [page 58]

Related Information

DTD Properties [page 58]
Attributes for DTDs [page 61]

2.2.10 Excel workbook format

Excel workbook characteristics

Characteristic	Description
	Excel workbook format icon.
Class	Reusable
Access	In the object library, click the *Formats* tab.
Description	An Excel workbook format describes the structure defined in an Excel workbook (denoted with a .xls extension). You store format templates for Excel data ranges in the object library. You use the template to define the format of a particular source in a data flow. SAP Data Services accesses Excel workbooks as sources only (not as targets).

Import or Edit Excel workbook format options [page 65]
Complete options in the top pane of the Import or Edit Excel workbook dialog box to name the format file and to configure the schema.

Excel Workbook Format tab [page 66]
Set options in the *Format* tab to define the parameters of the Excel workbook format.

Excel Workbook: Data Access tab [page 69]
Set options in the *Data Access* tab to specify how the software accesses the Excel file.

2.2.10.1 Import or Edit Excel workbook format options

Complete options in the top pane of the Import or Edit Excel workbook dialog box to name the format file and to configure the schema.

The top pane of the *Import Excel Workbook* or *Edit Excel Workbook* dialog box displays the following information:

Excel workbook option	Description
Format name	Displays the name of the Excel workbook format template in the object library.
Schema pane	Specifies the schema. When you create or edit an Excel Workbook schema, use the options in the *Schema* pane to manually define the schema. If you import a sample Excel Workbook, edit the schema that imports.

Parent topic: Excel workbook format [page 65]

Related Information

Excel Workbook Format tab [page 66]
Excel Workbook: Data Access tab [page 69]

2.2.10.2 Excel Workbook Format tab

Set options in the *Format* tab to define the parameters of the Excel workbook format.

Format tab option descriptions

Excel workbook format option	Description
File location	Specifies an existing file location object. Specifying a file location object is optional. The file location object contains file transfer protocol information, such as FTP or SFTP, and local and remote server information. The software uses this information to transfer a copy of the Excel workbook file from a remote server to a local server. The data flow uses the data in the local server as the source. **i Note** When this option is enabled, the following options are disabled: • *Directory* option • *Data Access* tab
Delete file after transfer	Specifies whether the software deletes the file from the local server after loading as a source. Available only when you set a file location object. • Selected: Deletes file from local server after the software loads file as a source in a data flow. • Not selected: Does not delete file from local server after the software loads file as a source in a data flow.
Directory	Specifies directory of the Excel workbook that contains the schema definition. Disabled when you enter a file location object name in *File Location*. If you specify variables or wild cards for this option, the software disables the *Import schema* button.

Excel workbook format option	Description
File name	Specifies a file name for the Excel workbook file that contains the schema definition. The Excel workbook file contains the schema definition. If you specify variables or wild cards for this option, the software disables the *Import schema* button. When you use an Excel workbook as a source, and your source is in cloud storage, specify the full remote path and file name. Include all subfolders in the full remote path. Supported cloud storages include Amazon S3, Azure Blob Storage, Azure Data Lake Store, Hadoop File System, and Google Cloud Storage.

Excel workbook format option	Description
Access method	Specifies whether to import a named range as defined in Excel or a range from a worksheet: • *Named range*: Select a range from the dropdown list. The list contains named ranges that you defined in the excel workbook. Alternately, type the name of the range > **i Note** > If your entries for *Directory* and *File name* do not contain a valid Excel workbook file, the dropdown list is empty. > **i Note** > The dropdown does not display noncontiguous named ranges, even if they exist in the Excel workbook. • *Worksheet*: Select a specific worksheet from the dropdown list. To designate the worksheet name as an ordinal number, select *Number*. The dropdown list contains all worksheets in the Excel workbook. Alternately, type the name of the worksheet. > **i Note** > If your entries for *Directory* and *File name* do not contain a valid Excel workbook file, the dropdown list is empty. > **i Note** > If the worksheet name starts with a dollar sign ($), Data Services treats it as a variable. To use a worksheet name that starts with a dollar sign, prefix (escape) it with a backslash (\). The *Worksheet* option includes the following *Range* options: • *All fields*: Includes everything in the worksheet from the uppermost left-hand populated cell to the lowest, right-hand cell. • *Custom range*: Select cell range from the worksheet by clicking the button at right. Launches a new instance of Excel. Use Excel notation, for example A1:B3. • *Extend range*: Select when you use a custom range to extend the custom range selection to the end of the worksheet.

Excel workbook format option	Description
	i Note Your system must have Excel installed for the software to open a new instance of Excel. **i Note** After you select the cell range and go back to Data Services, the software closes the Excel instance.
Code page	Specifies the character encoding the software uses for the character data in the Excel workbook. For more information about character encoding, see the Locales and Multi-Byte Functionality section in the *Reference Guide*.
Use first row values as column names	Specifies to use the first row of values as column names. • Select: Software uses the first row of values as column names. • Not selected: Software names the columns. For example `F1` is the first field, `F2` is the second field, and so on.

Parent topic: Excel workbook format [page 65]

Related Information

Import or Edit Excel workbook format options [page 65]
Excel Workbook: Data Access tab [page 69]

2.2.10.3 Excel Workbook: Data Access tab

Set options in the *Data Access* tab to specify how the software accesses the Excel file.

→ Tip

Consider using a file location object instead of defining data access in the Data Access tab. A file location object allows greater variety of file transfer methods, and provides for editing outside of the data flow.

Data access option	Description
FTP	Specifies whether the software uses FTP to access the data file. • Select: Software uses FTP to access the data file. • Not Selected: Software does not use FTP to access the data file **i Note** If you do not select both the *FTP* and *Custom* options, the software assumes that the data file is on the same computer as the Job Server.
Host	Specifies the host computer name. Enter a fully qualified domain name or the IP address of the computer where the data file resides.
User	Specifies the FTP user name.
Password	Specifies the FTP password associated with the value in *User*.
Directory	Specifies the directory that contains the Excel workbook data file for import. If you include a directory path here, then enter only the file name in the *File Name* option.
File name	Specifies the name of the Excel workbook data file. Browse for the file or enter the file name. Optionally use variables or wild cards (* or ?). If you do not complete the *Directory* option, then include the full path and file name for *File Name*.
Custom	Specifies to use a custom executable to access the Excel workbook data file. **i Note** If you do not select both the *FTP* and *Custom* options, the software assumes that the data file is on the same computer as the Job Server.
Executable	Specifies the program name that accesses and reads the Excel workbook data file.
User	Specifies the user name for the specified executable.
Password	Specifies the password associated with the value in *User*.
Arguments	Specifies additional program arguments.

Parent topic: Excel workbook format [page 65]

Related Information

Import or Edit Excel workbook format options [page 65]
Excel Workbook Format tab [page 66]

2.2.11 File format

File format objects contain metadata structures for flat files.

File format characteristics

Characteristic	Description
	File format icon.
Class	Reusable
Access	In the object library, click the *Formats* tab.
Description	A file format object describes the structure of a flat (ASCII) file. Use the file format object for templates. A file format template is a generic description that you can use for multiple data files. Store file format templates in the object library. Use the templates to define the format for a particular source or target file in a data flow.

File format properties and modes [page 72]
: Use the *File Format Editor* to set properties for file format templates, source files, and target files.

File format general options [page 73]
: Set options in the *General* group to define information about the file format such as join ranking, caching options, and parallel processing options.

File format data file options [page 78]
: Enter information about the data file in the *Data file* group of the *File Format Editor*.

File format delimiters options [page 83]
: Delimiters indicate where an element of a file format starts and ends for each record.

File format default options [page 86]
: SAP Data Services provides a group of options in the File Format editor where you can set default options, such as date and time formats.

File format input and output options [page 88]
: Provide instructions for how SAP Data Services processes specific aspects of the file, such as beginning and ending of file markers, on input and output.

Custom Transfer options for file formats [page 91]
: When you use a third party program to transfer the data from your file format, set options in the *Custom Transfer* group .

Locale options for file formats [page 92]
To specify the default language and code page to use for a file format, set options in the *Locale* group in the file format edditor.

File format error handling [page 93]
To specify how the software handles errors and warnings when processing data from the file format, set options in the *Error Handling* group in the File Format editor.

Flat file format source information [page 97]
Include information about the flat file source in the output file by setting options in the *Source Information* group of the File Format Editor.

2.2.11.1 File format properties and modes

Use the *File Format Editor* to set properties for file format templates, source files, and target files.

Available properties vary by the mode in which you open the *File Format Editor*. The following table describes the modes.

File Format Editor modes

Mode	Description
New mode	Create a new file format template.
Edit mode	Edit an existing file format template.
Source mode	Edit the file format of a source file in a data flow.
Target mode	Edit the file format of a target file in a data flow.

The file format editor has three work areas or panes:

Work areas in File Format Editor

Work area	Location	Description
Properties—values	Left pane of *File Format Editor*	Options that specify the type of file, location, delimiters, default format, and other settings.
Column attributes	Upper right pane of *File Format Editor*	Define the columns in the format file including field name, data type, size, and so on. **i Note** Field specific settings that you make in the column attribute area override the settings in the properties—values area.

Work area	Location	Description
Data preview	Lower right pane of *File Format Editor*	Displays sample data from the file specified in properties—values work area using the settings you make in the column attributes work area. See how changes you make in the column attribute area affect the sample data.

Parent topic: File format [page 71]

Related Information

File format general options [page 73]
File format data file options [page 78]
File format delimiters options [page 83]
File format default options [page 86]
File format input and output options [page 88]
Custom Transfer options for file formats [page 91]
Locale options for file formats [page 92]
File format error handling [page 93]
Flat file format source information [page 97]

2.2.11.2 File format general options

Set options in the *General* group to define information about the file format such as join ranking, caching options, and parallel processing options.

Many options that you choose in the *General* group determine which options SAP Data Services includes for completion.

General group option descriptions

File format option	Description
General group	

File format option	Description
Type	Specifies the format of the data file. Options include: • Delimited • Fixed width • SAP transport • Unstructured text • Unstructured binary For information about the SAP transport file format, see the *Supplement for SAP*.
Name	Specifies the name for the file format. The file format name identifies this file format in the object library.
Join rank	Specifies the rank of the data file relative to other tables and files joined in a data flow. Enter a positive integer. The default value is *0*. Data Services joins sources with higher join ranks before joining sources with lower join ranks. If the data flow includes a Query transform, the join rank specified in the Query transform overrides the *Join rank* specified in the file format editor. For new jobs, specify the cache only in the Query transform editor. For more information about join ranking, see "Source-based performance options" in the *Performance Optimization Guide*.
Cache	Specifies whether Data Services reads the data from the source and load it into memory or pageable cache. • *Yes*: Always caches the source unless the source is the outer-most source in a join. *Yes* is the default setting. • *No*: Never caches the source. If the data flow includes a Query transform, the cache setting specified in the Query transform overrides the *Cache* setting specified in the *Format File Editor* tab. For new jobs, specify the cache only in the Query transform editor. For more information about caching, see "Using Caches" in the *Performance Optimization Guide*.

File format option	Description

Applicable schema	Specifies whether the schema of a delimited file format is adaptable or fixed.
	• *Yes*: Indicates that the schema is adaptable. The actual file can contain fewer or more columns than indicated by the file format.
	> **i Note**
	> Ensure that the selected column and row delimiters do not appear inside the actual data.
	  • If a row contains fewer columns than expected, Data Services loads null values into the columns missing data.
	  • If a row contains more columns than expected, Data Services ignores the additional data.
	• *No*: Indicates that the schema is fixed. Data Services requires the number of columns in each row to match the number of columns specified in the file format. *No* is the default setting.

File format option	Description
Data alignment	Specifies how Data Services processes fixed-width file formats. • *Character*: Indicates that Data Services measures the fields in the file as characters, so processing is in character semantics. > **Example** > When you define a column as varchar(30), the column has 30 characters regardless of the number of bytes for each character. • *Byte*: Indicates that Data Services measures the field in the file as bytes, so processing is in byte semantics. > **Example** > When you define a column as varchar(30), the column has 30 bytes regardless of the number of characters. A transform reads bytes in a column based on the length of the column. For multibyte code pages, a character can be more than 1 byte. When trailing bytes of the last character of a column exceeds the length of the column, a transform continues reading until it reaches the total length of the character. A transform loader writes the number of bytes that are specified for the length of the column. The loader truncates trailing bytes if they exceed the length of the column and code page is multibyte. This behavior creates a potential for partial characters in the file. Under this situation, a transform logs errors in the reader.
Make port	Indicates whether the file is an embedded data flow port. • *Yes*: Data Services makes a source or target file an embedded data flow port. • *No*: Data Services does not make a source or target file an embedded data flow port. *No* is the default setting. For more information about embedded data flows, see *Designer Guide*.
Rows to read	Indicates the maximum number of rows in the data file that Data Services reads. The default is blank. If you enter a value of zero or a negative number, Data Services reads all rows.

File format option	Description
Custom transfer program	Specifies that Data Services uses a third-party file transfer program. When you select *Yes* for this option, Data Services displays additional custom transfer program options listed after the Input/Output group. → Tip Consider using a file location object instead of a custom transfer program. A file location object allows greater variety of file transfer methods, and enables you to edit outside of the data flow. To use a file location object, select *No* for this option and select the name of an existing file location object in the *Location* option under Data Files group.
Skip error handling	Specifies whether to disable the Error Handling group of options. • *Yes*: Disables the Error Handling options. • *No*: Does not disable the Error Handling options. *No* is the default setting.
Parallel process threads	Specifies the number of threads for parallel processing. Enter a positive integer. Parallel processing improves performance by maximizing CPU usage on the Job Server computer. For example, if you have four CPUs, enter 4 for this option. Set this option to *{none}* if you use the data file to process USPS certification tests. For more information, see "File multi threading" in the *Performance Optimization Guide*. i Note If you set the number of rows for *Batch size (rows)* to a nonzero positive integer, Data Services ignores a setting greater than 1 for the *Parallel process threads* option. *Batch size (rows)* is in the *Data File* group, and it applies to flat files, XML files, and JSON files.

Parent topic: File format [page 71]

Related Information

File format properties and modes [page 72]
File format data file options [page 78]
File format delimiters options [page 83]
File format default options [page 86]
File format input and output options [page 88]
Custom Transfer options for file formats [page 91]
Locale options for file formats [page 92]
File format error handling [page 93]
Flat file format source information [page 97]

2.2.11.3 File format data file options

Enter information about the data file in the *Data file* group of the *File Format Editor*.

Data File group option descriptions

File format option	Description
Data File group	

File format option	Description
Location	Specifies file location or file location name information based on the mode. Available options based on modes. **During design**, new or edit modes: • *Local*: Specifies that the data file is on the local machine that runs SAP Data Services Designer. • *Job Server*: Specifies that the data file is on the machine that runs the Job Server. Enter the absolute path to the file or files. **During execution**, source or target modes: • *Job Server*: Specifies that the data file or files are on the Job Server machine that executes the job. Enter the absolute path to the file or files. ℹ Note If you used different files during design, change the file before execution. Change the file using the *Root Directory* and *File Properties* options. • *File location object*: Specifies a file location object location. The file location object contains FTP, SFTP, SCP, and Local protocol information and locations for local and remote servers.

File format option	Description
Batch size (rows)	Specifies the number of rows to load at a time into the temporary local file before uploading to your Google Cloud Storage (GCS) account. This option controls the amount of local disk space used during the upload process. After each load from the local file to GCS, Data Services deletes the rows in the local file. After Data Services deletes the rows in the local file, it loads the next set number of rows to the local file and continues in this manner until it loads all rows to GCS. Set to a positive integer or zero (0): • **Positive integer:** Loads the set number of rows to the local file before uploading to GCS. Uses local disk space equal to the size of the set number of rows. ℹ Note Upload is single-threaded so Data Services ignores any setting greater than 1 for the *Parallel process threads* option in the *General* group. ℹ Note Upload does not use compression so Data Services ignores a setting of *gzip* for *Compression type* in the GCS file location. • **0:** Disables the option. Uploads all rows to the local temporary file before uploading to GCS. Uses the settings in the file format and in the GCS file location to upload rows from the local file to GCS. Uses local disk space equal to the size of the entire file. Zero (0) is the default setting. Applicable only for uploading to GCS and only for flat files, XML files, and JSON files. Applicable for target mode only. ℹ Note If you set *Location* or *File location* to a location and file name for a non-GCS file location object, Data Services ignores the setting in *Batch size (rows)*.

File format option	Description
Root directory	Specifies the directory where the data file is located. If you specify a directory name for this option, then enter only the file name for the *File name* option. If you leave this option blank, then enter a file name that includes the full path name in the *File name* option. For added flexibility, use a variable for this option.
File name	Specifies the file on which you base the file format description. The File Format Editor uses this file to display sample columns and rows in the left pane of the editor. If you want to define the file format without using a sample file. leave this option blank. In source and target modes, specifies the location of the actual file for this source or target. For added flexibility, you can use the following settings for this option: • A variable that you define to use a specific file with the full path name. Use variables to specify file names that you cannot otherwise enter, such as file names that contain multibyte characters. • A list of files, separated by commas, or a file name containing a wild card. In this case, Data Services reads all these files as a single source. For more information about listing multiple files, see "Reading multiple files at one time" in the *Designer Guide*. When you use a flat file as a source, and your source is in cloud storage, specify the full remote path and file name. Include all subfolders in the full remote path. Supported cloud storages include Amazon S3, Azure Blob Storage, Azure Data Lake Store, Hadoop File System, and Google Cloud Storage.
Read subfolders	Specifies whether Data Services reads the file or files in any nested subfolders for unstructured file formats. • *Yes*: Reads file or files and all nested subfolders for unstructured file formats. • *No*: Does not read any nested subfolders for unstructured file formats.

File format option	Description
Skip empty files	Specifies whether Data Services ignores empty files for unstructured file formats. • *Yes*: Skips empty files. • *No*: Reads empty files and completes the file with NULL data.
Number of files to read	Specifies the maximum number of files Data Services reads when it reads unstructured file formats. Enter a positive integer to limit the number of files Data Services reads to that number. To have the transform read all files, enter a zero or a negative value. The default setting is *{none}*.
Delete file	Specifies whether Data Services should delete the contents of the target file before loading generated output data to the target file: • *Yes*: Deletes the contents of the target file before loading the newly generated output data from the data flow. • *No*: Appends the newly generated output data to existing data in the target file. If you selected a file location object for *Location*, the setting affects data in your remote server: • *Yes*: Deletes the contents of the remote target file before loading the newly generated output data from the local file. • *No*: Appends the newly generated output data from the local file to the target file in the remote server.

File format option	Description
Delete file after transfer	Specifies whether to delete the file in the local server when you use the file for source or target. Applicable when you choose a file location object for *Location*. **Source** • *Yes*: Deletes local file after software reads data from local file into data flow. • *No*: Saves local file after software reads data from local file into data flow. **Target** • *Yes*: Overwrites target local file with generated output data. • *No*: Appends generated output data to the target local file. Applies only when file location object defines FTP or SFTP protocol.

Parent topic: File format [page 71]

Related Information

File format properties and modes [page 72]
File format general options [page 73]
File format delimiters options [page 83]
File format default options [page 86]
File format input and output options [page 88]
Custom Transfer options for file formats [page 91]
Locale options for file formats [page 92]
File format error handling [page 93]
Flat file format source information [page 97]

2.2.11.4 File format delimiters options

Delimiters indicate where an element of a file format starts and ends for each record.

In the file format editor, specify the delimiter options for the applicable elements such as columns and rows. Select the delimiters in the Delimiter group of options when you create or edit a file format.

Delimiter group option descriptions

File format option	Description
Delimiters group	
Column	Specifies the delimiter to use between columns in your data file. Possible values include: • Tab • Semicolon • Comma • Space • Any character sequence • Any ASCII characters including nonprinting characters **Note** 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. **Note** Column data can't include the designated column or row delimiters unless you specify a text delimiter. **Example** For example, if you specify a comma as your column delimiter, none of the strings in the file can contain commas. However, if you specify a comma as the column delimiter and a single quote as the text delimiter, the software correctly differentiates between commas that are for column delimiter and commas in strings.

File format option	Description
Row	Specifies the delimiter to use between rows. Specifies where one row ends and the next row begins. Possible values include: • {new line} • {Windows new line} • {Unix new line} • Any character sequence • Any ASCII characters including nonprinting characters **i Note** 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.
Row within text string	Specifies how the software interprets the row delimiter when it appears in a text string. • *Character*: Interprets as characters in the text string. • *Row delimiter*: Interprets as row delimiter and defines a row within the text string.
Text	Specifies the delimiter to use between text strings. The software interprets all characters between the first and second occurrence of the delimiter as a single text string. Characters can include the characters you specify as column delimiters. The software uses the setting in *Row within text string* to interpret row characters.

Parent topic: File format [page 71]

Related Information

File format properties and modes [page 72]
File format general options [page 73]
File format data file options [page 78]
File format default options [page 86]
File format input and output options [page 88]
Custom Transfer options for file formats [page 91]

Locale options for file formats [page 92]
File format error handling [page 93]
Flat file format source information [page 97]

2.2.11.5 File format default options

SAP Data Services provides a group of options in the File Format editor where you can set default options, such as date and time formats.

Set default values for your file format in the *Default Format* group in the File Format Editor.

File format Default Format option descriptions

File format option	Description
Default Format group	
Escape Character	Specifies a character sequence to indicate that the software should ignore the normal column delimiter. Enter any character sequence or select *{none}*. **i Note** Never use the characters that follow the escape character as column delimiters. Set for new and edit modes.
NULL Indicator	Specifies a character sequence to indicate to the software that the data is NULL. Enter any character sequence or select *{none}*. **i Note** The software ignores NULL indicators specified in the file format for blob columns. Set for new and edit modes.
Ignore Row Markers	Specifies a character sequence that appears at the beginning of specific rows. When the software reads the file, or when it automatically creates metadata, and it encounters the row markers, it ignores the row and moves to the next row. Enter any character sequence or select *{none}*. Set for new, edit, and source modes.

File format option	Description
Blank Padding	Specifies whether the software adds blanks before or after a field. Applies to fixed width files only. • *Leading*: Adds blanks to the left of (before) the data. • *Trailing*: Adds blanks to the right of (after) the data. Set for new and edit modes.
Blank Trimming	Specifies whether the software deletes extra blank spaces from fields. Applies to fixed width files only. • *Leading*: Deletes blanks to the left of (before) the data. • *Trailing*: Deletes blanks to the right of (after) the data. • *Both*: Deletes blanks to the left and right of the data. • *None*: Leaves blanks in the field. Set for new and edit modes.
Date	Specifies the date format for reading and writing date values to and from the file. • *yyyy.mm.dd* • Other valid combination Set for new and edit modes.
Time	Specifies the time format for reading and writing time values to and from the file. • *hh24:mi:ss* • Other valid combinations Set for new and edit modes.
Date-time	Specifies the date/time format for reading or writing date/time values to and from the file. • *yyyy.mm.dd hh24:mi:ss* • Other valid combinations Set for new and edit modes.

File format option	Description
Validate Decimal Data	Specifies whether the software validates decimal data to ensure that the decimal format is valid. The software validates decimal data by converting data in delimited files to decimal data type. Delimited files only. • *Yes*: Software converts data in delimited files to the decimal data type. Software converts even in the case of lazy decimal convertion. > **i Note** > Lazy decimal conversion is an optimization where the data for columns of type DECIMAL is stored as STRING data type in rows and is converted to internal DECIMAL format only when it is used in an operation. • *No*: Software does not validate decimal data to ensure that the decimal format is valid. Set for target mode only.

Parent topic: File format [page 71]

Related Information

File format properties and modes [page 72]
File format general options [page 73]
File format data file options [page 78]
File format delimiters options [page 83]
File format input and output options [page 88]
Custom Transfer options for file formats [page 91]
Locale options for file formats [page 92]
File format error handling [page 93]
Flat file format source information [page 97]

2.2.11.6 File format input and output options

Provide instructions for how SAP Data Services processes specific aspects of the file, such as beginning and ending of file markers, on input and output.

Complete the options in the *Input/Output* group when you create a new or edit an existing file format.

File format Input and output option descriptions

File format option	Description
Input/Output group	
Style	Specifies the format for the start and end of the file. • *Headers*: Specifies that the file includes headers that either have column headings, or other data. • *BOF/EOF*: Specifies that the file includes a beginning of file (BOF) and end of file (EOF) markers. BOF indicates the start of data in the file. EOF indicates the end of data in the file. **i Note** The value you choose for this option changes the remaining options.
Skipped Rows	Specifies the number of rows to skip before processing data. **Example** If your file contains 3 rows of header data, enter 3. Enter a positive integer. Applicable when you select *Headers* for *Style*.
Skip Row Header	Specifies that the software skips the first row of data because it contains header information. The software first skips any rows indicated in *Skipped Rows*. • *Yes*: Skips the first row of data after skipping the number of rows specified in *Skipped Rows*. • *No*: Doesn't skip the first row of data after skipping the number of rows specified in Skipped Rows. Applicable when you select *Headers* for *Style*.
Write Row Header	Specifies whether to write column headings in the first row of the output file. • *Yes*: Writes column headings in the first row of the output file. • *No*: Doesn't write column headings in the first row of the output file. Applicable when you select *Headers* for *Style*.

File format option	Description
Write BOM	Specifies the writing of byte order mark (BOM) characters into the file when file format uses UTF-8 and UTF-16 code pages. • *Yes*: Writes BOM characters into a file that uses either UTF-8 or UTF-16 code page in which byte order isn't otherwise defined. • *No*: Doesn't write BOM characters into the file. **i Note** For UTF 16 file, the software assumes the file is UTF-16be unless you select *Yes* for *Write BOM*. Applicable when you select *BOF/EOF* for *Style*.
BOF Marker	Specifies a string that marks the start of data in the file. • Enter a unique character sequence including a blank space or an empty string. • Select *{none}*. Applicable when you select *BOF/EOF* for *Style*.
EOF Marker	Specifies the string that marks the end of data in the file. • Enter a unique character sequence including a blank space or an empty string. • Select *{none}*. Applicable when you select *BOF/EOF* for *Style*.

Parent topic: File format [page 71]

Related Information

File format properties and modes [page 72]
File format general options [page 73]
File format data file options [page 78]
File format delimiters options [page 83]
File format default options [page 86]
Custom Transfer options for file formats [page 91]
Locale options for file formats [page 92]
File format error handling [page 93]
Flat file format source information [page 97]

2.2.11.7 Custom Transfer options for file formats

When you use a third party program to transfer the data from your file format, set options in the *Custom Transfer* group .

> → Tip
>
> Consider using a file location object instead of using a custom transfer program. A file location object allows greater variety of file transfer methods, and provides for editing outside of the data flow.
>
> If you plan to use a custom transfer program to transfer the file format data, complte the options in the Custom Transfer group when you create or edit a file format.

To enable the *Custom Transfer* group, make sure to set the *Custom Transfer Program* in the *General* group to *Yes*.

File format Custom Transfer options

File format option	Description
Custom Transfer group	
Program Executable	Required. Specifies the custom transfer program file executable name.
User Name	Specifies the user name to access the program executable.
Password	Specifies the password associated to the value in User Name.
Arguments	Flags any arguments that you create in your custom transfer program. **Example** Flag an argument that you created in the third party transfer program to indicate security or compression mechanisms in the program. Or you link connection data to your transfer program flags.

Parent topic: File format [page 71]

Related Information

File format properties and modes [page 72]
File format general options [page 73]
File format data file options [page 78]
File format delimiters options [page 83]

File format default options [page 86]
File format input and output options [page 88]
Locale options for file formats [page 92]
File format error handling [page 93]
Flat file format source information [page 97]

2.2.11.8 Locale options for file formats

To specify the default language and code page to use for a file format, set options in the *Locale* group in the file format edditor.

The following table describes the locale options to set when you create or edit a file format.

File format option	Description
Locale group	
Language	Specifies the human language in which the software stores and processes file data. Select the applicable 3 character language abbreviation from the dropdown list. **i Note** The 3 character abbreviations are ISO 639-2/T standard.
Code Page	Specifies the code page that the file uses. Select the applicable code page from the dropdown list. **Example** The Japanese code page contains ASCII, Greek, Cyrillic, and Japanese characters, and supports the English, Greek, Russian, and Japanese languages.

Parent topic: File format [page 71]

Related Information

File format properties and modes [page 72]
File format general options [page 73]
File format data file options [page 78]

File format delimiters options [page 83]
File format default options [page 86]
File format input and output options [page 88]
Custom Transfer options for file formats [page 91]
File format error handling [page 93]
Flat file format source information [page 97]

2.2.11.9 File format error handling

To specify how the software handles errors and warnings when processing data from the file format, set options in the *Error Handling* group in the File Format editor.

Access the *Error Handling* group when you create or edit a file format.

Error Handling option descriptions

File format option	Description
Error Handling group	
Log data conversion warnings	Specifies whether the software includes data type conversion warnings in the error log. • *Yes*: Includes data type conversion warnings in the error log. *Yes* is the default setting. • *No*: Does not include data type conversion warnings in the error log. Applicable for new, edit, and source modes.
Log row format warnings	Specifies whether the software includes row format warnings in the error log. • *Yes*: Includes row format warnings in the error log. *Yes* is the default setting. • *No*: Does not include row format warnings in the error log. Applicable for new, edit, and source modes.

File format option	Description
Log warnings	Specifies whether the software logs warnings for unstructured file formats. • *Yes*: Logs warnings for unstructured file formats. *Yes* is the default setting. • *No*: Does not log warnings for unstructured file formats. **i Note** Option appears only when you select *Unstructured Text* for *Type*. The software includes warnings for the following situations: • No files in the specified directory • No files match specified filter • Skipping irregular files on UNIX. For example, a FIFO, symbolic link, character or block device, or UNIX socket. • Empty file and *Skip empty files* set to *Yes* Applicable for new, edit, and source modes.
Maximum warnings to log	Specifies the maximum number of warnings the software logs. • Positive integer greater than 0. • *{no limit}* Applicable when you set *Log data conversion warnings* or *Log row format warnings* to *Yes*. Applicable for new, edit, and source modes.
Capture data conversion errors	Specifies whether the software captures data type conversion errors for flat file sources. • *Yes*: Captures data type conversion errors for flat file sources. • *No*: Does not capture data type conversion errors for flat file sources. *No* is the default setting. Applicable for new, edit, and source modes.

File format option	Description
Capture row format errors	Specifies whether the software captures row format errors for flat file sources. • *Yes*: Captures row format errors for flat file sources. *Yes* is the default setting. • *No*: Does not capture row format errors for flat file sources. Applicable for new, edit, and source modes.
Capture file access errors	Specifies whether the software captures file access errors for flat file sources. • *Yes*: Captures file access errors for flat file sources. *Yes* is the default setting. • *No*: Does not capture file access errors for flat file sources. Applicable for new, edit, and source modes.
Capture string truncation errors	Specifies whether the software captures string truncation errors for flat file sources. • *Yes*: Captures string truncation errors for flat file sources. • *No*: Does not capture string truncation errors for flat file sources. *No* is the default setting. Applicable for new, edit, and source modes.
Maximum errors to stop job	Specifies the maximum number of invalid rows the software processes before stopping the job. • Integer greater than 0. • *{no limit}*. *{no limit}* is the default setting. Applicable when *Capture data conversion errors* or *Capture row format errors* are set to *Yes*. Applicable for new, edit, and source modes.
Write error rows to file	Specifies whether the software writes invalid rows to an error file. • *Yes*: Writes error rows to error file. Also specify *Error file root directory* and *Error file name*. • *No*: Does not write error rows to error file. *No* is the default setting. Applicable for new, edit, and source modes.

File format option	Description
Error file root directory	Specifies the location of the error file. • Directory path • Blank • Select a variable **i Note** If you enter a directory path for this option, enter only a file name for *Error file name* option. If you leave this option blank, enter the full path and file name in *Error file name*. Applicable only when you select *Yes* for *Write error rows to file*. Applicable for new, edit, and source modes.
Error file name	Specify the file name for the error file. • File name if you only entered the directory path for *Error file root directory*. • File name including full path if you left *Error file root directory* blank. • Blank • Variable **i Note** Set the variable to a specific file with full path name. Use variables to specify file names that you cannot enter such as file names that contain multibyte characters. Applicable for new, edit, and source modes.

Parent topic: File format [page 71]

Related Information

File format properties and modes [page 72]
File format general options [page 73]
File format data file options [page 78]
File format delimiters options [page 83]
File format default options [page 86]

File format input and output options [page 88]
Custom Transfer options for file formats [page 91]
Locale options for file formats [page 92]
Flat file format source information [page 97]

2.2.11.10 Flat file format source information

Include information about the flat file source in the output file by setting options in the *Source Information* group of the File Format Editor.

SAP Data Services includes the source information in the output file when you use the flat file as a source in a data flow. Sets source information when you create or edit a file format for a flat file.

Flat file format source information options

Flat file format option	Description
Source Information group	
Include file name column	Specifies whether the software includes a column in the output file for source file name. • *Yes*: Adds a column to output file to hold source file name. When you select Yes, complete the remaining options in the *Source Information* group. • *No*: Does not add a column to output file and does not include source file name in output. *No* is the default setting. You may want to include the source file name in the output file under the following circumstances: • You specified a wildcard character to read multiple source files at one time. • You load from different source files on different runs.
Column name	Specifies a column name for the source file information.
Column size	Specifies the column size in number of characters for the source file name column. The default is 100 characters. If you do not set a large enough size, the software truncates the contents from the left.

Flat file format option	Description
Include path	Specifies whether to include the full path name of the source file. *Yes*: Includes the full path with the source file name in the output file. *No*: Does not include the full path with the source file name in the output file. *No* is the default setting.

Parent topic: File format [page 71]

Related Information

File format properties and modes [page 72]
File format general options [page 73]
File format data file options [page 78]
File format delimiters options [page 83]
File format default options [page 86]
File format input and output options [page 88]
Custom Transfer options for file formats [page 91]
Locale options for file formats [page 92]
File format error handling [page 93]

2.2.12 File location object

Create a file location object to establish a connection to a database with specific file transfer protocol settings.

File location characteristics

Characteristic	Description
	File location icon.
Class	Reusable
Access	Formats tab in the Designer object library.

Characteristic	Description
Description	File location objects contain details about a specific file transfer protocol so that the software safely transfers data from a remote server to a local server (as a source), or from local server to remote server (as a target). SAP Data Services supports the following protocols for file locations: • Local • FTP • SFTP • SCP • Google Cloud Storage • Azure Cloud Storage • Azure Data Lake Store • Amaazon S3 Cloud Storage • Hadoop This list can vary from version to version. The most up-to-date list for your version is in the dropdown list for *Protocol* in the File Location editor.

> **i Note**
>
> You can also set up custom secure file transfer when you create a file format. In the *File Format Editor*, select *Yes* for *Custom Transfer Program* under the General group of options, and complete the supporting options.

File location common options [page 99]
When you create a new file location object you set common options that determine the remaining options to complete.

Use built-in functions for file transfer [page 104]
Use file transfer built-in functions in scripts in a work flow to incorporate a file location object's information for file transfer protocol.

2.2.12.1 File location common options

When you create a new file location object you set common options that determine the remaining options to complete.

Some options in the *Create New File Location* window are common to more than one protocol type as indicated in the **Type of protocol** column in the following table.

Additional information:

To learn about file location options for such big data protocols as Google, Azure, Azure data lake store, and Amazon S3 cloud storage, see the *Data Services Supplement for Big data*. To learn about file location options for Hadoop, see the *Supplement for Hadoop*.

File location option	Description	Protocol type
Name	Specifies the name of the new file location. Follow your organization naming conventions.	All types
Connection group		
Protocol	Specifies the file transfer protocol for which you are creating this file format. Select a file transfer protocol from the dropdown list. The remaining options to complete are based on the protocol you choose in this option. For example, when you select *SCP*, options that describe your SCP connection details appear in the list of options. **i Note** The *DI Data Lake Storage* option is applicable to DI Hybrid adopters only.	All types
Host	Specifies the remote server name, fully qualified domain name, or IP address.	FTP, SFTP, SCP
Port	Specifies the port number for the remote server. Choose *<default>* to use the default port for the specified protocol.	FTP, SFTP, SCP
Hostkey Fingerprint	Specifies the code for the host computer expressed in hexadecimal string values.	SFTP, SCP

File location option	Description	Protocol type
Authorization Type	Specifies the authorization type to use to access the file location. • *Password*: Specifies to use a password to access the file location. • *Public Key*: Specify to use an SSH (UNIX secure shell) authorization information to access the file location. **i Note** Be prepared to supply the Private key file path, passphrase, and the public key file path.	FTP, SFTP, SCP
User	Specifies the authorized user for the remote server.	FTP, SFTP, SCP
Password	Specifies the password associated with the value in User for the remote server. Applicable when you choose *Password* for *Authorization Type*.	FTP, SFTP, SCP
SSH Authorization Private Key File Path	Specifies the file path to the private key file. Applicable when you choose *Public Key* for *Authorization Type*.	SFTP, SCP
SSH Authorization Private Key Passphrase	Specifies the passphrase for the private key file. Applicable when you choose *Public Key* for *Authorization Type*.	SFTP, SCP
SSH Authorization Public Key File Path	Specifies the file path to the public key file. Applicable when you choose *Public Key* for *Authorization Type*.	SFTP, SCP
Connection Retry Count	Specifies the number of times the computer tries to create a connection with the remote server when a previous connection is lost. The default is *10*.	All except Local
Connection Retry Interval	Specifies the amount of time in seconds that the computer waits until it retries to connect to the remote server when a previous connection is lost. Default is *10* seconds.	FTP, SFTP, SCP

File location option	Description	Protocol type
File System group		
Remote Directory	Specifies the file path for the remote server, excluding the server name, if applicable. Ensure that you have permission to this directory. If you leave this option blank, the software uses your local directory as the default file transfer protocol location. The software uses this directory as follows: • When the software uses an associated file format as a reader (source) in a data flow, it accesses the remote directory and transfers a copy of the data file to the local directory for processing. • When the software uses an associated file format as a loader (target) in a data flow, it accesses the local directory location and transfers a copy of the processed file to the remote directory.	FTP, SFTP, SCP, Amazon S3 Cloud Storage

File location option	Description	Protocol type
Local Directory	Specifies the file path of the local server for this file location object. The local server is on the same directory as the Job Server. Ensure that you have permission to the local directory. **i Note** If you leave this option blank, the software assumes that the default directory is `%DS_COMMON_DIR%/workspace`. The software uses this directory as follows: • When the software uses an associated file format as a reader (source) in a data flow, it accesses the remote directory and transfers a copy of the data file to the local directory for processing. • When the software uses an associated file format as a loader (target) in a data flow, it accesses the local directory location and transfers a copy of the processed file to the remote directory.	All types

Parent topic: File location object [page 98]

Related Information

Use built-in functions for file transfer [page 104]

File location objects
Amazon S3 file location protocol options
Azure Cloud Storage file location protocol
Azure Data Lake Storage
Google Cloud Storage file location
HDFS file location objects

2.2.12.2 Use built-in functions for file transfer

Use file transfer built-in functions in scripts in a work flow to incorporate a file location object's information for file transfer protocol.

You can use the built-in functions `copy_to_remote_system` and `copy_from_remote_system` in scripts in your work flow to move files from local to remote server or to move files from remote to local server. You can use the scripts alone or in combination with a file format as source or target.

Parent topic: File location object [page 98]

Related Information

File location common options [page 99]
Use built-in functions for file transfer protocol

2.2.13 Function

Use functions to process values for various purposes when you process jobs.

Function characteristics

Characteristic	Description
	Function icon.
Class	Reusable
Access	Access function objects in the following ways: • To access existing functions, click the *Functions* button in various object editors. • To access imported functions, open the Datastores tab in the object library, expand a datastore node, and expand the *Functions* node. • To access 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 • Stored procedures imported into SAP Data Services • Custom functions that you create • Validation functions that you import from SAP Information Steward or create locally

Function attributes [page 105]
Create and edit functions by setting function properties and attributes.

Related Information

Data Services Functions and Procedures [page 1064]

2.2.13.1 Function attributes

Create and edit functions by setting function properties and attributes.

The following table describes the tabs and options of the *Properties* dialog box.

Function attribute	Description
General tab	
Name	Specifies the name of the function. The function name appears in the function wizard and smart editor. The software also shows the function name when you use it in a script or expression.
Description	Specifies details about the function to identify the purpose or maybe the author. Enter descriptions when you create or import functions.
Function tab	
Function type	Specifies the type of function. Enter a function type when you create a custom function or import functions. For some functions, there is a function category.
Enable Parallel Execution	Specifies whether the software runs stored procedures and custom functions in parallel. If you select this option in the *Properties* dialog box *Function* tab, also enter a positive integer in the parent data flow *Properties* dialog box option for *Degree of parallelism*. For more information about parallel execution, see "Degree of parallelism" in the *Performance Optimization Guide*.
Validation function	Specifies whether the function is a validation function. Enable *Validation Function* in the *Function* tab of the *Properties* dialog box.

Parent topic: Function [page 104]

2.2.14 JSON

A JSON object contains a description of the data schema for a specific JSON message file.

JSON characteristics

Characteristic	Description
	JSON icon.
Class	Reusable
Access	In the object library, click the *Formats* tab, then open the Nested Schemas category.
Description	Data flows read and write data to messages or files based on a specified JSON format. Use the same JSON to describe multiple JSON sources or targets. To use JSON objects, import metadata into SAP Data Services. During import, the software converts the structure defined in the JSON into the nested relational data model (NRDM).

JSON Properties [page 106]

Set specific JSON properties to specify the software action when it processes the JSON in a data flow.

2.2.14.1 JSON Properties

Set specific JSON properties to specify the software action when it processes the JSON in a data flow.

Edit JSON objects by opening the JSON object Properties dialog.

JSON Properties option descriptions

JSON property option	Description
General tab	
Format Name	Specifies the name of the JSON object. The name is read only. This name appears in the object library under the Nested Schemas category of the *Formats* tab. Use the JSON file format in data flows as sources or targets (JSON files or messages) that reference the JSON format.
Description	Specifies a description for the JSON object. The description is read only.
Format tab	

JSON property option	Description
File Location	Optional. Specifies the name of an existing file location object. The file location object contains file transfer protocol information and local and remote server information to safely transfer data from remote to local server (source) and local to remote server (target). **i Note** When you enable this option, the software disables the *Directory* option and the *Data Access* tab.
Delete file after transfer	Specifies whether the software deletes the file from your local directory under the following circumstances. **As a source:** • *Yes*: Deletes local file after software reads data from local file into data flow. Not applicable for SCP file transfer protocol. • *No*: Saves local file after software reads data from local file into data flow. **As a target:** • *Yes*: Overwrites the target local file with generated output data. • *No*: Appends the target local file with generated output data.
File name	Specifies the JSON file name. Either select the JSON file name or browse to the JSON file. For added flexibility, use a variable for this option.
Imported from	Specifies the full path to the JSON file. The location is read only.

Parent topic: JSON [page 106]

2.2.15 Log

Create logs to record specific information about a job execution.

Log characteristics

Characteristic	Description
Class	Single use
Access	View logs run on a specific Job Server from Designer or from the Management Console Administrator: • In Designer: Open Designer and select the applicable repository. Open the project in the project area and open the *Log* tab located at the bottom of the project area. Expand the job tree. • In Management Console Administrator: Expand *Batch Jobs* and then *Repository*. Make sure that the repository is related to the applicable Job Server. 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. Both the logs listed in Designer and Management Console Administrator contains all logs for each execution. There are three types of logs: • Trace logs: Shows the execution progress through each component of the job. • Monitor logs: Quantifies the activities of each component in the job. • Error logs: Contains errors generated by the software and the source and target database management systems.

Trace logs [page 108]
Trace logs contain information about how SAP Data Services executed data through each object in the job.

Monitor logs [page 110]
View Monitor logs during and after job execution for details about the software processes for each object.

Error logs [page 112]
View error logs for a list of all errors generated during job execution.

2.2.15.1 Trace logs

Trace logs contain information about how SAP Data Services executed data through each object in the job.

View trace logs during and after job execution. The Trace log lists the following information:

- Process ID (PID)
- Thread ID (TID)
- Executed object type
- Start time for each event
- Event description

For unsuccessful jobs, use the trace log to see which components of a partially executed job completed or where an error occurred.

The following table describes the information in Trace logs.

Trace log information

Trace log entry	Description
PID	Indicates the process identification number of each thread executing.
TID	Indicates the thread identification number of each thread executing.
Type	Indicates the object being executed, such as a job, data flow or transform. The software labels generic job events as TRACE.
TimeStamp	Indicates the date and time the software generated the message for each thread.
Message	Gives a description of the events that are occurring or have occurred as the software executes each thread.

The trace log lists a prefix in the Type entry that indicates the type of information in the trace.

Trace type prefixes

Trace Type prefix	Description
ABAP	Traces messages for the ABAP query execution.
ADMIN	Administrative information like "server not responding" or "power failure."
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.

Trace Type prefix	Description
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.

Parent topic: Log [page 108]

Related Information

Monitor logs [page 110]
Error logs [page 112]
Checking system utilization

2.2.15.2 Monitor logs

View Monitor logs during and after job execution for details about the software processes for each object.

The monitor log lists the time spent in a given component of a job and the number of data rows that streamed through the component. Use the monitor log to help tune the performance of a job.

The following table describes the entries in the Monitor logs.

Monitor log information

Monitor log entry	Description

Path Name	Indicates which object is executing. The path name has the following format:
	`<dfname[_subdataflownumber]/objectname >`
	where
	• `<dfname>` is the name of the data flow
	• `<_subdataflownumber>` is the number of the sub data flow if SAP Data Services split the data flow into multiple sub data flows
	• `<objectname >` is the name of the source, transform, or target that the data flow is processing
	For example, the following path name is for the first sub data flow of a data flow named Orders_DF, and the object being processed is a source named Order Details:
	`/Orders_DF_1/ORDER DETAILS`
	The next example is a path name for the first sub data flow of the Orders_DF data flow, and the object being processed is temporary storage for data from Order Details. The 'TS' indicates that this object is temporary storage, and 'ORDERTEMP' is the name of the temporary storage specified in the Data_Transfer transform.
	`/Orders_DF_1/TS_ORDER DETAILS_ORDERTEMP`
	The next example is a path name for the second sub data flow of the Orders_DF data flow, and the object being processed is a query transform:
	`/Orders_DF_2/Query`

State	Indicates the current status of the execution of the object. If you view the log while the job is running, this value changes as the status changes. The possible values are START, PROCEED, and STOP. In a successfully run job, all of these values are STOP to indicate that they finished successfully.
Row Count	Indicates the number of rows the software processed through this object. The software updates this value based on the *Monitor sample rate (# of seconds)* set as a debug property.
Elapsed Time	Indicates the time in seconds since this object received its first row of data.

Monitor log entry	Description
Absolute Time	Indicates the time in seconds since the execution of this entire data flow, including all of the transforms began.

Parent topic: Log [page 108]

Related Information

Trace logs [page 108]
Error logs [page 112]

2.2.15.3 Error logs

View error logs for a list of all errors generated during job execution.

The *error log* lists errors generated by SAP Data Services, by the source or target DBMS, or the operating system. If you cannot access the error log because and the button appears grayed out, the job completed successfully.

The following table describes Error log information.

Error log information

Error log 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 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 following table describes the prefix listed in the Number entry that indicates the type of error.

Error number prefixes

Error log 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.

Error log Number prefix	Description
EML	E-mail errors.
FIL	Filespec errors.
OPT	Optimization errors.
PAR	Parser errors.
R3C	SAP connectivity errors.
R3S	SAP syntax errors.
REP	Repository errors.
RES	Resolver errors.
RUN	Runtime errors.
SCH	Job launcher errors.
SRV	Job Server errors.
SYS	System exceptions.
USR	User function errors.
VAL	Validator errors.
XRN	Transform errors.

Parent topic: Log [page 108]

Related Information

Trace logs [page 108]
Monitor logs [page 110]

2.2.16 Message function

Message function calls allow an adapter instance to collect requests and send replies.

Message function characteristics

Characteristic	Description
	Message function icon.
Class	Reusable
Access	In the object library, click the *Datastores* tab.

Characteristic	Description
Description	Available in certain adapter datastores, message functions can accommodate XML messages when properly configured. For more information about working with message functions, see the *Supplement for Adapters*.

Related Information

JMS adapter datastore
HTTP adapter datastore

2.2.17 Nested Schemas template

Use Nested Schema templates to create a JSON file or an XML file that matches a particular input schema.

Nested Schemas characteristics

Characteristic	Description
	Nested schemas template icon.
Class	Single use
Access	**Scenario** / **Description** Inserting as a target: Select the *Nested Schemas Template* icon in the tool palette, then click the data flow diagram in the workspace. Viewing options: Click the name of the Nested Schemas template in the workspace or in the project area to open the object editor.

Characteristic	Description
Description	The Nested Schemas template does not require and does not produce a corresponding JSON Schema, XML Schema, or DTD format. Likewise when a nested schema template generates XML, it does not create column attributes when they are present in the input schema. Use the Nested Schemas template to produce a JSON file, without having to predefine a JSON format. Also use the Nested Schemas template to produce an XML file, without having to predefine an XML format. Use a Nested Schemas template as a target in a batch or real-time job. **i Note** In a Nested Schemas template, the software converts all data types to varchar. After adding a Nested Schemas template to a data flow, specify the name and location of the file. In the Nested Schemas template target file editor, select either the *XML* or the *JSON* option as applicable. If your target is to a cloud storage, enter the full remote path and subfolders to the file location in your cloud storage account. This feature is applicable for Amazon S3, Azure Blog storage, Azure Data Lake Store, Hadoop File System, and Google Cloud Storage. **i Note** When you use Nested Schemas templates in real-time jobs, deselect the *Delete and recreate file* option in the target editor. The software selects this option by default when you create a Nested Schemas target.

Related Information

Target XML files, messages, and templates [page 207]

2.2.18 Outbound message

An outbound message is an XML-based hierarchical communication used by certain adapters.

Outbound message characteristics

Characteristic	Description
	Outbound message icon.
Class	Reusable

Characteristic	Description
Access	In the object library, click the *Datastores* tab.
Description	SAP Data Services can publish outbound messages 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. For more information about outbound messages, see the *Supplement for Adapters*.

Related Information

HTTP adapter datastore
JMS adapter datastore

2.2.19 Project

Use projects to group jobs so that they are available to access in one place.

Project characteristics

Characteristic	Description
	Project icon.
Class	Single use
Access	• Choose *Project* > *New*. • In the object library, click the *Projects* tab.
Description	A project is the highest level of organization offered by SAP Data Services. Open and expand a project node to view all jobs in the project. Projects have the following attribute: **Attribute** — **Description** Name — The name of the project. The project name displays in the *Projects* tab of the object library. The name also displays in the project area when you open the project. For more information about projects, see the *Designer Guide*.

2.2.20 Query transform

Use a Query transform in a data flow to retrieve a data set that satisfies the conditions that you specify.

Query transform characteristics

Characteristic	Description
	Query icon.
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 do the following: • 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. Nest and unnest data using a Query transform. The Query transforms has one attribute: **Attribute** — **Description** Name — The name of the object, which appears on the object in the diagram.

Related Information

Nested Data

Query transform [page 960]

2.2.21 Real-time job

A real-time job receives requests and sends replies in real-time and can run as a real-time service.

Real-time job characteristics

Characteristic	Description
	real-time job icon.
Class	Reusable
Access	Access a real-time job in the following ways: • In the object library, click the *Jobs* tab. • In the project area, right-click a project name and select *New Real-time Job*.
Description	A real-time job receives requests from an ERP system or Web application. It sends replies immediately after retrieving the requested data from a data cache or a second application. A real-time job consists of a set of objects that you 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. Create a real-time job in the Designer. Then configure it 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. Real-time jobs process messages and generates responses. The real-time source in the processing loop determines the message type. Likewise, the target determines the response format. For more details about real-time jobs, see the *Designer Guide*.

Real-time job attributes [page 119]
Real-time jobs have built-in attributes that are the same as a batch job.

Related Information

Real-time Jobs

2.2.21.1 Real-time job attributes

Real-time jobs have built-in attributes that are the same as a batch job.

Real-time job attribute descriptions

Attribute	Description
Name	The name of the object. This name appears on the object in the *Jobs* tab of the object library. The name also appears in the calls to the object.
Description	You or the job creator includes a description when the job is created.
Date created	The date on which the real-time job was created.

Parent topic: Real-time job [page 118]

2.2.22 Script

Use a script to assign values to local, global, or environment variables in a job or work flow.

Characteristic	Description
	Script icon.
Class	Single use
Access	With a work flow or job diagram in the workspace, click the script icon in the tool palette.
Description	Define the script using the SAP Data Services scripting language. Scripts can contain function calls, `IF` statements, `WHILE` statements, Assignment statements, and operators. Scripts have the following attribute: **Attribute** — **Description** Name — The name of the object. This name appears on the object in the diagram. For details about scripts, see the *Designer Guide*.

Related Information

Scripts

Data Services scripting language [page 1307]

2.2.23 Source

A source is an object, such as a table or file, from which SAP Data Services reads data in a data flow.

Characteristic	Description
Class	Single use
Access	Access a source in the following ways: • Drag the object from the object library onto the workspace and select *Make Source*. Applicable for documents, tables, template tables, or flat files. • Drag the object from the object library onto the workspace and select *Make XML message source* or *Make XML file source*. Applicable for XML messages or files as a source, 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. **i Note** Ensure that the source is connected to other objects in the data flow to be able to see the source options.
Description	In a batch job, a source can be a document, a file, a table, a previously defined template table, an XML file, a source-specific data flow, or an embedded data flow. 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. Options available for sources from adapter datastores depend on the adapter implementation. Thus, options vary by data source and adapter version. See the *Supplement for Adapters* for more information.

Types of table sources [page 121]
SAP Data Services supports many types of tables to use as sources.

COBOL copybook source options [page 127]
Complete the source options when you use the COBOL copybook as a source in a data flow.

Excel workbook source options [page 131]
When you add an Excel workbook as a source in a data flow, you complete options in the source editor.

JSON file source [page 133]
When you use a JSON file as a source, complete JSON-specific options in the source editor.

JSON message source [page 136]
When you use a JSON message as a source, complete the additional options in the source editor.

Persistent cache source [page 138]
Use a persistent cache table as a source only after you create a persistent cache table as a target.

Teradata source [page 139]
Complete the *Teradata Options* tab in the source editor when you use a Teradata source.

XML file source [page 142]

An XML file source has the same *Join rank* and *Make port* options as tables.

XML message source [page 143]

When you use an XML message as a source in a data flow, complete the XML message specific options in the Source editor.

2.2.23.1 Types of table sources

SAP Data Services supports many types of tables to use as sources.

The following table contains the types of tables supported by SAP Data Services with a brief description and where you can find more information.

Table types supported

Type	Description	More information
Amazon Redshift	The software supports the following features when you use an Amazon Redshift table as a source: • All Redshift data types • Optimized SQL • Basic push-down functions	*Data Services Supplement for Big Data*
Changed data capture (CDC)	To use a table from a CDC datastore, complete the options in the Source editor, in the *CDC Options* tab.	*Designer Guide* Reference Guide: CDC table source [page 123]
Flat file	A flat file source has the Join rank and Cache options in addition to the other General file format options listed for a source table. The flat file source options are the same in the Source File Editor as in the File Format Editor except for unstructured text and unstructured binary file. Unstructured text and binary file sources have an additional option named *Number of files to read*. You can associate an existing file location object to a flat file format. A file location object contains specific file transfer protocol information and local and remote server information, for file transfer between remote and local servers.	*Designer Guide*

Type	Description	More information
HP Vertica	To access HP Vertica tables, create an HP Vertica datastore.	*Data Services Supplement for Big Data*
SAP application sources	SAP application sources include Open Hub tables and SAP ODP sources.	*Supplement for SAP*
SAP Vora	To access SAP Vora tables, create an SAP Vora datastore.	*Data Services Supplement for Big Data*
Netezza	To access Netezza tables, create a Netezza database datastore.	Netezza table source [page 124]
PostgreSQL	To use PostgreSQL tables as sources and targets in data flows, create a PostgreSQL datastore.	*Data Services Supplement for Big Data*

CDC table source [page 123]
When a table is the result of a change data capture datastore, you can use it as a source table in a data flow.

Netezza table source [page 124]
Use a Netezza table as a source in a data flow and complete the options in the source editor.

Table source options to enhance performance [page 125]
When you use a table as a source in a data flow, configure specific source editor options to enhance processing performance.

Parent topic: Source [page 120]

Related Information

COBOL copybook source options [page 127]
Excel workbook source options [page 131]
JSON file source [page 133]
JSON message source [page 136]
Persistent cache source [page 138]
Teradata source [page 139]
XML file source [page 142]
XML message source [page 143]
Amazon Redshift database
Creating a file format
HP Vertica table source
Open Hub Table source
ODP source
SAP Vora table source options

2.2.23.1.1 CDC table source

When a table is the result of a change data capture datastore, you can use it as a source table in a data flow.

Complete the options in the *CDC Options* tab of the source editor. Options vary by CDC source database type.

CDC Options tab option descriptions

Option	Description
CDC subscription name	Specifies the name the software uses 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. Select a name from the dropdown list or type a new name to create a new subscription. Because you can use multiple CDC subscription names, ensure that the name is unique for the datastore, owner, and table name. **Example** For example, 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. Use multiple CDC subscription names to identify different users who read from the same imported CDC table. The subscription saves the position of each user. This value is required for the Microsoft SQL server Change Tracking and Replication Server methods.
Enable check-point	Specifies whether the software restricts CDC reads using check points. Check points provide a marker that indicates the rows the software reads each time the job runs. The software reads only the rows inserted into the CDC table since the last check point. This option is enabled by default.
Automatically delete rows after reading	For Microsoft SQL Server only. Specifies whether the software deletes rows after it reads the rows. • Selected: The software deletes the rows after it reads the rows. However, software behavior depends on the CDC method: Replication Server method or Changed Data Capture (CDC) method. • Not selected: The software does not delete any rows from the CDC table. **Descriptions of CDC methods:** • **Replication Server method:** If more than one data flow uses the same CDC table as a source, the software deletes only the rows that have been read by all readers. If any of those readers have the option cleared, the software does not delete any rows. • **CDC method:** The software deletes all rows of the CDC table regardless of whether other readers have finished reading from the table. Therefore, if more than one data flow uses the same CDC table as a source, only select this option for the data flow that executes last.

Option	Description
Get before-image for each update row	Species whether the software reads a before image during change data capture jobs, or only retrieves the after image. Applicable for specific databases that allow two images to be associated with an UPDATE row: Before image and after image. • Selected: Reads a before image during change data capture jobs. Selected is the default setting. • Not selected: Does not read before image during change data capture jobs, and retrieves only the after image.
Enable Auto Correct	Specifies whether the software corrects records in the target for transactions that are captured by the Replication Server during an initial load. Applicable for databases with a Replication Server CDC reader. This option is useful only when the loader can identify records in the target using keys from the source table. • Selected: Corrects records in the target for transactions that are captured by the Replication Server during an initial load. • Not selected: Does not correct records in the target for transactions that are captured by the Replication Server during an initial load. The Replication Server CDC reader does the following for auto-correct processing: • For all inserted records, creates a DELETE record followed by an INSERT record. • Creates a DELETE record for the before image, followed by DELETE and INSERT records for the after image for updated records that had a key value change.

Parent topic: Types of table sources [page 121]

Related Information

Netezza table source [page 124]
Table source options to enhance performance [page 125]
Changed Data capture

2.2.23.1.2 Netezza table source

Use a Netezza table as a source in a data flow and complete the options in the source editor.

Source option descriptions

Option	Description
Table name	Specifies the name of the table that you added as a source to the dataflow.

Option	Description
Table owner	Specifies the owner that you entered when you created the Netezza table.
Datastore name	Specifies the name of the Netezza datastore.
Database type	Set to Netezza by default. Specifies the database type that you chose when you created the datastore. You cannot change this option.
Table schema name	Specifies the name of the schema for the table that you added as a source to the dataflow.

Parent topic: Types of table sources [page 121]

Related Information

CDC table source [page 123]
Table source options to enhance performance [page 125]

2.2.23.1.3 Table source options to enhance performance

When you use a table as a source in a data flow, configure specific source editor options to enhance processing performance.

Each table type contains common source options to complete when the table is used as a source in a data flow. The following table contains descriptions of options in the source editor that enhance job performance.

Source options that enhance performance

Option	Description
Make Port	Specifies to make the source table an embedded data flow port.
Enable partitioning	Enables SAP Data Services to use the partition information in this table. If this option is selected, the software reads table data using the number of partitions in the table as the maximum number of parallel instances.

Option	Description
Join rank	Indicates the rank of the source relative to other tables and files joined in a data flow. The software joins sources with higher join ranks before it joins sources with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. Best practice is to specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0. For more information, see the Other Tuning Techniques section in the *Performance Optimization Guide*.
Cache	Indicates whether the software should read the required data from the source and load it into memory or pageable cache. Because an inner source of a join must be read for each row of an outer source, you might want to cache a source when it is used as an inner source in a join. Options are: • *Yes*: The source is always cached unless it is the outer-most source in a join. • *No*: The source is never cached. The default is *Yes*. Cache specified in the Query transform editor FROM tab overrides any cache specified in a source. Best practice is to specify the cache only in the Query transform editor.
Array fetch size	Indicates the number of rows retrieved in a single request to a source database. The default value is 1000. Higher numbers reduce requests, lowering network traffic, and possibly improve performance. Maximum value is 5000. This option is available for source tables from Amazon Redshift, DB2, Informix, ODBC, Oracle, Snowflake, and SQL Server datastores. When retrieving a column with an Oracle LONG data type, the software automatically sets *Array Fetch Size* to 1. If a column has an Oracle LONG data type, the software can only retrieve one row at a time.
Use overflow file	For Oracle source tables. Specifies to use a specific file for error handling. The software logs errors that occur while reading data 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 Datastore Editor for the tab.e

Parent topic: Types of table sources [page 121]

Related Information

CDC table source [page 123]
Netezza table source [page 124]
Create embedded data flows
Parallel Execution
Performance options for tuning system performance

2.2.23.2 COBOL copybook source options

Complete the source options when you use the COBOL copybook as a source in a data flow.

Data Services uses the source information to identify and process the COBOL copybook. Open the source editor in the data flow. The editor includes the following tabs:

- Source
- Field clauses
- Data File
- Data Access

COBOL copybook Source tab [page 127]
To control performance, error handling, and other processing information for the COBOL copybook source file, set options in the *Source* tab of the source editor

COBOL copybook Field Clauses tab [page 130]
To set attributes for a selected column in the COBOL copybook source file data layout, set options in the *Field Clauses* tab.

Parent topic: Source [page 120]

Related Information

Types of table sources [page 121]
Excel workbook source options [page 131]
JSON file source [page 133]
JSON message source [page 136]
Persistent cache source [page 138]
Teradata source [page 139]
XML file source [page 142]
XML message source [page 143]

2.2.23.2.1 COBOL copybook Source tab

To control performance, error handling, and other processing information for the COBOL copybook source file, set options in the *Source* tab of the source editor

When you place a COBOL copybook object as a source, open the source object and configure the source as described in the following tables.

COBOL copybook Source tab option descriptions

Source option	Description
Make port	Specifies whether to make the source table an embedded data flow port. For more information, see the Embedded Data Flows section in the *Designer Guide*.

COBOL copybook Performance tab options

Performance option	Description
Join rank	Specifies the rank of the source relative to other tables and files joined in a data flow. Enter a positive number. the default value is *0*. The software joins sources with higher join ranks before joining sources with lower join ranks. Join rank specified in the Query transform editor *FROM* tab overrides any join rank specified in a source. For new jobs, specify the join rank only in the Query transform editor. For more information about join ranks, see the Other Tuning Techniques section in the *Performance Optimization Guide*.
Cache	Indicates whether the software reads the required data from the source and loads it into memory or pageable cache. The software reads an inner source of a join for each row of an outer source. Therefore, best practice suggests that you cache a source when it is an inner source in a join. • *Yes*: The software always caches the source unless it is the outer-most source in a join. *Yes* is the default setting. • *No*: The software never caches the source. 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.

COBOL copybook Error handling tab options

Error handling option	Description
Log data conversion warnings	Specifies whether to include data type conversion warnings in the error log. • *Yes*: Includes data type conversion warnings in the error log. *Yes* is the default setting. • *No*: Excludes data type conversion warnings in the error log.

Error handling option	Description
Maximum warnings to log	Specifies a limit for the number of warnings the software logs. **i Note** This option is available only if you set the *Log data conversion warnings* option to *Yes*. The default is *{no limit}*.

COBOL copybook file name column options

File name column option	Description
Include file name column	Specifies whether to add a column that contains the source file name in the source output. • *Yes*: The software adds a column that contains the source file name in the source output. • *No*: The software does not add a column that contains the source file name in the source output. No is the default setting. When you select *Yes*, the software identifies the source file in the following situations: • You specified a wildcard character to read multiple source COBOL copybooks at one time • You load from different source copybooks on different runs
Modify	Specifies to modify the *File name column* and *Column size*.
File name column	Specifies the name of the column that holds the source file name. Applicable if the file name is included. The default value is *DI_FILENAME*.
Column size	Specifies the size (in characters) of the column that holds the source file name. Applicable if the file name is included. Defaults to *100*. If the size of the file name column is not large enough to store the file name, the software truncates the field from the left.
Include path	Specifies whether to include the full path name of the source file. The default value is *No*.

Parent topic: COBOL copybook source options [page 127]

Related Information

COBOL copybook Field Clauses tab [page 130]

2.2.23.2.2 COBOL copybook Field Clauses tab

To set attributes for a selected column in the COBOL copybook source file data layout, set options in the *Field Clauses* tab.

The Field Clauses tab consists of various clause types.

Option descriptions for the Field Clauses tab

COBOL copybook option	Description
Possible values	Specifies a value or multiple values to force the software to process only rows that contain the specified value or values. Separate multiple values with the pipe character (|). Click the *Ellipses* button to open the smart editor, if applicable.
Level	Specifies the level number assigned to the field in the source record definition. Values are 01-50.
Original name	Specifies the name of the field in the copybook.
Original picture	Specifies the PICTURE clause of the field in the copybook.
Original usage	Specifies the USAGE clause of the field in the copybook.
Min occurs	If this field is a part of an OCCURS group, specifies the minimum number of occurrences for this field.
Max occurs	If this field is a part of an OCCURS group, specifies the maximum number of occurrences for this field.
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 value.
Sign position	Specifies whether the sign is LEADING or TRAILING.
Multiply by	Specifies whether the software scales, multiplies, or divides the field by a certain number. **Example** If the field PICTURE clause is 9(5)P(3), the value of the field from the data file is multiplied by 1000.

Parent topic: COBOL copybook source options [page 127]

Related Information

COBOL copybook Source tab [page 127]
Smart Editor and the function wizard [page 1061]

2.2.23.3 Excel workbook source options

When you add an Excel workbook as a source in a data flow, you complete options in the source editor.

There are three tabs in the lower portion of the Excel workbook source editor: *Source*, *Format*, and *Data Access*. The following tables describe options in the *Source* tab.

Source option	Description
Make port	Specifies to make the source table an embedded data flow port.

Performance

Source option	Description
Join rank	Indicates the rank of the source relative to other tables and files joined in a data flow. The software joins sources with higher join ranks before joining sources with lower join ranks. Enter a positive integer. The default setting is *0*. If the data flow includes a Query transform, the join rank specified in the *Query* transform overrides the *Join Rank* specified in the *Source* tab. For new jobs, specify the join rank only in the Query transform editor. For more information about setting "Join rank", see "Source-based performance options" in the *Performance Optimization Guide*.
Cache	Indicates whether the software reads source data and loads it into memory or pageable cache. • *Yes*: Software always caches the source unless the source file is the outer-most source in a join. *Yes* is the default setting. • *No*: Software never caches the source. The software reads the inner source of a join for each row of an outer source, therefore, cache the source when you use it as an inner source in a join. If the data flow includes a Query transform, the cache setting specified in the *Query* transform overrides the *Cache* setting specified in the *Source* tab. For new jobs, specify the cache only in the Query transform editor. For more information about caching, see "Using Caches" in the *Performance Optimization Guide*.
Skip all empty rows	Specifies whether the software excludes empty rows when importing the Excel workbook data source. • Selected: The software does not import empty rows from the Excel workbook data source. • Not selected: The software imports empty rows from the Excel workbook data source, and adds Nulls.

Error handling

Source option	Description
Log errors to file	Specifies whether the software logs errors to the error log file. • Selected: Logs errors to error log file. • Not selected: Does not log errors to error log file. **i Note** The software adds Nulls to columns with runtime conversion errors.
Maximum errors to log	Specifies a limit to the number of errors the software logs. The default setting is *{no limit}*.
Error file directory	Specifies the location of the error log file.
Error file name	Specifies the name of the error log file.

Include file name column

Source option	Description
Include file name column	Specifies whether the software adds a column to the output that contains the name of the Excel workbook data source. • *No*: Does not add a column to the output file that specifies the name of the Excel workbook data source. *No* is the default setting. • *Yes*: Adds a column to the output file that specifies the name of the Excel workbook data source. Select *Yes* for the following situations: • You specified a wildcard character to read multiple source Excel workbook files at one time. • You load from different source workbook files on different runs.
Modify	Specifies whether the software allows you to modify the values in *File name column* and *Column size* when you enable *Include file name column*.
File name column	Specifies the name of the column that holds the name of the Excel workbook data source. Applicable when you enable *Include file name column*. The default file name is *DI_FILENAME*.
Column size	Specifies the size of the column that holds the name of the Excel workbook data source. Applicable when you enable *Include file name column*. The default size is *100*. If the size of the file name column is not large enough to store the file name, the software truncates the file name from the left.
Include path	Specifies the path of the Excel workbook data source in the column that holds the name of the Excel workbook data source. Applicable when you enable *Include file name column*. The default setting is *No*.

Other options

Source option	Description
Skip first	Specifies the number of rows the software skips from the top of the Excel workbook data source. The default is *{none}*.
Read total	Specifies the total number of rows the software reads from the Excel workbook data source. The software starts at the top of the worksheet or after the number of rows specified in *Skip first*, if applicable. The default is *{no limit}*.

Parent topic: Source [page 120]

Related Information

Types of table sources [page 121]
COBOL copybook source options [page 127]
JSON file source [page 133]
JSON message source [page 136]
Persistent cache source [page 138]
Teradata source [page 139]
XML file source [page 142]
XML message source [page 143]
Designer Guide: Embedded Data Flows
Excel workbook format [page 65]

2.2.23.4 JSON file source

When you use a JSON file as a source, complete JSON-specific options in the source editor.

A JSON file source has the same *Join rank* and *Make port* options as tables.

Source tab option descriptions

Option	Description
File Location	Optional. Specifies the name of a file location object related to this source. A file location object contains file transfer protocol information, such as FTP or SFTP, and local and remote server information. The software uses this information to transfer a copy of the JSON file from the remote server to the local server to use as a source in a data flow.
Delete file after transfer	Specifies whether the software deletes the file copy in the local server after the software loads it as a source in the data flow. Available when you select a file location object. • Selected: Deletes the local file copy after the software loads it as a source in the data flow. • Not selected: Saves the local file copy after the software loads it as a source in the data flow.
File	Specifies the location of a JSON-formatted file relative to the Job Server to use as the source. For convenience, use a variable for this option. Type an absolute path or a relative path to the JSON-formatted file. Or, for convenience, select a variable from the dropdown list. Ensure that the Job Server can access the location. **i Note** If your Job Server is on a different computer than Data Services Designer, you cannot select to browse from the *File* dropdown list. When you use a JSON file as a source, and your source is in cloud storage, specify the full remote path and file name. Include all subfolders in the full remote path. Supported cloud storages include Amazon S3, Azure Blob Storage, Azure Data Lake Store, Hadoop File System, and Google Cloud Storage.
Enable validation	Specifies whether the software compares the incoming data to the stored JSON schema. • Selected: Compares the incoming data to the stored JSON schema. If the incoming source is not valid, the software throws an exception. • Not selected: Does not compare the incoming data to the stored JSON schema.

Option	Description
Format name	Specifies the name of the JSON schema format file that you use as a source in the data flow. **i Note** If you specify a file location object in *File location*, the JSON schema format file that you reference here contains the file location information.
Include file name column	Specifies whether the software includes a column in the source output that contains the source JSON file name. • *Yes*: Includes a column in the generated output that contains the source JSON file name. • *No*: Does not include a column in the generated output for the source JSON file name. *No* is the default setting. Select *Yes* to identify the source JSON file under the following situations: • You specified a wildcard character to read multiple source JSON files at one time. • You load from different source JSON files on different runs. **i Note** When you set this option to *Yes*, complete the following options: • *File name column* • *Modify button* • *Column size* • *Include path*
File name column	Specifies the name of the source file column to include in the generated output. The default is DI_FILENAME. Applicable only when you enable *Include file name column*.
Modify	Specifies to open the *Column Properties* dialog box so you can modify the file name column and column size. Applicable only when you enable *Include file name column*.

Option	Description
Column size	Specifies the file name column size in characters. The default is *100*. If the size of the file name column is not large enough to store the file name, the software truncates the column from the left. Applicable only when you enable *Include file name column*.
Include path	Specifies to include the full path name of the source file in the file name column. • *Yes*: Includes the path name in the file name column. • *No*: Does not include the path name in the file name column. *No* is the default setting. Applicable only when you enable *Include file name column*.

Parent topic: Source [page 120]

Related Information

Types of table sources [page 121]
COBOL copybook source options [page 127]
Excel workbook source options [page 131]
JSON message source [page 136]
Persistent cache source [page 138]
Teradata source [page 139]
XML file source [page 142]
XML message source [page 143]
JSON [page 106]

2.2.23.5 JSON message source

When you use a JSON message as a source, complete the additional options in the source editor.

A JSON message source has the same *Make port* option as tables. The JSON message source has these options in addition to its read-only schema information:

Option	Description
Enable validation	Specifies whether the software compares the incoming data to the stored JSON schema. • Selected: Compares the incoming data to the stored JSON schema. If the incoming source is not valid, the software throws an exception. • Not selected: Does not compare the incoming data to the stored JSON schema. Consider the following situations when you set this option: • For real time jobs, the validation helps to ensure that the sample data is valid and well formed. • For production jobs, include appropriate error handling in either the Data Services job or the client application. The software should through an error if a data flow in a real time job receives data that does not validate against the imported format.
Test file	Specifies the location relative to the Job Server of a JSON-formatted file to use as the message source when you execute the job in test mode. For convenience, use a variable for this option. **i Note** If your Job Server is on a different computer than Data Services Designer, you cannot select to browse using the *Test file* dropdown list. Instead, type an absolute path or a relative path. Ensure that the Job Server can access the location.
Format name	Specifies the name of the JSON Schema format used in Data Services Designer.
Include file name column	Specifies whether the software includes a column in the source output that contains the source JSON file name. • *Yes*: Includes a column in the generated output that contains the source JSON file name. • *No*: Does not include a column in the generated output for the source JSON file name. *No* is the default setting. Select *Yes* to identify the source JSON file under the following situations: • You specified a wildcard character to read multiple source JSON files at one time. • You load from different source JSON files on different runs.
`<implied>`: *Join rank*	The JSON 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.

Parent topic: Source [page 120]

Related Information

Types of table sources [page 121]
COBOL copybook source options [page 127]
Excel workbook source options [page 131]
JSON file source [page 133]
Persistent cache source [page 138]
Teradata source [page 139]

XML file source [page 142]
XML message source [page 143]
JSON [page 106]

2.2.23.6 Persistent cache source

Use a persistent cache table as a source only after you create a persistent cache table as a target.

Persistent cache table source option descriptions

Option	Description
Make Port	Specifies whether to make the persistent cache source an embedded data flow port.
Join rank	Indicates the rank of the source relative to other tables and files joined in a data flow. The software joins sources with higher join ranks before it joins sources with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. Best practice is to specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0. For this option, SAP Data Services uses the same interpretation for both persistent cache tables and database tables. For more information, see the Other Tuning Techniques section in the *Performance Optimization Guide*.
Cache	Indicates whether the software should read the required data from the source and load it into memory or pageable cache. Because the software reads an inner source of a join for each row of an outer source, cache a source when it is used as an inner source in a join. Options are: • *Yes*: The source is always cached unless it is the outer-most source in a join. • *No*: The source is never cached. The default is *Yes*. Cache specified in the Query transform editor FROM tab overrides any cache specified in a source. Best practice is to specify the cache only in the Query transform editor. For this option, the software uses the same interpretation for both persistent cache tables and database tables.
Table name	Displays the table name that you entered when you created the persistent cache table. You cannot change this name.
Table owner	Displays the table owner that you entered when you created the persistent cache table. You cannot change this name.
Datastore name	Displays the datastore name that you entered when you created the persistent cache table. You cannot change this name.
Database type	Displays the *Persistent Cache* option. You cannot change this value.

Parent topic: Source [page 120]

Related Information

Types of table sources [page 121]
COBOL copybook source options [page 127]
Excel workbook source options [page 131]
JSON file source [page 133]
JSON message source [page 136]
Teradata source [page 139]
XML file source [page 142]
XML message source [page 143]
Use persistent cache

2.2.23.7 Teradata source

Complete the *Teradata Options* tab in the source editor when you use a 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.

Teradata source option descriptions

Option	Description
General	
Clean up bulk reader directory after export	Specifies to delete all files in the bulk reader directory after successfully exporting data. • *Yes*: Deletes all files in the bulk reader directory after successfully exporting data. • *No*: Saves all files in the bulk reader directory after successfully exporting data.
Minimum number of sessions	Specifies the minimum number of sessions required for the export driver job to continue. Enter a positive integer. The default setting is *1*.

Option	Description
Maximum number of sessions	Specifies the maximum number of connections to Teradata. Enter a positive integer greater than zero. The default is *1* per available AMP. Use this parameter with *Number of export operator instances* and *Parallel process threads* for performance tuning when reading from a Teradata source. For large volumes of data, more sessions allow 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 with *Maximum number of sessions* and *Parallel process threads* for performance tuning when reading from a Teradata source. Multiple export instances can improve performance. Ideally this value should equal the number of CPUs.
Tenacity hours	Specifies the number of hours the export driver attempts to log in when the maximum number of load and export operations are already running on the Teradata database. Enter a positive integer. The Default is *4* hours.
Tenacity sleep	Specifies the number of minutes the export driver pauses before attempting to log in when the maximum number of load and export operations are already running on the Teradata database. Enter a positive integer. The default is *6* minutes.
Data handling	
Block size	Specifies the block size in bytes when returning data to the client.
Data encryption	Specifies whether to enable full security encryption of SQL requests, responses, and data. • *Yes*: Enables full security. • *No*: Disables encryption.
Query band session	Specifies a user-defined query band expression to be set for every SQL session.
Notification	
Level	Indicates the level at which certain events are reported: • *Off*: No notifications. *Off* is the default setting. • *Low*: Notifications occur for Initialize, CLIv2/DBS Error, and Exit. • *Medium*: Notifications occur for all events except File or OUTMODE Open and Statement Fetch Begin and End. • *High*: Notifications occur for all events.

Option	Description
Method	Specifies the method for reporting events. • *None*: No event logging. *None* is the default setting. • *Message*: Sends events to a log, for example the EventLog on Windows. • *Exit*: Sends the events to a user-defined notify exit routine and to the system log.
User-defined string	Specifies a user-defined string that precedes all messages sent to the system log.
User-defined exit routine	Specifies the name of a user-defined notify exit routine.
Trace	
Level	Specifies the type or types 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. • *CLI*: Activates the tracing function for CLIv2-related activities. • *PX*: Activates the tracing function for activities involving the common library. • *Oper*: Activates the tracing function for driver-specific activities. • *Notify*: Activates the tracing function for activities related to the Notify feature.
Tracing file	For API mode, specifies the name of the external log file used for trace messages.
Miscellaneous	
Parallel process threads	Specifies the number of threads for parallel processing. Parallel processing can improve performance by maximizing CPU usage on the Job Server computer. Use this parameter with *Maximum number of sessions* and *Number of export operator instances* performance tuning when reading from a Teradata source. The data loads into buffers in Data Services, and the parallel process threads break these buffers into rows and columns. Ideally this number should be equal to the number of CPUs.
Logon mechanism	Specifies which logon mechanism to use: • *Kerberos 5* • *NT Lan Manager* • *Lightweight Directory Access Protocol* • *Simple and Protected GSSAPI Negotiation Mechanism*
Logon mechanism data	Specifies additional optional logon mechanism data.
AccountId	Optional. Specifies the account associated with the user name specified in the datastore.

Option	Description
Private log name	Specifies the name of the log file that the Teradata Parallel Transporter Logger inside the public log maintains. The private log contains all of the output provided by the export operator.

Parent topic: Source [page 120]

Related Information

Types of table sources [page 121]
COBOL copybook source options [page 127]
Excel workbook source options [page 131]
JSON file source [page 133]
JSON message source [page 136]
Persistent cache source [page 138]
XML file source [page 142]
XML message source [page 143]
Bulk loading and reading in Teradata
Parallel process threads for flat files
Teradata datastore
Teradata target table options [page 191]

2.2.23.8 XML file source

An XML file source has the same *Join rank* and *Make port* options as tables.

The options for setting up an XML file as a source are the same as the options in the XML Properties dialog.

Parent topic: Source [page 120]

Related Information

Types of table sources [page 121]
COBOL copybook source options [page 127]
Excel workbook source options [page 131]
JSON file source [page 133]
JSON message source [page 136]
Persistent cache source [page 138]

Teradata source [page 139]
XML message source [page 143]
XML Properties [page 226]

2.2.23.9 XML message source

When you use an XML message as a source in a data flow, complete the XML message specific options in the Source editor.

An XML message source has the same *Make port* option as tables. Additionally, there are options for XML message sources that are read only. Read only option values are from the XML Schema or DTD format.

XML message source option descriptions

Option	Description
XML test file	Specifies the location relative to the Job Server of an XML-formatted file to use as the message source. Applicable when you execute the job in test mode. Type the absolute path or relative path. Ensure that the Job Server can access the location. For convenience, use a variable for this option. **i Note** If your Job Server is on a different computer than the Data Services Designer, you cannot select the browse option from the dropdown list. Type the path instead.
Enable validation	Specifies whether the software compares the incoming message to the stored XML schema or DTD format. • Selected: Compares the incoming message to the stored XML schema or DTD format. For a real-time job, the software throws an exception if the incoming message is not valid. • Not selected: Does not compare the incoming message to the stored XML schema or DTD format. Consider the following situations when you set this option: • For real time jobs, the validation helps to ensure that the sample data is valid and well formed. • For production jobs, include appropriate error handling in either the Data Services job or the client application. The software should through an error if a data flow in a real time job receives data that does not validate against the imported format.
`<implied>` : *Join rank*	The XML message source is always the outer table in a join. You cannot change its join rank. This option is implied and does not appear in the editor.

Parent topic: Source [page 120]

Related Information

Types of table sources [page 121]
COBOL copybook source options [page 127]
Excel workbook source options [page 131]
JSON file source [page 133]
JSON message source [page 136]
Persistent cache source [page 138]
Teradata source [page 139]
XML file source [page 142]

2.2.24 Table

Tables contain data that you use for SAP Data Services processing.

Table characteristics

Characteristic	Description
	Table icon
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. To view column attributes, open the table, right-click the column name and select *Properties*.

Characteristic	Description
Description	A table contains columns and rows. The rows represent the records in the table. The columns represent the fields in each record. Import table metadata through the datastore object. Right-click a table name and select Properties to view the *Properties* dialog box. Along with the General, Class Attributes, and View Data tabs, the Properties dialog box contains table attribute tabs. Table Properties **Tab** / **Description** *Indexes* — Contains information about the table indices. Lists the primary index followed by any secondary index. Select an index to view the columns in that index under *Column*. *Partition* — Contains table partitioning information. Import partitions with a table or create metadata for tables within SAP Data Services. *Attributes* — Displays built-in table attributes.

Table attributes [page 145]
After you use a table in a batch job, the software populates the table attributes with information related to the job.

Column attributes for tables [page 147]
Open the *Column Properties* dialog box for a specific table column to view and edit attributes.

Related Information

Template table [page 216]

2.2.24.1 Table attributes

After you use a table in a batch job, the software populates the table attributes with information related to the job.

Built-in attributes for tables

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.

Table Attribute	Description
Description	A configurable description field.
Table_Usage	A configurable label field. Use it to mark a table as `fact` or `dimension` for example.
Total_Number_Of_ Rows_Processed	The number of rows loaded into the table in the last successful load.
Date_last_loaded	The time the table was last successfully loaded.
Number_Of_Rows_ Rejected	The number of rows rejected in the last successful load.
Number_Of_Inserts	The number of rows inserted in the last successful load.
Number_Of_Updates	The number of rows updated in the last successful load.
Date_Created	The date that the object was created.
Estimated_Row_ Count	A configurable estimate of the table size used in calculating the order in which tables are read to perform join operations; used for SAP tables only.
Number_Of_Deletes	The number of rows deleted in the last successful load.
Elapsed_Time_For_ Load	The time it took to load this table in the last successful load.
Table_Type	The type of datastore object for tables and hierarchies. Most often the value `TABLE` i displayed. However, the software might display the following values for SAP sources: BW master data transfer, BW transaction data transfer, BW hierarchy data transfer, SAP hierarchy.
SAP_Table_Class_ Name	Imported with SAP table metadata.
Loader_Is_Template_Table	If `YES`, indicates that the table is an internal, template table created in the software. Before running production jobs, execute the job to load the target table then right-click the template table in the object library or in a data flow and the software creates the table in your database and imports it.
SavedAfterCheckOut	If `YES`, indicates that you saved the table after it was checked out of the central repository. The software uses this information to determine whether to save the table in the central repository when it is checked in.
PartitionModified	If `YES`, indicates that you modified the partitions in this table using the software after you imported the table metadata.

Parent topic: Table [page 144]

Related Information

Column attributes for tables [page 147]

2.2.24.2 Column attributes for tables

Open the *Column Properties* dialog box for a specific table column to view and edit attributes.

Column Attribute	Description
Business Name	Specifies the logical business name for the column. You can edit this value.
Business Description	Specifies the business-level description of the column. You can edit this value.
Associated Dimension	Specifies an associated dimension for the column. Set this value only if *Column Usage* is set to `Detail`. Enter the value using the following format: `table.column`.
Acta autojoin	Generated by SAP Data Services. Not configurable.
Associated domain	Specifies the associated domain. Use for databases that use domains such as PeopleSoft.
Physical Name	Specifies the physical name of the column. Use for applications that allow logical names for a column such as Oracle Applications.

Parent topic: Table [page 144]

Related Information

Table attributes [page 145]

2.2.25 Target

A target object in a data flow receives generated output data from the processes in the data flow.

Target characteristics

Characteristic	Description
Class	Single-use
Access	View target options by opening the target object editor. Open the target in the applicable data flow to open the target editor. Also view target object properties by right-clicking the target in the data flow and selecting *Properties*.
Description	A target is an object to which SAP Data Services loads extracted and transformed data in a data flow. There are many other objects that you can use as a target in a data flow. For example, use a table as a target or an XML message.

Target objects [page 148]

There are many object types that you can include as a target in your data flow.

2.2.25.1 Target objects

There are many object types that you can include as a target in your data flow.

The following table contains objects that you can place in a data flow as a target. The icon appears in the tool pallet at the right of your workspace. The target object appears only when it is applicable to the data flow.

Icon for target object	Target object
	Document
	Flat file
	JSON file
	JSON message

Icon for target object	Target object
	Nested Schemas template
	Outbound message
	Table
	Template table
	XML file
	XML message

Documents and outbound messages are available only from adapter datastores. Options available for these targets depend on the adapter implementation. Thus, options vary by data source and adapter version. See the *Supplement for Adapters* more information

Parent topic: Target [page 147]

Related Information

Target files [page 149]
Target persistent cache tables [page 150]
Common Target Table editor options [page 152]
Types of target tables [page 167]
Target Data Transfer files and tables [page 200]
Target JSON files, messages, and templates [page 201]
Target XML files, messages, and templates [page 207]
Target flat files [page 213]

2.2.25.2 Target files

When you use a file as a target, a flat file, for example, the *Target File Editor* is similar to the file format editor.

In the *Target File Editor*, you cannot edit all properties of a particular target file. You can change some properties of the file format.

When you execute the job, the software provides validation errors when the schema defined in the file format does not match the schema that is input to the target. Use the validation error information to update the schema in the *Target File Editor*.

You can also change the name of the target file object using the object properties. Right-click the target file and choose *Properties*.

Parent topic: Target [page 147]

Related Information

Target objects [page 148]
Target persistent cache tables [page 150]
Common Target Table editor options [page 152]
Types of target tables [page 167]
Target Data Transfer files and tables [page 200]
Target JSON files, messages, and templates [page 201]
Target XML files, messages, and templates [page 207]
Target flat files [page 213]

2.2.25.3 Target persistent cache tables

Use a persistent cache table as a target in a data flow.

To add a persistent cache table as a target in a data flow, take one of the following actions:

- Drag a persistent cache template table from the object library onto the workspace.
- Click the template table icon in the tool palette and click the workspace. Choose a persistent cache datastore from the popup dialog.

> **i Note**
>
> You cannot update a persistent cache table. If the data within it changes, re-create the persistent cache table and load it.

Persistent cache target option descriptions

Option	Description
Make port	Specifies whether to make the target table an embedded data flow port. • Selected: Creates an embedded data flow port from the target. • Not selected: Does not create an embedded data flow from the target.
Table name	Specifies the name of the table. You entered the table name when you created the persistent cache table. You cannot change this value.

Option	Description
Table owner	Specifies the name of the table owner. You entered the table owner name when you created the persistent cache table. You cannot change this value.
Datastore name	Specifies the name of the datastore that you entered when you created the persistent cache. You cannot change this value.
Database type	Specifies the database type, which is Persistent Cache. You cannot change this value.
Column comparison	Specifies how the software maps the input columns to persistent cache table columns. • *compare_by_position*: Software maps source columns to target columns by position. Ignores the column names. • *compare_by_name*: Software maps source columns to target columns by column name.
Include duplicate keys	Specifies whether the software includes duplicate keys in the persistent cache. • Selected: Includes duplicate keys in the persistent cache. Selected is the default setting. • Not selected: Does not include duplicate keys in the persistent cache.
Key column	Specifies one or more columns to use as the key for the persistent cache. Select column names from the dropdown list. Change the order of the selected columns in the key by one of the following actions: • Right-click the column and select *Move Up* or *Move Down*. • Select the column and click the down or up arrow in the top right corner of the *Keys* tab. Remove a column by one of the following actions: • Right-click the column and select *Delete*. • Select the column and click the delete icon in the top right corner of the *Keys* tab.

Parent topic: Target [page 147]

Related Information

Target objects [page 148]
Target files [page 149]
Common Target Table editor options [page 152]
Types of target tables [page 167]
Target Data Transfer files and tables [page 200]
Target JSON files, messages, and templates [page 201]
Target XML files, messages, and templates [page 207]
Target flat files [page 213]
Embedded Data Flows

2.2.25.4 Common Target Table editor options

When you configure a target in a data flow, you complete options in the Target Table editor.

Access the Target Table editor by opening the target object in the data flow.

There are target table options that are common to all supported database types. Set common options in the following tabs of the Target Table editor:

- Target
- Options
- Load Triggers
- Preload Commands
- Post Load Commands

Target Table editor: Target tab [page 152]
Complete options in the Target tab of the target editor to set basic information about the target file.

Target Table editor: Options tab [page 155]
The Options tab of the target table editor contains options that control data loading, error handling, and update control.

Target Table editor: Load Triggers tab [page 165]
A load trigger operation is a template SQL statement that has placeholders for column and variable values.

Parent topic: Target [page 147]

Related Information

Target objects [page 148]
Target files [page 149]
Target persistent cache tables [page 150]
Types of target tables [page 167]
Target Data Transfer files and tables [page 200]
Target JSON files, messages, and templates [page 201]
Target XML files, messages, and templates [page 207]
Target flat files [page 213]

2.2.25.4.1 Target Table editor: Target tab

Complete options in the Target tab of the target editor to set basic information about the target file.

To open the target editor and target options, click the name of the target in the workspace or in the project area. The following table contains options in the Target tab of the object editor that are common to all supported database types.

Target tab option descriptions

Target option	Description
Make Port	Specifies whether to make this target an embedded data flow port. • Selected: Creates an embedded data flow port from the target. • Not selected: Does not create an embedded data flow port. Not selected is the default setting.

Target option	Description
Database type	Specifies the database type of the target object. Select the applicable database type and version from the dropdown list. After you select the database type, the software sets additional values on other tabs based on your selection. Selecting a value for *Database type* enables you to quickly set target option values in data flows. The software obtains the database information, and databases listed in the database type dropdown list, from the related datastore. When the datastore has more than one configuration, the dropdown list contains database types and versions from each configuration. If the database type and version are not available in the list, edit the datastore and add a new configuration with the missing database and version. The software sets the additional values on other tabs based on the following settings in the datastore: • If the datastore has one configuration for one database type, then the software populates the options in the Target tab with the information from the datastore. • If the datastore has more than one configuration and there are different database types and versions, then the software determines the initial values for the additional database types and versions based on your settings in the *Values for table targets* section. The software uses settings from the *Use values from* option in the *Create New Configuration* dialog box of the datastore editor. • If you also select the *Restore values if they already exist* option in the *Create New Configuration* dialog box of the datastore editor, the software looks for previously defined values that once existed for that database type or version. **i Note** It is possible for a data flow to contain target table editor values for a database type or version, even if you deleted the datastore configuration. 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 *Use values from* option.

Target option	Description
	Example For example, suppose that you create a configuration for Oracle 8i for the applicable datastore. When you edit the target table editor options, you change the *Rows Per Commit* default value of 1000 to 500. Later you add a new datastore configuration to the applicable datastore for a Microsoft SQL Server 2000 database. You set the *Use values from* option to Oracle 8i. The target table editor settings for SQL Server inherit the value 500 for *Rows per Commit* because 500 was the value set in the Oracle 8i configuration. The values you set for the options in the target table editor are specific to the instance and database type and version of that object in the data flow. Setting values for one target table in your data flow does not affect the other targets in the same data flow.

Parent topic: Common Target Table editor options [page 152]

Related Information

Target Table editor: Options tab [page 155]
Target Table editor: Load Triggers tab [page 165]
Datastore [page 54]

2.2.25.4.2 Target Table editor: Options tab

The Options tab of the target table editor contains options that control data loading, error handling, and update control.

The following table contains descriptions for settings in the Options tab, which has general and advanced options.

Option tab descriptions

Target table option	Description
General options:	

Target table option	Description
Rows per commit	Specifies the number of rows to process before the software sends a commit to the underlying database. Enter a positive integer. The default setting is *1000*. Because the software never splits Load triggers across transaction boundaries, when the load triggers crosses transaction boundaries, the software extends the size of the transaction to accommodate the entire trigger. **Example** For example, suppose that you set *Rows per commit* to 3 and specified an insert trigger. The insert trigger specifies that the software converts an incoming insert statement into 5 statements. The software automatically extends the *Rows per commit* to 5 to accommodate each insert trigger statement in a single transaction. This option is not available for targets in real time jobs.
Delete data from table before loading	Specifies whether the software deletes the contents of the table before loading new data to it. • Selected: For batch jobs, deletes the contents of the table before loading new data to it. For real-time jobs, deletes data after processing each message. **Note** For real-time jobs, do not select this option for development and testing phases. • Not selected: Truncates new data to the existing data in the table. Not selected is the default setting. The software sends a TRUNCATE statement to supporting databases. Or the software sends a DELETE statement to databases that do not support TRUNCATE.
Drop and re-create table	Drops the existing table and creates a new one with the same name before loading. This option displays only for template tables. Template tables are used in design or test environments.
Table type	Specifies whether the software creates tables organized by row or column. • *Column store*: Creates tables organized by column. The software does not support the following data types for this setting: blob, dbblob, and clob. • *Row store*: Creates tables organized by row.
Advanced options, General group:	

Target table option	Description
Column comparison	Specifies how the software maps the input columns to the output columns. • *Compare by position*: Maps the source columns to the target columns by position, and ignores the column names. • *Compare by name*: Maps the source columns to the target columns by name. The software issues validation errors when the data types of the columns do not match.
Number of loaders	Specifies the number of loaders the software uses. Enter a positive integer. The default is *1*. There are different types of loading: • Single loader loading: Loading with one loader. • Parallel loading: Loading with two or more loaders. When parallel loading, each loader receives the number of rows indicated in the *Rows per commit* option, and processes the rows in parallel with other loaders. **Example** For example, if *Rows per commit* = 1000 and *Number of Loaders* = 3: • First 1000 rows go to the first loader • Second 1000 rows go to the second loader • Third 1000 rows go to the third loader • The fourth 1000 rows go to the first loader

Target table option	Description
Identity upsert	Applicable when the target table contains an SAP HANA Identity column. Specifies whether the software generates the SQL statement to load the SAP HANA table that contains the identity column. Uses the input value for insert, update, and upsert operations. • *On*: When you map the Identity column, the software generates the SQL statement to load the SAP HANA table that contains the identity column, and uses the input value for insert, update, and upsert operations. • *Off*: Does not generate the SQL statement to load the SAP HANA table that contains the identity column. *Off* is the default setting. This option works with both the normal SAP HANA loader and the bulk loader. If you do not map the Identity column, Data Services issues an error.
Enable partitioning	Applicable when you partition the target table data. Specifies whether the software loads data using the number of partitions in the table as the maximum number of parallel instances. Select only one of the following loader options: • Number of Loaders • Enable Partitioning • Transactional Loading **i Note** If you set *Enable Partitioning* to Yes and *Include in transaction* to Yes, the *Include in transaction* setting overrides the *Enable Partitioning* option.

Target table error handling [page 159]
To determine how SAP Data Services processes errors related to your target table, set options in the *Error Handling* group in the *Options* tab of the target table object editor.

Update Control [page 160]
Descriptions of *Update Control* settings in the *Options* tab of the target table object editor.

Target table transaction control [page 163]
Descriptions of *Transaction Control* settings in the Options tab of the target table object editor.

Parent topic: Common Target Table editor options [page 152]

Related Information

Target Table editor: Target tab [page 152]
Target Table editor: Load Triggers tab [page 165]

2.2.25.4.2.1 Target table error handling

To determine how SAP Data Services processes errors related to your target table, set options in the *Error Handling* group in the *Options* tab of the target table object editor.

Error Handling

Error handling option	Description
Use overflow file	For recovery purposes. If the software can't load a row, it writes the row to a file. When you set this option to *Yes*, the *File Name* and *File Format* options are enabled. The default setting is *No*.
File name *File format*	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). The *File name* and *File format* options are available only when you set *Use overflow file* to *Yes*. Optional. For convenience, use a variable for *File Name*.

Related Information

Target Table editor: Options tab [page 155]

2.2.25.4.2.2 Update Control

Descriptions of *Update Control* settings in the *Options* tab of the target table object editor.

Update control option descriptions

Option	Description
Use input keys	If the target table does not contain a primary key, this option enables the software to use the primary keys from the input. The default setting is *No*. If the target is a Microsoft SQL Server database table, and the identity column is mapped as the primary key, this option must = *No*.
Update key columns	This option is set to *No* by default. If you select *Yes* for this option, the software updates key column values when it loads data to the target.

Option	Description
Auto correct load **i Note** The optimizer automatically uses MERGE statements to improve auto correct load performance.	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*. **i Note** This option is not available for targets in real-time jobs or target tables that contain LONG column(s). When you select *Yes* for this option, the software reads a row from the source, then checks if the row exists in the target table with the same values in the primary key. If *Use input keys* is set to *Yes*, the software uses the primary key of the source table. Otherwise, the software uses the primary key of the target table; if the target table has no primary key, the software considers the primary key to be all the columns in the target. If a matching row does not exist, a new row is inserted, regardless of other options. If a matching row exists, the row is updated depending on the values of *Ignore columns with value*, and *Ignore columns with null*: • When the column data from the source matches the value in *Ignore columns with value*, the corresponding column in the target table is not updated. The value may be spaces. Otherwise, the corresponding column in the target is updated with the source data. • When the *Ignore columns with null* option is set to *Yes* and the column data from the source is NULL, then the corresponding column in the target table is not updated. Otherwise, the corresponding target column is updated as NULL since the source column is NULL. For supported databases, when the *Allow merge or upsert* option is enabled, the software can optimize data flows such that the database completes the auto correct load operation. When all other operations in the data flow can be pushed down to the source database, the auto-correct loading operation is also pushed down. The generated SQL implements the *Ignore columns with value* value when completed in the target editor, and the *Ignore columns with null* Yes/No setting.

Option	Description
Allow merge or upsert (DB2, Teradata, and Microsoft SQL Server 2008 and higher only) **i Note** The optimizer automatically uses MERGE statement for HP Vertica when user chooses Yes for *Auto correct load*.	Specifies whether the optimizer may use a MERGE statement to improve the performance of auto correct load functionality. *Yes:* Allows the optimizer to consider using a MERGE statement during an auto correct load operation. If the Optimizer does not use a MERGE statement, it uses a T-SQL block to identify, insert, and update rows. *No:* The optimizer does not use a MERGE statement to improve auto correct load performance. The default is Yes. **i Note** When the data flow contains a Query transform with an Order by clause, the optimizer always uses a T-SQL block to identify, insert, and update rows.
Ignore columns with value	Enter a value that might appear in a source column and that you do not want updated in the target table. The value must be a string, it can include spaces, but the string cannot be in single or double quotations. When this value appears in the source column, the corresponding target column is not updated during auto correct loading.
Ignore columns with null	Select *Yes* if you do not want NULL source columns updated in the target table during auto correct loading. This option is only available when you select *Yes* for the *Auto correct load* option.

Related Information

Target Table editor: Options tab [page 155]

2.2.25.4.2.3 Target table transaction control

Descriptions of *Transaction Control* settings in the Options tab of the target table object editor.

Transaction Control options

Transaction control option	Description
Include in transaction	Indicates that this target is included in the transaction processed by a batch or real-time job. This option allows you to commit data to multiple tables as part of the same transaction. If loading fails for any one of the tables, no data is committed to any of the tables. The tables must be from the same datastore. 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. If you choose to enable transactional loading, the following options aren't available: • Rows per commit • Use overflow file, and overflow file specification • Number of loaders • Enable partitioning • Bulk loader options • Preload commands • Post load commands • Delete data from table before loading • Identity insert The software doesn't push down a complete operation to the database if transactional loading is enabled.
Transaction order	Transaction order indicates where this table falls in the loading order of the tables being loaded. By default, there's no ordering. All loaders have a transaction order of zero. If you specify orders among the tables, the loading operations are applied according to the order. Tables with the same transaction order are loaded together. Tables with a transaction order of zero are loaded at the discretion of the data flow process. In Data flow view, the specified transaction order number appears nearer to the corresponding target.

Transaction control option	Description
Commit at end of INSERT...SELECT	Specifies whether to commit a single transaction log statement for each load or to limit the transaction size. • *Yes*: Commits a single transaction log statement per load. *Yes* is the default setting. • *No*: Limits transaction size to the value in *Rows per commit* in the *Options* tab. Creates smaller transaction size than the single commit per job default behavior. **i Note** Setting *Yes* for *Commit at end of INSERT...SELECT* decreases performance. If your transaction log size is too small for a single transaction of this type, consider selecting *No* for this option under the following circumstances: • The job has a source and target that use the same datastore. • The job has an Oracle target table. • The software is optimizing and executing the job by pushing down the read operation to an Oracle target table host. **i Note** The software generates an INSERT...SELECT SQL statement and a transaction for an INSERT...SELECT by default at the end of the job. • The job failed with an Oracle transaction log full error. If you select *No* for this option, set a value for the *Rows per commit* option. Before you set this option, view the SQL that Data Services generates to see if an INSERT ...SELECT statement is in use: 1. In Designer, open the data flow in the workspace. 2. Select *Validate* > *Display Optimized SQL*. If an INSERT...SELECT statement is in use, set the *Commit at end of INSERT...SELECT* to *No* and enter a value for *Rows per commit*.

Transaction control option	Description
Use NVARCHAR for VARCHAR columns in supported databases	Specifies whether Data Services creates NVARCHAR columns in the template table for all VARCHAR columns in the input schema. The data type displays as VARCHAR in the Designer, and when supported by the DBMS, displays as NVARCHAR in the database table. The following database management systems don't support the NVARCHAR data type: • DB2 (non-UTF-8) • Oracle 8.x • Informix • SAP ASE • SAP Sybase IQ For these systems, the software creates columns with VARCHAR data types and increases the column size using a codepage conversion factor based on the client code page defined in the datastore. ⚠ Caution When the system transcodes from one national language to another national language, it can lose data. The system won't lose data when it transcodes from a national language to Unicode. ⚠ Caution Data Services truncates data when the column size of the source exceeds the maximum size allowed by the target DBMS.

Related Information

Target Table editor: Options tab [page 155]

2.2.25.4.3 Target Table editor: Load Triggers tab

A load trigger operation is a template SQL statement that has placeholders for column and variable values.

A load trigger specifies a SQL command that is performed by the database on an INSERT, UPDATE, or DELETE operation. For more information about load triggers and completing the SQL statement, see the *Designer Guide*.

> **i Note**
>
> Data Services does not support load trigger options for Microsoft APS or Azure DW.

Set the load trigger operation to occur before, after, or instead of normal operations.

Load Triggers option descriptions

Load trigger option	Description
On operation	Specifies the type of SQL statement for the load trigger. Select an operation from the dropdown list. • INSERT: Specifies an INSERT operation. • UPDATE: Specifies an UPDATE operation. • DELETE: Specifies a DELETE operation.
SQL command text box	Specifies the load trigger. Enter load triggers manually or drag column names from the *Schema In* pane. Enclose column names in curly braces or square brackets. For example, `{SalesOffice}` or `[SalesOffice]`. The software automatically includes quotes around field name in curly braces, if applicable. The software does not include quotes around field names in square brackets. 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.

Parent topic: Common Target Table editor options [page 152]

Related Information

Target Table editor: Target tab [page 152]
Target Table editor: Options tab [page 155]
Load Triggers tab in target editor

2.2.25.5 Types of target tables

SAP Data Services supports several target table types that support the various types of data transactions, including transactions with database management, Big Data, and cloud systems.

If Data Services can write to the application or database containing the table, you can add the table to a data flow diagram as a target.

If the schema defined in the table doesn't match the schema that is input to the target, Data Services issues validation errors to identify the mismatch.

When loading DB2, ODBC, or Oracle tables, Data Services parameterizes the SQL, which results in quicker load times. To parameterize SQL, Data Services has to be able to generate, parse, and compile the statement. For example, Data Services can't parameterize SQL when the target table is set for transactional loading or triggers.

The following table describes the types of target tables supported in Data Services, and provides a source for more information.

Target table type	Description	More information
Amazon Redshift	The software supports the following features when you use an Amazon Redshift table as a source: • All Redshift data types • Optimized SQL • Basic push-down functions	*Data Services Supplement for Big Data*
DB2	Use tables from your DB2 RDBMS.	*Reference Guide*: DB2 target table options [page 169]
Google BigQuery	Create a Google BigQuery datastore to load data from Data Services into a prepared table in your Google BigQuery dataset for Google BigQuery analysis.	*Supplement for Google BigQuery*
HP Vertica	Create an HP Vertica datastore to access HP Vertica tables.	*Data Services Supplement for Big Data*
Informix	Use tables from your Informix RDBMS.	*Reference Guide*: Informix target table options [page 173]
Microsoft SQL Server	Use tables from your Microsoft SQL Server RDBMS.	*Reference Guide*: Microsoft SQL Server target table options [page 176]
Netezza	Create a Netezza datastore to access Netezza tables.	*Reference Guide*: Netezza target table options [page 180]
ODBC	Use ODBC to access tables from your RDBMS.	*Reference Guide*: ODBC target table options [page 182]

Target table type	Description	More information
Oracle	Use tables from your Oracle RDBMS.	*Reference Guide*: Oracle target table options [page 183]
PostgreSQL	Use PostgreSQL tables as targets in a data flow. To access tables, create a PostgreSQL datastore.	*Data Services Supplement for Big Data*
SAP HANA	Create an SAP HANA datastore to access SAP HANA tables.	*Data Services Supplement for Big Data*
SAP ASE	Use tables from your SAP ASE RDBMS.	*Reference Guide*: SAP ASE target table options [page 187]
SAP IQ	Use tables from your SAP IQ RDBMS.	*Reference Guide*: SAP Sybase IQ target table options [page 188]
SAP Vora	Create an SAP Vora datastore to access SAP Vora tables.	*Data Services Supplement for Big Data*
Teradata	Use tables from your SAP Vora RDBMS.	Reference Guide: Teradata target table options [page 191]

DB2 target table options [page 169]
When you use DB2 tables as a target in a data flow, complete the target editor options specific to DB2 as well as the common table options.

Informix target table options [page 173]
When you use Informix tables as a target in a data flow, complete the Informix-specific options in addition to the common options.

Microsoft SQL Server target table options [page 176]
When you use Microsoft SQL Server target tables, set specific options in addition to the common table options.

Netezza target table options [page 180]
When you use a Netezza target table, complete the Netezza-specific options in the target editor.

ODBC target table options [page 182]
When you use ODBC target tables, complete the ODBC-related options in addition to the common options in the target editor.

Oracle target table options [page 183]
When you use Oracle as a target table, complete the Oracle-specific options in addition to the common options in the table editor.

SAP ASE target table options [page 187]
When you use SAP ASE as a target table, complete the SAP ASE-specific options in addition to the common options in the table editor.

SAP Sybase IQ target table options [page 188]
When you use SAP IQ for the target table, complete the SAP IQ-specific options in addition to the common options in the target editor.

Teradata target table options [page 191]
When you use Teradata as a target in the data flow, complete the Teradata-specific options in addition to the common options in the target editor.

Parent topic: Target [page 147]

Related Information

Target objects [page 148]
Target files [page 149]
Target persistent cache tables [page 150]
Common Target Table editor options [page 152]
Target Data Transfer files and tables [page 200]
Target JSON files, messages, and templates [page 201]
Target XML files, messages, and templates [page 207]
Target flat files [page 213]
Amazon Redshift target table options
Use Google BigQuery as a target
HP Vertica target table configuration
SAP HANA target table options
SAP Vora target table options

2.2.25.5.1 DB2 target table options

When you use DB2 tables as a target in a data flow, complete the target editor options specific to DB2 as well as the common table options.

Target tab DB2 option description

Option	Description
Make port	Specifies whether to make the table an embedded data flow port. • Selected: Makes the table an embedded data flow port. • Not selected: Does not make the table an embedded data flow port.

Options tab option descriptions

Option	Description
Allow Merge or upsert	Specifies whether the Optimizer uses a MERGE statement to improve the performance of auto correct load functionality. • *Yes*: Optimizer uses a MERGE statement when applicable during an auto correct load operation. *Yes* is the default setting. > **i Note** > If the Optimizer does not use a MERGE statement, it uses a stored procedure to identify, insert, and update rows. • *No*: Optimizer does not use a MERGE statement to improve auto correct load performance.
Table type	Specifies whether to create tables organized by row or column. • *Column Store*: Creates tables organized by column. > **i Note** > Column Store does not support the following data types: blob, dbblob, and clob. > **i Note** > The software supports column organized tables only on Linux and AIX: > • Linux: x86-x64, Intel, and AMD processors > • AIX: POWER processors > There are additional steps required to set up DB2 to use the *Column Store* option. • *Row store*: Create tables organized by row.

Bulk loader tab DB2 option descriptions

Option	Description
Bulk load	Specifies the bulk load method to use. • *CLI load*: Uses 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. Applicable for DB2 version 8.x or later. • *Import*: Uses 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. • *No*: Does not use bulk loading.

Option	Description
Generate files only	Specifies to generate a data file and a control file for the *Import* method of bulk loading. • Selected: Generates a data file and a control file. Use these files for targets of DB2 bulk loading. Useful when DB2 server is on a system that runs a different operating system than the Job Server. The software writes the data file and the control file in the bulk loader directory that you specify in a datastore definition. If you have not specified a bulk loader directory, the software writes the files in the `<DS_COMMON_DIR>\log\bulkloader` directory. To load the data, manually copy the files to the remote system and start the bulk load execution. When you select this checkbox, only the *Text delimiter* and *Column delimiter* options are available. • Not selected: Does not generate a data file and a control file. Available only when you set *Bulk load* to *Import*.
Clean up bulk loader directory after load	Specifies to delete the files in the bulk loader directory after loading completes. • Selected: Deletes all files in the bulk loader directory after the load completes successfully. If you have not specified a bulk loader directory in the *Connections* tab in the datastore definition, the software writes the files in the `<DS_COMMON_DIR>\log\bulkloader` directory. • Not selected: Does not delete files in the bulk loader directory after the load completes successfully. When this option is selected, the software deletes the following files after each bulk load unless the software throws an error: • Message file, `.log` that DB2 creates for the *Import*, *Load*, or *CLI load* • Control file, `.ctl` that the software generates only when *Bulk load* is set to *Import* • Data file, `.dat` that the software generates only when *Bulk load* is set to *Import*
Mode	Specifies the mode for loading data in the target table. Available modes depend on the bulk load method that you select. Available modes when *Bulk load* is set to *Import*: • *Insert*: Adds new records to the table. Use when loading data into an empty table or when appending data to an existing table that contains data that you want to maintain. • *Insert-update*: If a record with matching primary keys exists in the table, updates that record; otherwise, adds new record to the table. This method requires that the target table has primary keys. • *Replace*: Deletes all existing records in the table, then adds new records. • *Truncate*: Deletes all existing records in the table, then adds new records. Available modes when *Bulk load* is set to *CLI load*: • *Insert*: Appends the new records into the target table. • *Replace*: Deletes the existing records, and then inserts the loaded data.

Option	Description
Rows per commit	Specifies the number of rows that the software loads before a commit takes place. Enter a positive integer. If you leave this option blank, the load utility uses the default value at runtime. The default is 1000. This option is available only when *Bulk load* is set to *Import*.
Warning row count	Specifies the number of warnings allowed for each load operation. Enter a positive integer. This option is available only when *Bulk load* is set to *CLI load*.
Text delimiter	Specifies the character to use as the character string delimiter. Specify a single character string delimiter. The specified character can be any printable or nonprintable ASCII character, escaped with a double slash "\\". This option is available only when *Bulk load* is set to *Import*.
Column delimiter	Specifies a single character as the column delimiter. Specify a single character column delimiter. The default is a comma (,). This option is available only when *Bulk load* is set to *Import*.
Maximum bind array	Specify the maximum number of rows the software extracts before it sends the data to the DB2 table or view target. Enter a positive integer. The default is 1000. This option is available only when *Bulk load* is set to *CLI load*.
Exception table name	Specifies the table to use for rows that violate a table constraint. The software deletes rows that violate constraints from the target table and inserts them into the file that you specify for the exception table. This option is available only when *Bulk load* is set to *CLI load*.
Recoverable	Specifies to support data recovery through the DB2 roll-forward recovery feature. • Selected: DB2 writes a backup copy of the loaded data. Use DB2 roll-forward recovery after failure. Also specify the directory for writing the backup file in the *Copy target directory* option. Select this option only if your DB2 target database is roll-forward enabled. • Not selected: The software disables recover from failure using DB2 roll forward. This option is available only when *Bulk load* is set to *CLI load*.
Copy target directory	Specifies the directory for the copy files. Supports only local directories. Applicable when you select *Recoverable*. This option is available only when *Bulk load* is set to *CLI load*.

Parent topic: Types of target tables [page 167]

Related Information

Informix target table options [page 173]
Microsoft SQL Server target table options [page 176]
Netezza target table options [page 180]
ODBC target table options [page 182]
Oracle target table options [page 183]
SAP ASE target table options [page 187]
SAP Sybase IQ target table options [page 188]
Teradata target table options [page 191]
Setting column based target options for IBM DB2 10.5

2.2.25.5.2 Informix target table options

When you use Informix tables as a target in a data flow, complete the Informix-specific options in addition to the common options.

> **i Note**
>
> Commit at the end of INSERT..SELECT option is not applicable.

Options tab Informix option description

Option	Description
Drop and re-create table	Specifies to drop the existing table and create a new table using the same name as before loading. • Selected: Creates a new table with the same name as the table before loading, and replaces the content with the content of the generated output. • Not selected: Does not create a new table and loads data into existing table. **i Note** If the existing table schema does not match the existing schema, the software generates runtime errors. Use the runtime errors to remap the schema.

Bulk Loader Informix option descriptions

Option	Description
Bulk load	Specifies to use the bulk load features for loading data to this target. • Selected: Enables bulk loading for this target and this data flow. • Not selected: Does not use bulk loading for this target in this data flow. Not selected is the default setting.
Mode	Specifies the mode to use for loading data in the target table. • *Append*: Adds new records to the table. *Append* is the default setting. • *Replace*: Deletes all existing records in the table and then adds new records.

Option	Description
Generate files only	Specifies to generate a data file and a control file for bulk loading. • Selected: Generates a data file and a control file. Use these files for targets of Informix bulk loading. Useful when Informix server is on a system that runs a different operating system than the Job Server. The software writes the data file and the control file in the bulk loader directory that you specify in a datastore definition. If you have not specified a bulk loader directory, the software writes the files in the `<DS_COMMON_DIR>\log\bulkloader` directory. To load the data, manually copy the files to the remote system and start the bulk load execution. The files names are `<tablename>.ctl` and `<tablename>.dat`, where `<tablename>` is the name of the target table. • Not selected: Does not generate a data file and a control file.
Lock table	Specifies to lock the table for the duration of the load. • Selected: Locks the target table until loading finishes. • Not selected: Does not lock the target table.
Clean up bulk loader directory after load	Specifies whether the software cleans up the bulk loader directory after data loads to the target. • Selected: Deletes all bulk load files from the directory after the load is completed successfully. The software deletes the following files: Control file, Log file, Bad file. • Not selected: Does not delete files from the bulk load directory.
Bulk loader server name	Specifies the name of the Informix database server.
Bulk loader database name	Specifies the name of the bulk loader database, such as the name of the target warehouse database.
Rows per commit	Specifies the number of rows that the software loads before a commit takes place. Enter a positive integer. If you leave this option blank, the load utility uses the default value at runtime. The default is 1000.
Field delimiter	Specifies the character to use as the field delimiter. Specify a single character field delimiter. The specified character can be any printable or nonprintable ASCII character, escaped with a double slash "\\".
Maximum rejects	Specifies the maximum number of acceptable warnings before the software stops bulk loading. Enter a positive integer. If you enter 0 or leave the option blank, the software continues bulk loading regardless of the number of warnings it issues. Default value is 10. Set this parameter when you expect no warnings, but want to verify that the software uses the correct file and table.
Begin/end column character	Specifies the character the software uses that delimits the beginning and ending of the column.

Parent topic: Types of target tables [page 167]

Related Information

DB2 target table options [page 169]
Microsoft SQL Server target table options [page 176]
Netezza target table options [page 180]
ODBC target table options [page 182]
Oracle target table options [page 183]
SAP ASE target table options [page 187]
SAP Sybase IQ target table options [page 188]
Teradata target table options [page 191]

2.2.25.5.3 Microsoft SQL Server target table options

When you use Microsoft SQL Server target tables, set specific options in addition to the common table options.

Options tab Microsoft SQL server option descriptions

Option	Description
Identity Insert	Specifies whether the software or SQL server generates identity column values for the target table. • *On*: Software generates identity column values in the target table. **i Note** Map the identity column in the output schema. • *Off*: SQL Server generates the identity column values are generated by SQL Server. • *Not applicable*: Disables this option. **i Note** Make sure the setting that you make for this option is compatible with other related settings to avoid validation errors: • If *Identity Insert* is set to *On*, map the identity column in the target or else the software issues a validation error. • If the identity column is a key column, set the option *Update Key Column* in the *Options* tab to *No*. If you set it to *Yes*, the software issues a validation error. • Set the *Identity Insert* option to *Off* when you use the transaction loader or the software issues a validation error. **i Note** If you set *Include in transaction* under *Transaction Control* to *Yes*, the software automatically sets the *Identity Insert* option to *Not Applicable*. **i Note** The *Identity Insert* option is not supported for Microsoft APS or Azure DW.

Option	Description
Allow merge or upsert	Applicable only for Microsoft SQL Server 2008 and higher. Specifies whether the Optimizer uses a MERGE statement to improve the performance of auto correct load functionality. • *Yes*: Allows the Optimizer to consider using a MERGE statement during an auto correct load operation. If the Optimizer does not use a MERGE statement, it uses a T-SQL block to identify, insert, and update rows. > **i Note** > When the data flow contains a Query transform with an Order by clause, the Optimizer always uses a T-SQL block to identify, insert, and update rows. • *No*: The Optimizer does not use a MERGE statement to improve auto correct load performance. > **i Note** > The *Allow merge or upsert* option is not supported for Microsoft APS or Azure DW.

Bulk loader Microsoft SQL Server option descriptions

Option	Description
Bulk load	Specifies whether the software uses SQL Server bulk loading to the target table. • Selected: Enables bulk loading. • Not selected: Does not enable bulk loading. Not selected is the default setting. > **i Note** > The *Bulk load* option is not supported for Microsoft APS. > **i Note** > In Azure DW, if you insert a NULL in a DW decimal column using bulk loading, an error occurs.
Rows per commit	Specifies the number of rows to load before the software commits the transaction. Enter a positive integer. The default setting is 1000 rows.

Option	Description
Mode	Specifies the mode for loading data to the target table. • *Append*: Adds new records to the table. • *Truncate*: Deletes all existing records in the table, and then adds new records.
Maximum rejects	Specifies the maximum number of errors encountered in the data before the bulk load operation is stopped. When the software exceeds the value for *Maximum rejects*, the bulk loader stops. Set this parameter when you expect no errors but want to verify that the software is loading the correct file and table. • **Blank**: No errors are allowed. If an error is encountered, Data Services stops processing and rolls back all the previous transactions. • **0**: Data Services continues bulk loading regardless of the number of errors it encounters. • **Positive number (greater than 1)**: Data Services stops processing after `<n>` errors. A rejected row contains data that does not match the expected format or information specified for the table being loaded. **i Note** In Azure DW, if the number of rejected rows is lower than the number of maximum rejects, the job completes successfully without an error message and also without a listing of the rows that failed during the load.
Network packet size	Specifies the network packet size in KB. Enter an integer. The default setting 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.
Options specific to Microsoft Azure DW:	

Option	Description
File Location	Specifies the file location object for the target. Select the file location object from the dropdown list. The dropdown list includes Azure blob storage and Data Lake Store file locations that are set up as global variables or system parameters. Select the file location you want to use from the dropdown menu. **i Note** Data Services supports Azure Data Lake Gen2 connections only with a shared key authentication file location.
File Options	Enter the delimiters you want to use for both columns and text. The default value for the Number of loaders is 4. Increasing or decreasing this number may affect the bulk load performance. The software saves the files in the bulkload directory. If you check *Clean up bulk loader directory after load*, the software deletes the data file or files from the local bulkload directly and Azure blob storage or Data Lake Store after the job completes successfully.

Parent topic: Types of target tables [page 167]

Related Information

DB2 target table options [page 169]
Informix target table options [page 173]
Netezza target table options [page 180]
ODBC target table options [page 182]
Oracle target table options [page 183]
SAP ASE target table options [page 187]
SAP Sybase IQ target table options [page 188]
Teradata target table options [page 191]
Update Control [page 160]
Target table transaction control [page 163]

2.2.25.5.4 Netezza target table options

When you use a Netezza target table, complete the Netezza-specific options in the target editor.

> **i Note**
>
> The options under the Transaction Control category are not available for Netezza. Unavailable options include: *Include in Transaction*, *Transaction ID*, *Commit at the end of INSERT...SELECT*.

Bulk Loader Netezza option descriptions

Option	Description
Bulk load	Specifies the method the software uses for bulk loading to the target table. • *File*: Writes all data to a file before loading through the external table to the staging table. For files that are smaller than 4 GB in size, select this option for faster performance. • *Named pipe*: Streams data as it is written to the named pipe through the external table to the staging table. For files that are larger than 4 GB in size, select this option for faster performance. • *None*: Does not use bulk loading. *None* is the default setting.
Mode	Specifies the mode the software uses for loading data to the target table. • *Append*: Adds new records to table. *Append* is the default setting. • *Truncate*: Deletes all existing records in table then adds new records.
Update method	Specifies how the software applies UPDATE SQL. • *Delete-insert*: Issues a DELETE to the target table for data that matches the old data in the staging table, and then issues an INSERT with the new data. Select this option for faster performance. *Delete-insert* is the default setting for this option. • *Update*: Issues an UPDATE to the target table. **i Note** If the update rows contain data in only some of the columns, do not use this option. SAP Data Services replaces missing data with NULLs.
Maximum rejects	Specifies the maximum number of warnings issues before the software terminates the job. Enter a positive integer. The default is 10.
Commit size	Specifies the maximum number of rows loaded to the staging and target tables before the software saves the data. Enter an integer. If you enter 0, the software loads and saves all rows. 0 is the default setting. The value for this option overrides the *Rows per commit* setting.
Text delimiter	Specifies the character to use to delimit character or varchar data. Valid values are single quotes, double quotes, or blank. Blank is the default setting. If your data contains single or double quotes, use an escape character with the delimiter.

Option	Description
Field delimiter	Specifies the character to use to delimit columns in a row. Enter any printable character except single quotes. The default delimiter is a pipe (|). If your data has the field delimiter character in the varchar or char data, use a text delimiter.
Generate files only	Specifies that the software generate data into a file on the Job Server. • Selected: Generates data into a file on the Job Server. • Not selected: Does not generate data into a file on the Job Server. Not selected is the default setting. This option is useful to manually load data using the SQL statements generated in the .sql script.
Escape character	Specifies to use the back slash (\) as an escape character. • Selected: Select if you expect the text delimiter character or the row delimiter character (ASCII character or line feed) to be present in any varchar or char data in your data. The escape character signals to the software to escape the text or row delimiter before writing to a file or named pipe. Also select when you expect a NULL string as part of the varchar data. The software uses the NULL string as a null indicator to identify null values during bulk loading. ℹ Note The null indicator string is not case sensitive so if you have varchar data as NULL (case insensitive) select this option. ℹ Note The software also escapes back slashes in the data with a back slash (\\). • Not selected: Does not use the back slash character as an escape character. No selected is the default setting. Enabling this optin can degrade job execution performance.
Clean up bulk loader directory after load	Specifies whether the software deletes items in the bulk loader directory after a successful load. • Selected: Deletes all bulk loader files after the load completes successfully. Includes file types such as .dat, .sql, .nzlog, .nzbad, and so on. Selected is the default setting. ℹ Note If an error occurs during the bulk load, the Netezza server creates a nzbad file in the Database server working directory defined in the datastore editor. ℹ Note If you enable 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. • Not selected: Does not delete all bulk loader files after the load completes successfully.

Parent topic: Types of target tables [page 167]

Related Information

DB2 target table options [page 169]
Informix target table options [page 173]
Microsoft SQL Server target table options [page 176]
ODBC target table options [page 182]
Oracle target table options [page 183]
SAP ASE target table options [page 187]
SAP Sybase IQ target table options [page 188]
Teradata target table options [page 191]

2.2.25.5.5 ODBC target table options

When you use ODBC target tables, complete the ODBC-related options in addition to the common options in the target editor.

> **i Note**
>
> Bulk loader options are visible only if you are loading to a Netezza target.

Bulk loader ODBC option description

Option	Description
Rows per commit	Specifies the number of rows the software processes before it commits the rows. Enter a positive integer. The default setting is 1000. If you do not add a value and delete the default setting, the load utility uses the default value of 1000 at runtime.

Parent topic: Types of target tables [page 167]

Related Information

DB2 target table options [page 169]
Informix target table options [page 173]
Microsoft SQL Server target table options [page 176]
Netezza target table options [page 180]
Oracle target table options [page 183]
SAP ASE target table options [page 187]
SAP Sybase IQ target table options [page 188]
Teradata target table options [page 191]

2.2.25.5.6 Oracle target table options

When you use Oracle as a target table, complete the Oracle-specific options in addition to the common options in the table editor.

Options tab Oracle option descriptions

Option	Description
Allow merge	Specifies whether the Optimizer can consider using a MERGE statement to improve the performance of auto correct load functionality. • *Yes*: Allows the Optimizer to consider using a MERGE statement during an auto correct load operation. If the Optimizer does not use a MERGE statement, it uses a PL/SQL block to identify, insert, and update rows. *Yes* is the default setting. • *No*: The Optimizer does not use a MERGE statement to improve auto correct load performance.

Bulk loader Oracle option descriptions

Option	Description
Bulk load	Specifies the bulk loading method. • *API*: Uses an Oracle direct-path load API to load table data directly to database files. Ensure that the target database is Oracle version 8.1 or later. • *File*: Uses a staging file and the Oracle SQL*Loader to load table data. Ensure that the version of the Oracle loader specified in the datastore is the same version as the Oracle loader specified in the target editor. If you also want to use the direct-path load method, manually select it from the File Options section. Otherwise, the loader performs a conventional load. Conventional loads are slower than direct-path loads because data is loaded to tables rather than directly to database files associated with tables • *None*: Allows you to use normal load functionality.
Mode	Specifies the mode for loading data in the target table. Available modes when Bulk load is set to File: • *Append*: Adds new records that contain the loaded data. • *Insert*: Adds new records that contain the loaded data. Requires that the table is empty before loading. If the table is not empty, SQL loader terminates with an error. • *Replace*: Deletes all existing records in the table, and then adds new records containing the loaded data. • *Truncate*: Deletes all existing records in the table, and then adds new records containing the loaded data. This mode does not execute any delete triggers. Available modes when Bulk load is set to API: • *Append*: Adds new records containing the loaded data. • *Truncate*: Deletes all existing records in the table, and then adds new records containing the loaded data.

Option	Description
Rows per commit	Specifies the number of rows that the software loads before it commits the data to the underlying database. Enter a positive integer. The default setting is 1000. If you leave the option blank, the software uses the default setting. Bulk loading is not available for targets in real-time jobs.
Maximum rejects	Specifies the maximum number of errors before the software terminates the job. Enter a positive integer. The default setting is 10. If you delete the value from this option, the software uses the default number of errors.
Recoverable	Specifies whether the software logs direct-path information about the loaded data in the Oracle redo log. • Selected: Logs direct-path information about the loaded data in the Oracle redo log. • Not Selected: Does not log direct-path information about the loaded data in the Oracle redo log. **i Note** The software automatically specifies all conventional loads as recoverable.
Use local index to rebuild primary key	Specifies whether the software uses the partitioned index to rebuild the primary key. Applicable only when the table has local partitioned indexes on primary key. • Selected: Uses the partitioned index to rebuild the primary key when the table has local partitioned indexes on primary key. Does not use any other indexes that are available on the primary key. • Not Selected: Does not use the partitioned index to rebuild the primary key when the table has local partitioned indexes on primary key. Applicable only when you select *API* for the *Bulk Load* option.
*SQL *Loader version*	Specifies the SQL *Loader version that the software uses to load data into the table. **i Note** The SQL *Loader version and the Oracle database version specified in the datastore must match. Applicable only when you select *File* for the *Bulk Load* option.
Text delimiter	Specifies the character to use as the char and varchar string delimiter. Accept the default, or enter a character that is not in any of the data columns. The default character is double quotation marks. Applicable only when you select *File* for the *Bulk Load* option. Applicable only when you select *File* for the *Bulk Load* option.

Option	Description
Field delimiter	Specifies the character to use as the column delimiter. Accept the default or enter a character that is not used in any of the data columns. The default delimiter is a comma (,). To specify a non printable character, enter the ASCII equivalent, such as **/ASCII_number**. Applicable only when you select *File* for the *Bulk Load* option.
Maximum bind array	Specifies the maximum bind array to use. Enter a positive integer. If you do not enter a value, the software uses the default Oracle Bulk Loader value. Ensure that the bind array is large enough to contain a single row. For good performance, set the bind array large enough to hold 100 rows. Applicable only when you select *File* for the *Bulk Load* option.

Use the control file	Specifies to use a specific bulk load control file to load data from the associated data file instead of loading data from the source specified in the data flow.
	
	• Selected: Uses a specific bulk load control file and data file for loading data. The software directs Oracle to load data from the data file associated with the named control file.
	• Not selected: Uses the source specified in the data flow from which to load data.
	
	If you select this option, enter the path and file name of the control file, excluding the `.ctl` extension. Ensure that the control file and the associated data file are in the same directory. For convenience, you can also use a variable.
	
	If you do not specify a complete path, the software searches for the file in one of the following locations:
	
	• The path you specify as the bulk loader directory in the datastore definition.
	• The default file path:
	  • Windows: `%DS_COMMON_DIR%\log\bulkloader`
	  • Linux: `$LINK_DIR/log/bulkloader`
	
	If you select this option, also select the following options or specify values for the following options:
	
	• *Direct path*
	• *Clean up bulk loader directory after load*
	• *Rows per commit*
	• *Maximum rejects*
	• *Maximum bind array*
	
	Applicable only when you select *File* for the *Bulk Load* option.

Option	Description
Generate files only	Specifies to generate a data file and a control file as the target instead of the target specified in the data flow. Rather than loading data into the target in the data flow, the software generates a control file and a data file that you can later load using Oracle bulk loading. • Selected: Generates a data file and a control file. Use these files for targets of Oracle bulk loading. Useful when Oracle server is on a system that runs a different operating system than the Job Server. The software writes the data file and the control file in the bulk loader directory that you specify in the datastore definition. If you have not specified a bulk loader directory, the software writes the files in the `<DS_COMMON_DIR>\log\bulkloader` directory. To load this data, manually copy the files to the remote system and start the bulk load execution. The software names the files `tablename.ctl` and `tablename.dat`. • Not Selected: Does not generate a data file and a control file. Applicable only when you select *File* for the *Bulk Load* option.
Direct path	Specifies to use a direct path load. • Selected: Uses a direct path load. Ensure that the version of SQL *Loader in the Job Server that executes the job is the same version as the target database version. • Not Selected: Does not use a direct-path load method. Applicable only when you select *File* for the *Bulk Load* option.
Clean up bulk loader directory after load	Specifies to delete the files in the bulk loader directory after loading completes. • Selected: Deletes all files in the bulk loader directory after the load completes successfully. If you have not specified a bulk loader directory in the Connections tab in the datastore definition, the software writes the files in the `<DS_COMMON_DIR>\log\bulkloader` directory. If an error occurs during the bulk load, the software creates a .bad file and does not delete any files. Errors can occur when the software does not create a log file, or the log file contains the characters ORA- or SQL*Loader. • Not selected: Does not delete files in the bulk loader directory after the load completes successfully. Applicable only when you select *File* for the *Bulk Load* option.
Trailing nullcols	Specifies whether the software treats as null columns the columns that are not represented in the loaded data. Use when a data record is not complete but the existing data needs to be loaded. • Selected: Treats as null columns the columns that are not represented in the loaded data. • Not Selected: Does not treat as null columns the columns that are not represented in the loaded data. The software issues an error when the option is not selected and there are columns that are not represented in the loaded data.

Parent topic: Types of target tables [page 167]

Related Information

DB2 target table options [page 169]
Informix target table options [page 173]
Microsoft SQL Server target table options [page 176]
Netezza target table options [page 180]
ODBC target table options [page 182]
SAP ASE target table options [page 187]
SAP Sybase IQ target table options [page 188]
Teradata target table options [page 191]

2.2.25.5.7 SAP ASE target table options

When you use SAP ASE as a target table, complete the SAP ASE-specific options in addition to the common options in the table editor.

Bulk loader SAP ASE option descriptions

Option	Description
Mode	Specifies how the software loads data to the target table. • *Append*: Adds new records to the table. • *Truncate*: Deletes all existing records in the table and then adds new records.
Bulk load	Specifies whether to use SAP ASE bulk loading options to write data. • Selected: Uses SAP ASE bulk loading options to write data. • Not Selected: Does not use bulk loading options to write data.
Rows per commit	Specifies the number of rows the software processes before it commits data. Enter a positive integer or accept the default. The default is 1000 rows.
Maximum rejects	Specifies the number of warnings for rejected records the software encounters before it stops bulk loading. • Enter a positive integer: The software performs bulk loading until it exceeds the set number of rejected record warnings. Then it stops bulk loading. Useful when you do not expect any rejected rows, but you want to verify that the software loads the correct file and table. • Enter 0 or leave blank: The software keeps bulk loading even when there are reject warnings. A rejected row contains data that does not match the expected format or information specified for the table being loaded.

Option	Description
Network packet size	Specifies a network packet size in Kilobytes. Enter a positive integer. The default setting is 4 KB. While loading, the client caches rows until it fills a network packet or reaches the number of rows set for *Rows per commit*. If the client reaches the number of rows set for *Rows per commit* first, it sends the packet to the server regardless of whether the network packet is full. Improve performance by tuning the *Rows per commit* and the *Network packet size* settings. See the *Performance Optimization Guide* for more information.

Parent topic: Types of target tables [page 167]

Related Information

DB2 target table options [page 169]
Informix target table options [page 173]
Microsoft SQL Server target table options [page 176]
Netezza target table options [page 180]
ODBC target table options [page 182]
Oracle target table options [page 183]
SAP Sybase IQ target table options [page 188]
Teradata target table options [page 191]

2.2.25.5.8 SAP Sybase IQ target table options

When you use SAP IQ for the target table, complete the SAP IQ-specific options in addition to the common options in the target editor.

For more information about bulk loading with SAP Sybase IQ, see the *Performance Optimization Guide*.

Bulk loader SAP IQ option descriptions

Option	Description
Bulk load	Specifies whether to use SAP IQ bulk loading options to write data. • Selected: Uses SAP IQ bulk loading options to write data. • Not selected: Does not use SAP IQ bulk loading options to write data.

Option	Description
SAP Sybase IQ checkpoint	Specifies whether the software uses the SAP IQ checkpoint as part of the `LOAD TABLE SQL` statement for executing the bulk load. • Selected: Uses SAP IQ checkpointing. • Not selected: Does not use SAP IQ checkpointing. Not selected is the default setting.
Binary format	Specifies whether the software loads the staging and target tables in binary format. • Selected: Loads the staging and target tables in binary format. May improve performance. • Not selected: Loads the staging and target tables in delimited format. Useful if you want to view the data.
Ignore conversion error	Specifies whether the software ignores SAP IQ conversion errors. • Selected: Sets the SAP IQ database option Conversion_Error to off and ignores conversion errors. • Not selected: Executes with the current database value for Conversion_Error. For more information about the conversion error setting, see your SAP IQ documentation. The software reports conversion errors as warnings.
Mode	Specifies the mode to use for loading data into the target table. • *Append*: Adds new records to the table. • *Truncate*: Deletes all existing records in the table and then adds new records.
Block size (bytes)	Specifies the size of a block in bytes. Enter a positive integer. The default is 500,000 bytes. **i Note** If you make inappropriate adjustments to this option, you can dramatically affect performance.
File Options	
Field delimiter	Specifies the character to use to delimit columns in a row. Enter a single character, including non printable characters. Represent non printable characters using the ASCII number preceded by a forward slash. For example /95 is the ASCII entry for an underscore. Default is /127
Row delimiter	Specifies the character to use to delimit rows in a table. The character indicates where one row of data ends and the next begins. Enter a single character or a string of up to 4 hexadecimal ASCII characters preceded by a backward slash. The default is \n. **i Note** Ensure that the field and row delimiters are different values.

Option	Description
Generate files only	Specifies to generate a data file and a control file instead of bulk loading to the target specified in the data flow. • Selected: Generates a data file and a control file. Use these files for targets of SAP IQ bulk loading. Useful when SAP IQ server is on a system that runs a different operating system than the Job Server. The software writes the data file and the control file in the bulk loader directory that you specify in the datastore definition. If you have not specified a bulk loader directory, the software writes the files in the `<DS_COMMON_DIR>\log\bulkloader` directory. To load the data, manually copy the data files to the remote system and start the bulk load execution. • Not selected: Does not generate a data file and a control file for bulk loading.
Clean up bulk loader directory after load	Specifies whether the software deletes files from the bulk loader directory after a load completes successfully. • Selected: Deletes the data file and the auxiliary files in the bulk loader directory after a load completes successfully. • Not selected: Does not delete the files in the bulk loader directory after a load completes successfully.
Error Handling	
Constraints	Specifies the constraint violation types for which the software logs errors. • *Unique* • *Null* • *Data value* • *Foreign key* • *All*
Ignore errors	Specifies the maximum number of violations to ignore for any selected constraint before stopping the job. Enter a positive integer. If you enter 0, the software ignores all errors of the constraint type or types you chose.
Log errors	Specifies the type or types of constraint violations the software includes in the message and row log files.

Parent topic: Types of target tables [page 167]

Related Information

DB2 target table options [page 169]
Informix target table options [page 173]
Microsoft SQL Server target table options [page 176]
Netezza target table options [page 180]
ODBC target table options [page 182]

Oracle target table options [page 183]
SAP ASE target table options [page 187]
Teradata target table options [page 191]

2.2.25.5.9 Teradata target table options

When you use Teradata as a target in the data flow, complete the Teradata-specific options in addition to the common options in the target editor.

On the *Bulk Loader Options* tab, choose one of the following bulk loading and reading tools to use.

- FastLoad
- MultiLoad
- TPump
- Parallel Transporter
- Load Utility
- None

FastLoad, MultiLoad, TPump, and Parallel Transporter bulk loaders include several attributes. The software displays the attributes to complete based on the bulk loading and reading tool you select. Attribute names in bold indicate that a value is required.

For details about the Teradata bulk loading options, see your Teradata documentation. For more information about bulkloading with Teradata, and for information about each bulk loader and reader tool to use, see the *Performance Optimization Guide*.

Common Teradata bulk loader option descriptions

Option	Description
File option	Specifies the type of file that contains the data to bulk load. • *Data File* • *Generic Named Pipe* • *Named Pipes Access Module*
Data Services options	

Option	Description
Generate files only	Specifies whether the software generates a control file and a data file that you later load using Teradata bulk loading. • Not selected: Does not generate a data file and a script file to later use for bulk loading. • Selected: Generates a data file and a script file, and ignores the *Number of loaders* setting in the *Options* tab. Useful when the target database is on a system that runs a different operating system than the operating system of the Job Server. The software writes the data file and the control file to the bulk loader directory that you specify in the datastore definition. If you did not specify a location, or you accepted the default, the software uses the default location: `<DS_COMMON_DIR>\log\bulkloader`. Manually copy the data file and the control file to the remote system. The software uses the following naming conventions for these files: `<<DatastoreName_OwnerName_TableName_n>.ctl` and `<DatastoreName_OwnerName_TableName_n>.dat`. • `<OwnerName>` is the table owner • `<TableName>` is the target table • `<n>` is a positive integer, optionally used to guarantee that the software does not overwrite a pre-existing file
Clean up bulk loader directory after load	Specifies that the software deletes all bulk loader-related files (script, data files, temporary file) after the load completes successfully. • Selected: Deletes all bulk loader-related files from the bulk loader directory after the load completes successfully. • Not selected: Does not delete all bulk loader-related files from the bulk loader directory. If an error occurs during bulk loading, the software does not delete script and data files. Common errors: • Syntax errors in the script. • Error tables contain rows that cannot be inserted into the target table due to data conversion or constraint violation.
Mode	Specifies the bulk loading mode the software uses for loading data into the target table. • *Append*: Adds new records to the table. • *Replace*: Deletes all existing records in the table and inserts the loaded data as new records.
Field delimiter	Specifies a single character to delimit fields in the data. Specify a single character or an ASCII non printable character. The default value for this options is /127.

Option	Description
Bulk operation	Specifies the bulk operation the software uses. • *Insert*: Inserts rows. • *Upsert* : Updates existing rows and inserts new rows. Applicable only for MultiLoad, TPump, and Parallel Transporter bulk loading and reading tools. For Parallel Transporter, the option applies to Update and Stream operators.
Named pipes access module	Specifies whether to use named pipe options for the Teradata Access Module. • Selected: Activates the named pipe options to complete. • Not selected: Does not activate the named pipe options. Applicable only for MultiLoad, TPump, FastLoad, and Parallel Transporter bulk loading and reading tools. Options include the following: • *Logdirectory* • *Loglevel* • *Blocksize* • *FallbackFilename* • *FallbackDirectory* • *SignatureChecking* Override the default settings for these options, or set your own values. The Teradata Access Module creates a log file to record the load status. The Access Module log file differs from the tbuild log that you specify in the *Log directory* option. The Teradata Access Module writes information to fallback data files. If the job fails, the Teradata Access Module uses the log file and fallback data files to restart the load. For more information about these Access Module parameters, see your Teradata tools and utilities documentation.

Teradata FastLoad bulk loader options [page 194]
When you choose Teradata FastLoad for bulk loading and reading, complete the applicable options in the target editor.

Teradata MultiLoad bulk loader options [page 196]
When you choose Teradata MultiLoad for bulk loading and reading, complete the applicable options in the target editor.

Teradata TPump bulk loader options [page 197]
When you choose Teradata TPump for bulk loading and reading, complete the applicable options in the target editor.

Teradata Parallel Transporter bulk loader options [page 198]
When you choose Teradata Parallel Transporter for bulk loading and reading, complete the applicable options in the target editor.

Teradata Load utilities options [page 200]
Set Teradata Load utility options when you select certain bulk loader and reader tools.

Parent topic: Types of target tables [page 167]

Related Information

DB2 target table options [page 169]
Informix target table options [page 173]
Microsoft SQL Server target table options [page 176]
Netezza target table options [page 180]
ODBC target table options [page 182]
Oracle target table options [page 183]
SAP ASE target table options [page 187]
SAP Sybase IQ target table options [page 188]
Bulk loading and reading in Teradata

2.2.25.5.9.1 Teradata FastLoad bulk loader options

When you choose Teradata FastLoad for bulk loading and reading, complete the applicable options in the target editor.

FastLoad bulk loader option descriptions

FastLoad paramaters	Description
Data encryption	Specifies whether the software encrypts the requests. • Selected: Encrypts the requests. • Not selected: does not encrypt the requests.
Print all requests	Specifies whether the software prints every request that it sends to the Teradata database. • Selected: Prints all requests it sends to the Teradata database. • Not selected: Does not print the requests it sends to the Taradata database.

FastLoad paramaters	Description
Buffer size	Specifies a buffer size for output of FastLoad messages to the Teradata database. Specify a positive integer for number of kilobytes. The maximum allowable buffer size is 63 KB. The default value is 63 KB. The output buffer size and the size of the rows in the Teradata FastLoad table determine the maximum number of rows that the software includes in each message to the database. A larger buffer size reduces processing overhead by including more data in each message. If you specify a value greater than 63 KB, Teradata FastLoad does the following: • Responds with a warning message • Resets the buffer size back to the default value • Continues with the Teradata FastLoad job
Character set	Specifies the character set specification to use for mapping between characters and byte strings for the target table. For example, ASCII or UTF-8.

Parent topic: Teradata target table options [page 191]

Related Information

Teradata MultiLoad bulk loader options [page 196]
Teradata TPump bulk loader options [page 197]
Teradata Parallel Transporter bulk loader options [page 198]
Teradata Load utilities options [page 200]

2.2.25.5.9.2 Teradata MultiLoad bulk loader options

When you choose Teradata MultiLoad for bulk loading and reading, complete the applicable options in the target editor.

MultiLoad bulk loader option descriptions

MultiLoad parameters	Description
Reduced print output	Specifies to limit the print output to the minimal information required to determine the success of the job. • Selected: Limits the print output to the minimal information required to determine job success. • Not selected: Does not limit the print output to the minimal information required to determine job success. Not selected is the default.
Data encryption	Specifies whether the software encrypts all data and requests in all sessions used by the job. • Selected: Encrypts all data and requests for all sessions. • Not selected: Does not encrypt data and requests for all sessions. Not selected is the default.
Character set	Specifies the character set specification to use for mapping between characters and byte strings for the target table. For example, ASCII or UTF-8.

Parent topic: Teradata target table options [page 191]

Related Information

Teradata FastLoad bulk loader options [page 194]
Teradata TPump bulk loader options [page 197]
Teradata Parallel Transporter bulk loader options [page 198]
Teradata Load utilities options [page 200]

2.2.25.5.9.3 Teradata TPump bulk loader options

When you choose Teradata TPump for bulk loading and reading, complete the applicable options in the target editor.

TPump bulk loader option descriptions

TPump parameters	Description
Reduced print output	Specifies to limit the print output to the minimal information required to determine the success of the job. • Selected: Limits the print output to the minimal information required to determine job success. • Not selected: Does not limit the print output to the minimal information required to determine job success. Not selected is the default.
Retain macros	Specifies to keep macros that the software creates during the job run. • Selected: Keeps macros that the software creates during the job run. Use the macros as predefined macros for subsequent runs of the same job. • Not selected: Does not keep macros.
Data encryption	Specifies whether the software encrypts all data and requests in all sessions used by the job. • Selected: Encrypts all data and requests for all sessions. • Not selected: Does not encrypt data and requests for all sessions. Not selected is the default.
Print all requests	Specifies whether the software operates in verbose mode and prints every request sent to the Teradata database. • Selected: Operates in verbose mode and prints all requests. Provides additional statistical data in addition to the regular statistics. • Not selected: Does not operate in verbose mode.
Number of buffers	Specifies the number of request buffers that TPump uses for SQL statements to maintain the Teradata database. The default is 2. The maximum setting is 10.
Periodicity value	Specifies the rate at which TPump transfers SQL statements to the Teradata database. Specify the number of periods per minute. Enter from 1 to 600. The default value is 4, which allows 4 15-second periods per minute. However, to improve TPump work flow, adjust to a value from 1 to 30. The setting affects the rate when the Teradata `BEGIN LOAD` command uses the `RATE` parameter to control the rate.

TPump parameters	Description
Character set	Specifies the character set specification to use for mapping between characters and byte strings for the target table. For example, ASCII or UTF-8.
Configuration file	Specifies the configuration file that contains various configuration and tuning parameters for TPump.

Parent topic: Teradata target table options [page 191]

Related Information

Teradata FastLoad bulk loader options [page 194]
Teradata MultiLoad bulk loader options [page 196]
Teradata Parallel Transporter bulk loader options [page 198]
Teradata Load utilities options [page 200]

2.2.25.5.9.4 Teradata Parallel Transporter bulk loader options

When you choose Teradata Parallel Transporter for bulk loading and reading, complete the applicable options in the target editor.

Parallel Transporter bulk loader option descriptions

Parallel Transporter option	Description
Operator	Specifies the Parallel Transporter operator values. • *Load* • *Update* • *Stream* • *SQL Inserter* Note that the attributes vary depending on the *Operator* that you select. Refer to your Teradata documentation for information on attributes.
Number of instances	Specifies the number of load operator instances in the loading scripts. The software includes the information in the generated Parallel Transporter script. Enter a positive integer. Applicable when you choose *Data File* or *Named pipes access module* for *File Option* under the Common settings. For *Named pipes access module*, the setting for Number of instances sets the number of named pipes, and the setting for *Number of DataConnector instances* does not apply.

Parallel Transporter option	Description
Number of DataConnector instances	Specifies the number of read operator instances in the loading scripts. The software includes this information in the generated Parallel Transporter script. Enter a positive integer. Enter a value that is less than or equal to the number of data files. Applicable when you choose *Data File* for *File Option* under the Common settings.
tbuild parameters	
Debug all tasks	Specifies whether the software enables debug trace functions for all tasks. • Selected: Enables debug trace functions for all tasks. Outputs termination return codes that help with script debugging. Corresponds to the tbuild -d option. • Not selected: Does not enable debug trace functions for all tasks.
Trace all tasks	Specifies whether the software enables the trace option for all tasks. • Selected: Enables the trace option for all tasks. Corresponds to the tbuild -t option. • Not selected: Does not enable the trace option for all tasks.
Latency interval (sec)	Specifies the number of seconds between each flush of stale buffers. Enter a positive integer for the number of seconds. Corresponds to tbuild -l option.
Checkpoint interval (sec)	Specifies the number of seconds between checkpoints. Enter a positive integer for the number of seconds. The default is 10 seconds. Corresponds to tbuild -z option.

Parent topic: Teradata target table options [page 191]

Related Information

Teradata FastLoad bulk loader options [page 194]
Teradata MultiLoad bulk loader options [page 196]
Teradata TPump bulk loader options [page 197]
Teradata Load utilities options [page 200]

2.2.25.5.9.5 Teradata Load utilities options

Set Teradata Load utility options when you select certain bulk loader and reader tools.

Load Utilities

Option	Description
Command line	Specifies a call to a custom script for Load utilities. Load utilities include FastLoad, MultiLoad, and TPump. **Example** For example for FastLoad, you could enter: `fastload < C:\Teradata\FLScripts\myScript.ctl` Data Services does not parse or modify these scripts.
Named pipe name	Specifies the pipe name for a Load Utility. Applicable when you choose *Named Pipes Access Module* or *Generic Named Pipes* for the *File option* option in the common bulk loader options. On UNIX, the pipe is a FIFO (first in, first out) file name that uses the following format: `/temp/<filename>.dat`. On Windows, the pipe name has the following format: `\\.\pipe\<datastorename_ownername_tablename_loadernum>.dat`

Parent topic: Teradata target table options [page 191]

Related Information

Teradata FastLoad bulk loader options [page 194]
Teradata MultiLoad bulk loader options [page 196]
Teradata TPump bulk loader options [page 197]
Teradata Parallel Transporter bulk loader options [page 198]

2.2.25.6 Target Data Transfer files and tables

The Data Transfer transform enables the software to push down certain operations to the database server for more efficient processing.

A data flow writes data from a source into the Data Transfer transform. The transform uses a transfer object such as a relational database table or file to push down operations to the database server. When you set the *Transfer type* option in the Data Transfer transform to *Table*, the software displays additional tabs for configuring the transform behavior as a target object.

The software displays the following tabs for the Data Transfer target table:

- Options
- Bulk Loader Options
- Pre-Load Commands
- Post-Load Commands

For descriptions of the target editor tabs for the Data Transfer transform, see the Data_Transfer transform in the Transform section.

Parent topic: Target [page 147]

Related Information

Target objects [page 148]
Target files [page 149]
Target persistent cache tables [page 150]
Common Target Table editor options [page 152]
Types of target tables [page 167]
Target JSON files, messages, and templates [page 201]
Target XML files, messages, and templates [page 207]
Target flat files [page 213]

2.2.25.7 Target JSON files, messages, and templates

When you use a JSON schema as a target in a data flow, complete the JSON-specific options in addition to the common options in the target editor.

You can also create a JSON file target without creating a format by using a Nested Schema template. The following table contains options from the *Target* tab of the nested schemas target file editor or the nested schemas target message editor.

JSON Nested Schema target option descriptions

Option	Description
Make port	Specifies whether to make the schema an embedded data flow port. • Selected: Makes the schema an embedded data flow port. • Not selected: Does not make the schema an embedded data flow port.

Option	Description
Parallel process threads	Specifies the number of threads for parallel processing. Enter a positive integer. Parallel processing improves performance by maximizing CPU usage on the Job Server computer. For example, if you have four CPUs, enter 4 for this option. Set this option to *{none}* if you use the data file to process USPS certification tests. For more information, see "File multi threading" in the *Performance Optimization Guide*. **! Restriction** If you set the number of rows for *Batch size (rows)* to a nonzero positive integer, Data Services ignores a setting greater than 1 for the *Parallel process threads* option. *Batch size (rows)* is in the *Data File* group, and it applies to flat files, XML files, and JSON files.
File location	Specifies the name of the file location object related to the schema. The file location object contains file transfer protocol information and local and remote server information. SAP Data Services uses this information in target mode to transfer a copy of the generated output file from a local server to a remote server safely. Completing this option is optional.

Option	Description
Batch size (rows)	Specifies the number of rows to load at a time into the temporary local file before uploading to your Google Cloud Storage (GCS) account. This option controls the amount of local disk space used during the upload process. After each load from the local file to GCS, Data Services deletes the rows in the local file. After Data Services deletes the rows in the local file, it loads the next set number of rows to the local file and continues in this manner until it loads all rows to GCS. Set to a positive integer or zero (0): • **Positive integer:** Loads the set number of rows to the local file before uploading to GCS. Uses local disk space equal to the size of the set number of rows. ℹ Note Upload is single-threaded so Data Services ignores any setting greater than 1 for the *Parallel process threads* option in the *General* group. ℹ Note Upload does not use compression so Data Services ignores a setting of *gzip* for *Compression type* in the GCS file location. • **0:** Disables the option. Uploads all rows to the local temporary file before uploading to GCS. Uses the settings in the file format and in the GCS file location to upload rows from the local file to GCS. Uses local disk space equal to the size of the entire file. Zero (0) is the default setting. Applicable only for uploading to GCS and only for flat files, XML files, and JSON files. Applicable for target mode only. ℹ Note If you set *Location* or *File location* to a location and file name for a non-GCS file location object, Data Services ignores the setting in *Batch size (rows)*.

Option	Description
Delete file after transfer	Specifies that Data Services deletes the contents of the local server file after Data Services transfers the generated output data from the local file to the remote file. • *Yes*: Deletes the contents of the local server file after Data Services transfers the generated output data from the local file to the remote file. • *No*: Saves the contents of the local server file after Data Services transfers the generated output data from the local file to the remote file. Available only when you select a file location object for the *File Location* option.
File	Specifies the location relative to the Job Server of the file to use as the target data file. For JSON file targets only. If the file does not exist, Data Services creates it. If the file exists, Data Services overwrites the file with the output from the data flow. **i Note** If your Job Server is on a different computer than Data Services, the option `<Select file>` from the drop-down list is not available. Therefore there is not an option to browse for the file. Type an absolute path or a relative path that the Job Server can access. You can alternatively choose a variable. When you use a JSON file as a target, and your target location is a cloud storage, specify the full remote path and file name. Include all subfolders in the full remote path. Supported cloud storages include Amazon S3, Azure Blob Storage, Azure Data Lake Store, Hadoop File System, and Google Cloud Storage.

Option	Description
Delete and re-create file	Specifies whether to delete the target file and create a file containing the generated output. When you do not select a file location object in *File location*: • Selected: Deletes the target file and creates a file that contains the generated output. Overrides the default behavior. • Not selected: Appends generated data to the data in the existing file. When you choose a file location object in *File location*, this setting affects the file in the remote server. Data Services outputs data to the local data file and then securely transfers the data to the remote data file. • Selected: Applicable for FTP, SFTP, and SCP. Overwrites existing data in the remote server file with generated output data from the local server. • Not selected: Not applicable for SCP and Local. Appends generated output data from the local file to an existing file in the remote server. **i Note** For all file transfer protocols except Local, if the file doesn't already exist in the remote server, Data Services creates a file and populates it with generated output data from the local server.
Enable validation	Specifies whether Data Services compares the outgoing data to the stored Schema created by the JSON file. • Not Selected: Data Services does not compare the outgoing data to the stored Schema created by the JSON file. • Selected: Data Services compares the outgoing data to the stored Schema created by the JSON file. Data Services issues an exception if the outgoing data is not valid. Useful when you develop real-time jobs to determine that sample data is valid and well formed. For production jobs, ensure that you include appropriate error handling either in the job or the client application when Data Services finds data that does not compare to the Schema.

Option	Description
Test file	Specifies the location relative to the Job Server of a file to use as the message target when you run the job in test mode. **i Note** If the file does not exist, Data Services creates it. If the file exists, Data Services overwrites the file with the output from the data flow. **i Note** If your Job Server is on a different computer than Data Services, the option `<Select file>` from the drop-down list is not available. Therefore there is not option to browse for the file. Type an absolute path or a relative path that the Job Server can access. You can alternatively choose a variable. Applicable only for message targets.
Test file type	Specifies the type of file for the test file. • XML • JSON

Parent topic: Target [page 147]

Related Information

Target objects [page 148]
Target files [page 149]
Target persistent cache tables [page 150]
Common Target Table editor options [page 152]
Types of target tables [page 167]
Target Data Transfer files and tables [page 200]
Target XML files, messages, and templates [page 207]
Target flat files [page 213]
File location object [page 98]

2.2.25.8 Target XML files, messages, and templates

When you use an XML Schema or DTD format as a target in a data flow, complete the related options in addition to the common options in the target editor.

XML, messages, templates target option descriptions

Option	Description
Make port	Specifies whether to make the target file an embedded data flow port. • Selected: Makes the target file an embedded data flow port. • Not selected: Does not make the target file an embedded data flow port.
Parallel process threads	Specifies the number of threads for parallel processing. Enter a positive integer. Parallel processing improves performance by maximizing CPU usage on the Job Server computer. For example, if you have four CPUs, enter 4 for this option. Set this option to *{none}* if you use the data file to process USPS certification tests. For more information, see "File multi threading" in the *Performance Optimization Guide*. **! Restriction** If you set the number of rows for *Batch size (rows)* to a nonzero positive integer, Data Services ignores a setting greater than 1 for the *Parallel process threads* option. *Batch size (rows)* is in the *Data File* group, and it applies to flat files, XML files, and JSON files.
File location	Specifies the name of the file location object related to the schema. The file location object contains file transfer protocol information and local and remote server information. Data Services uses this information in target mode to safely transfer a copy of the generated output file from a local server to a remote server. This is an optional setting.

Option	Description
Batch size (rows)	Specifies the number of rows to load at a time into the temporary local file before uploading to your Google Cloud Storage (GCS) account. This option controls the amount of local disk space used during the upload process. After each load from the local file to GCS, Data Services deletes the rows in the local file. After Data Services deletes the rows in the local file, it loads the next set number of rows to the local file and continues in this manner until it loads all rows to GCS. Set to a positive integer or zero (0): • **Positive integer:** Loads the set number of rows to the local file before uploading to GCS. Uses local disk space equal to the size of the set number of rows. ℹ Note Upload is single-threaded so Data Services ignores any setting greater than 1 for the *Parallel process threads* option in the *General* group. ℹ Note Upload does not use compression so Data Services ignores a setting of *gzip* for *Compression type* in the GCS file location. • **0:** Disables the option. Uploads all rows to the local temporary file before uploading to GCS. Uses the settings in the file format and in the GCS file location to upload rows from the local file to GCS. Uses local disk space equal to the size of the entire file. Zero (0) is the default setting. Applicable only for uploading to GCS and only for flat files, XML files, and JSON files. Applicable for target mode only. ℹ Note If you set *Location* or *File location* to a location and file name for a non-GCS file location object, Data Services ignores the setting in *Batch size (rows)*.

Option	Description
Delete file after transfer	Specifies that Data Services deletes the contents of the local server file after Data Services transfers the generated output data from the local file to the remote file. • *Yes*: Deletes the contents of the local server file after Data Services transfers the generated output data from the local file to the remote file. • *No*: Saves the contents of the local server file after Data Services transfers the generated output data from the local file to the remote file. Available only when you select a file location object for the *File Location* option.
File	Specifies the location relative to the Job Server of the file to use as the target data file. For JSON File targets only. If the file does not exist, Data Services creates it. If the file exists, Data Services overwrites the file with the output from the data flow. **i Note** If your Job Server is on a different computer than Data Services, the option `<Select file>` from the drop-down list is not available. Therefore there is not option to browse for the file. Type an absolute path or a relative path that the Job Server can access. You can alternatively choose a variable. When you use an XML file as a target, and your target location is a cloud storage, specify the full remote path and file name. Include all subfolders in the full remote path. Supported cloud storages include Amazon S3, Azure Blob Storage, Azure Data Lake Store, Hadoop File System, and Google Cloud Storage. Applicable for file targets only.

Option	Description
XML test file	Specifies the location relative to the Job Server of a file to use as the message target when you run the job in test mode. **i Note** If the file does not exist, Data Services creates it. If the file exists, Data Services overwrites the file with the output from the data flow. **i Note** If your Job Server is on a different computer than Data Services, the option `<Select file>` from the drop-down list is not available. Therefore there is not option to browse for the file. Type an absolute path or a relative path that the Job Server can access. You can alternatively choose a variable. Applicable only for message targets.
Delete and re-create file	Specifies whether to delete the target file and create a new file containing the generated output. When you choose a file location object in *File location*, this setting affects the file in the remote server. Data Services outputs data to the local data file and then securely transfers the data to the remote data file. • Selected: Applicable for FTP, SFTP, and SCP. Overwrites existing data in the remote server file with generated output data from the local server. • Not selected: Not applicable for SCP and Local. Appends generated output data from the local file to an existing file in the remote server. **i Note** For all file transfer protocols except Local, if the file doesn't already exist in the remote server, Data Services creates a file and populates it with generated output data from the local server.

Option	Description
Print comment	Specifies whether Data Services includes a comment in the target file that identifies the data as having been processed by Data Services. • Selected: Includes a comment in the target file that identifies the data as having been processed by Data Services. • Not selected: Does not include a comment.
Replace NULL or blank	Specifies whether Data Services replaces Null or blank values in element data with a specified value. • Selected: Replaces Null or blank values in element data with a specified value. Specify the value in the *With* text box. • Not selected: Does not replace Null or blank values in element data with a specified value.
Enable validation	Specifies whether Data Services compares the outgoing data to the stored XML Schema or DTD from which this XML file was created. • Not selected: Does not compare the outgoing data to the stored XML Schema or DTD from which this XML file was created. • Selected: Compares the outgoing data to the stored XML Schema or DTD from which this XML file was created. Data Services issues an exception if the outgoing data is not valid. Useful when you develop real-time jobs to determine that sample data is valid and well formed. For production jobs, ensure that you include appropriate error handling either in the job or the client application for when Data Services finds data that does not validate against the imported XML Schema or DTD.
Include schema location	Specifies whether to include the schema location in the XML output. • Selected: Includes the schema location in the XML output. Selected is the default setting. • Not selected: Does not include schema location in the XML output.

Option	Description
Include DTD	Specifies whether Data Services include the DTD format in the target file or message output. • Selected: Includes the DTD format in the target file or message output. • Not selected: Does not include the DTD format in the target file or message output. Applicable for targets created only from DTD formats.
XML encoding	Specifies an XML encoding for the XML target file. Select a value from the dropdown list. If you do not select a value, Data Services uses the encoding in the XML header field. If the XML header field is empth, Data Services uses UTF-8. You can save XML file targets using different encoding or code page than Data Services system locale. XML message source and target encodings default to UTF-8 and cannot be changed.
XML header	Specifies whether to use a unique XML header for each target file First enter the header information to use in the target. Then edit the information from this field. If you need to edit only the XML encoding for this file target, use the *XML encoding* option. However, if your header information contains additional information, enter the header in this field.
DTD file in DOCTYPE	Specifies the path to a DTD format so that Data Services includes the information in the DOCTYPE element in the XML target. Enter the path or browse to select the path. Available only for targets created from DTD formats.
Format name	Specifies the name of the DTD or XML Schema format that Data Services uses for this job. The information is read only.
Root element name	Specifies the name of the root element in the DTD or XML schema. The information is read only.
Namespace	Specifies the name space in the XML schema. The information is read only.

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.

> **i Note**
>
> When the target file format contains a connection to a file location object, Data Services uses the information in the file location object for file transfer protocol to move the output file from the local server to the defined remote server.

Parent topic: Target [page 147]

Related Information

Target objects [page 148]
Target files [page 149]
Target persistent cache tables [page 150]
Common Target Table editor options [page 152]
Types of target tables [page 167]
Target Data Transfer files and tables [page 200]
Target JSON files, messages, and templates [page 201]
Target flat files [page 213]
Nested Schemas template [page 114]
Locales and multi-byte functionality [page 1437]
File location object [page 98]

2.2.25.9 Target flat files

When you use a flat file format as a target in a data flow, complete the related options in addition to the common options in the target editor.

The following table contains the option in the target editor specific for a flat file. For descriptions of all options, see the sections under File format [page 71].

Flat file target editor options

Option	Descriptions
Make port	Specifies whether to make the target file an embedded data flow port. • Selected: Makes the target file an embedded data flow port. • Not selected: Does not make the target file an embedded data flow port.

Option	Descriptions
Parallel process threads	Specifies the number of threads for parallel processing. Enter a positive integer. Parallel processing improves performance by maximizing CPU usage on the Job Server computer. For example, if you have four CPUs, enter 4 for this option. Set this option to *{none}* if you use the data file to process USPS certification tests. For more information, see "File multi threading" in the *Performance Optimization Guide*. **! Restriction** If you set the number of rows for *Batch size (rows)* to a nonzero positive integer, Data Services ignores a setting greater than 1 for the *Parallel process threads* option. *Batch size (rows)* is in the *Data File* group, and it applies to flat files, XML files, and JSON files.
Location	Specifies the name of the file location object related to the schema. The file location object contains file transfer protocol information and local and remote server information. Data Services uses this information in target mode to safely transfer a copy of the generated output file from a local server to a remote server. This is an optional setting.

Option	Descriptions
Batch size (rows)	Specifies the number of rows to load at a time into the temporary local file before uploading to your Google Cloud Storage (GCS) account. This option controls the amount of local disk space used during the upload process. After each load from the local file to GCS, Data Services deletes the rows in the local file. After Data Services deletes the rows in the local file, it loads the next set number of rows to the local file and continues in this manner until it loads all rows to GCS. Set to a positive integer or zero (0): • **Positive integer:** Loads the set number of rows to the local file before uploading to GCS. Uses local disk space equal to the size of the set number of rows. ℹ Note Upload is single-threaded so Data Services ignores any setting greater than 1 for the *Parallel process threads* option in the *General* group. ℹ Note Upload does not use compression so Data Services ignores a setting of *gzip* for *Compression type* in the GCS file location. • **0:** Disables the option. Uploads all rows to the local temporary file before uploading to GCS. Uses the settings in the file format and in the GCS file location to upload rows from the local file to GCS. Uses local disk space equal to the size of the entire file. Zero (0) is the default setting. Applicable only for uploading to GCS and only for flat files, XML files, and JSON files. Applicable for target mode only. ℹ Note If you set *Location* or *File location* to a location and file name for a non-GCS file location object, Data Services ignores the setting in *Batch size (rows)*.
File name(s)	Name of the temporary local file for staging.

Parent topic: Target [page 147]

Related Information

Target objects [page 148]
Target files [page 149]
Target persistent cache tables [page 150]
Common Target Table editor options [page 152]
Types of target tables [page 167]
Target Data Transfer files and tables [page 200]
Target JSON files, messages, and templates [page 201]
Target XML files, messages, and templates [page 207]

2.2.26 Template table

A template table provides a quick way to add a target table to a data flow without having to define a schema or import metadata.

Template table characteristics

Characteristic	Description
	Template table icon.
Class	Reusable
Access	Use a template table as a target in a data flow, and then as a source. • As a target, open a data flow in the work space, click the template table icon in the tool palette, and click anywhere in the data flow. Connect to the data flow as a target. • As a source, open a data flow in the workspace, open the *Datastores* tab in the object library, select the desired template table, and drag onto the workspace. Connect to the data flow as a source. • To view and set template table options, open the template table connected to the data flow in the workspace or in the project area. The software opens the source or target object editor. To view template tables in a specific datastore, first relate the target template to a specific database. The software saves the target template table as a template table under the applicable datastore.

Characteristic	Description
Description	Template tables are new tables that you add to a database. Use a new template table one time as a target and multiple times as a source. You cannot use a template table in an ABAP data flow. When you connect a template table to a data flow as a target and execute the data flow, the software automatically creates the table and schema in the target table. The software uses the related database management system to create the table based on the schema of the executed data flow. After you create a template table as a target in one data flow, use the same template table as a source in any other data flow. A template table is useful in the design and testing phases of your projects. When you modify the target schema of a template table, the software applies any changes to all other instances of the template table. During the validation process, the software warns you of any errors that result from changing the schema. When you first use the template table as a target, validate, save, and execute the data flow at least once. The template table is now established. Then you can use the template table as a source in different data flows. When running a job where you use an established template table as a target, use care when setting the target options. If you accept the default setting of selected for the *Drop and re-create table* option, the software deletes the existing table and creates a new one. If you uncheck the *Drop and re-create table* option, the software attempts to load data to the existing table. If the existing table schema does not match the schema generated in the data flow, the software generates runtime errors. Use the runtime errors to adjust the target schema and execute the job again.

Template table options [page 217]

When you use a template table as a target for the first time, the options are similar to options for a regular table target.

2.2.26.1 Template table options

When you use a template table as a target for the first time, the options are similar to options for a regular table target.

The following table describes the options in the *Create Template* dialog box. The dialog box appears when you add a *Template* icon from the tool palette to the workspace of a data store.

Option	Description
Template name	Specifies the name of the template table. Use standard naming conventions, such as no spaces. **i Note** SAP Data Services automatically appends the name of the related datastore and the owner name to the table name when you add the template to the data flow.
In datastore	Specifies the datastore related to the database type of the target table.
Owner	Specifies the owner related to the datastore. **i Note** Data Services automatically populates the owner name based on the datastore you select. You can enter a different name, if necessary.

Quote names	Specifies to preserve the casing of the entered name in *Template name*.
	
	• Selected: Preserves the casing of the template name.
	
	i Note
	Select *Quote names* if the database naming conventions require exact case.
	
	• Not selected: Doesn't preserve the casing of the template name. Template name appears in upper case in the following locations:
	  • Data flow
	  • Project area
	  • Template table node of the datastore in the object library
	The option isn't selected by default.

Parent topic: Template table [page 216]

2.2.27 Transform

Use transforms to define your data transformation requirements.

Transform characteristics

Characteristic	Description
	Transform icon.
Class	Reusable
Access	In the object library, click the *Transforms* tab.
Description	Transforms use the operation codes associated with each row of data that the software reads from a source. The descriptions of individual transforms indicate which operation codes the transforms ignore or use.

Transform attributes [page 219]

Transforms have build-in attributes that you can view through the *Properties* dialog box.

Related Information

Transforms [page 240]

2.2.27.1 Transform attributes

Transforms have build-in attributes that you can view through the *Properties* dialog box.

Transform attribute descriptions

Attribute	Description
Name	The name of the transform. This name appears on the object in the object library and in the calls to the object.
Description	SAP Data Services description of the transform. You can edit any transform description except for the Query transform description. The Query transform description is read only.

Parent topic: Transform [page 219]

2.2.28 Try

The Try object initiates the try/catch sequence where the software directs applicable data in the data flow to an alternative work flow when specific errors occur.

Characteristic	Description
	Try icon.
Class	Single-use
Access	With a work flow diagram in the workspace, click the try icon in the tool palette and click in the workspace.
Description	A try is part of a serial sequence called a try/catch block. The Try portion of a try/catch block initiates the process. The Try indicates to the software that data in the data flow is going to follow an alternative work flow when specified errors occur. Try/catch blocks catch classes of errors, apply solutions that you provide, and continues execution. In any given data flow, you use one Try object but you may include one or more Catch objects based on the purpose of the try/catch block. A Try object does not have an editor. Therefore, there are no options to set. If you plan to use automatic recovery, do not reference output variables from a try/catch block in any subsequent steps. Referencing such variables could alter the results during automatic recovery. Try object attributes **Attribute** — **Description** Name — The name of the object. The name appears on the object in the diagram.

Related Information

Catch [page 36]

2.2.29 While loop

A while loop object redirects data back to the beginning of a series of steps in a work flow as long as a set condition is true.

While loop object characteristics

Characteristic	Description
	While loop icon.
Class	Reusable
Access	With a work flow diagram in the workspace, click the while loop icon in the tool palette and click in the workspace.
Description	A while loop repeats a sequence of steps as long as a condition is true. Design the sequence of steps during the while loop so that the condition changes at some point. If you set the sequence so it doesn't change the condition, data continues in an infinite loop. Once the condition resolves, the data can continue through the rest of the data flow. For each while loop, provide the following information: • Condition: Include a Boolean expression that the job uses to evaluate data. Construct the expression to evaluate to true or false. Use constants, functions, variables, parameters, and standard operators to construct the expression. Set of steps: Include a set of steps to complete when the condition is true. Any other valid object: Add any objects that are valid in the work flow such as scripts, work flows, and data flows. Connect these objects to represent the step order. **i Note** You can include the parent work flow in the while loop, however recursive calls can create an infinite loop.

Related Information

While loops

Smart Editor and the function wizard [page 1061]

2.2.30 Work flow

A work flow defines the decision making process for executing data flows.

Work flow object characteristics

Characteristic	Description
	Work flow icon.
Class	Reusable
Access	Create a work flow when you have a job opened in the workspace by clicking the icon in the tool pallet and clicking in the workspace. Access an existing work flow from the *Work Flows* tab in the object library
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 a special type of work flow. You can execute jobs just like you can execute work flows. However, a job does not have parameters. The following objects can be elements in work flows: • Work flows • Data flows • Scripts • Conditionals • While loops • Try/catch blocks. Work flows can call other work flows, and you can nest calls to any depth. A work flow can also call itself. The connections you make between the icons in the workspace determine the order in which work flows execute, unless the jobs containing those work flows execute in parallel. You can designate a work flow as a recovery unit. A recovery unit contains a work flow that has steps that depend on each other and therefore they execute together. Read more about recovery units in the *Designer Guide*.

Work flow attributes [page 223]
View work flow attributes in the *Properties* dialog box.

Related Information

Marking recovery units
Work flows

2.2.30.1 Work flow attributes

View work flow attributes in the *Properties* dialog box.

The following table describes the options in the General tab of the Properties dialog box for a work flow.

Attribute	Description
Name	Specifies the name of the work flow object. This name appears on the object in the object library, in the workspace for the job that contains the work flow, and in the calls to the object.
Execution type	Specifies the type of work flow: • *Regular*: Runs a work flow based on the checkbox selection. • *Continuous*: Runs all data flows in a loop and keeps them in memory for the next interation. • *Single*: Runs all data flow in one operating system process.
Execute only once	Specifies to run the contents of a regular work flow only one time.
Recover as a unit	Specifies to run the work flow as a recovery unit.
Description	Describes the work flow. Enter a description to help you remember the purpose of the work flow.
Bypass	Specifies to bypass the execution of a work flow during design time. Leave the default setting of *No Bypass*, or select a predefined substitution parameter from the list. For details, see the Design and debug section of the *Designer Guide*.

If you select Continuous for Execution type, open the Continuous Options tab to complete the options as described in the following table.

Option	Description
After number of runs	Specifies to release memory resources after a set number of work flow runs. Enter an integer that represents the number of times Data Services runs the work flow before it releases the resources from memory. The default is 100 runs.
After *Number of hours*	Specifies to release memory resources after a set number of hours. Enter an integer that represents the number of hours. The default is 1 hour.

Option	Description
After *Number of days*	Specifies to release memory resources after a set number of days. Enter an integer that represents the number of days. The default is 1 day.
After *Result of the function is not equal to zero*	Specifies to release memory resources after a set custom function ≠ 0. Enter a function, or select *Function* to use the Smart Editor.
Stop *When result of the function is zero or less*	Specifies to stop the work flow after a set function is =< 0. Enter a function, or select *Function* to use the Smart Editor.

Parent topic: Work flow [page 222]

2.2.31 XML file

Represent an XML file as an XML Schema or a document type definition (DTD).

For complete details about XML files and how to use them in Data Services, see the *Designer Guide*.

XML file object characteristics

Characteristic	Description
 	XML file icons.
Class	Single-use
Access	In the object library, click the *Formats* tab, then open the Nested Schemas category.

Characteristic	Description
Description	Use XML documents (files or messages) as sources or targets in batch or real-time jobs. XML documents are hierarchical. The valid structure of an XML file is in separate format documents: • XML Schema `.xsd` file • Document type definition file `.dtd` The software stores the metadata in the XML Schema and DTD in the repository. When you import the metadata of a format document, the software structures it into an internal schema for hierarchical documents. The internal schema uses the nested relational data model (NRDM). The data that the software reads into or writes out of an XML file has 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.

XML file source or target [page 225]
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.

XML file mapping rules [page 226]
The software uses internal mapping rules to translate an XML Schema or DTD into the internal schema definition.

XML Properties [page 226]
View the XML file attributes in the *Properties* dialog box.

2.2.31.1 XML file source or target

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.

Associate a file location object to an XML source or target file to include file transfer protocol information. The file transfer protocol information determines how SAP Data Services transfers the XML file:

- Source: From a remote server to a local server
- Target: From a local server to a remote server

When you drop the XML schema or DTD format in the workspace, SAP Data Services prompts you to specify that the resulting object is a source or target file or a source or target message.

Parent topic: XML file [page 224]

Related Information

XML file mapping rules [page 226]

XML Properties [page 226]
File location object [page 98]
Target XML files, messages, and templates [page 207]

2.2.31.2 XML file mapping rules

The software uses internal mapping rules to translate an XML Schema or DTD into the internal schema definition.

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

Parent topic: XML file [page 224]

Related Information

XML file source or target [page 225]
XML Properties [page 226]
Format XML documents

2.2.31.3 XML Properties

View the XML file attributes in the *Properties* dialog box.

There are two tabs in the XML file Properties dialog: *General* and *Format*.

General tab option descriptions

XML property	Description
Name	Specifies the name of the XML schema file. Read only. This name appears in the object library under the Nested Schemas category of the *Formats* tab. The name appears for sources and targets (XML files or messages) that reference this format in data flows.
Description	Specifies the description of the XML schema file. You enter the text when you create the DTD format file.

Format tab option descriptions

XML property	Description
File Location	Specifies the name of an existing file location object. The file location object contains file transfer protocol information, such as FTP or SFTP. The file location also contains local and remote server information. The software uses this information in a data flow to copy the XML data from a remote to a local server (source) and from a local to a remote server (target).
Delete file after transfer	Specifies whether the software deletes the local file after the software finishes loading the data into the data flow. • Selected: Software deletes the file from the local server after it loads the file into the data flow. • Not selected: Software does not delete the file from the local server after it loads the file into the data flow. Not applicable for SCP file transfer protocol. Applicable only when you select a file location object in *File Location*.
File name	Specifies the name of the XML-formatted file to use as the source. Enter the location relative to the Job Server of the file. You can enter a variable for this option. **i Note** If your Job Server is on a different computer than the Designer, you cannot choose `<Select file>` to use Browse to specify the file path. Manually enter the path. You can type an absolute path or a relative path, but the Job Server must be able to access it. When you use an XML file as a source, and your source is in cloud storage, specify the full remote path and file name. Include all subfolders in the full remote path. Supported cloud storages include Amazon S3, Azure Blob Storage, Azure Data Lake Store, Hadoop File System, and Google Cloud Storage.
Imported from	Specifies the full path to the XML file. Read only.
Root element name	Specifies the name of the primary node of the XML file. Read only. SAP Data Services imports only elements of the format that belong to this node or any sub nodes.

XML property	Description
Namespace	Specifies the Namespace name associated with the imported XML schema. Read only

Parent topic: XML file [page 224]

Related Information

XML file source or target [page 225]
XML file mapping rules [page 226]
XML file source [page 142]

2.2.32 XML message

An XML message object allows you to indicate a real-time source or target in a job.

XML message object characteristics

Characteristic	Description
 XML Source  XML Target	XML message icons.
Class	Single-use
Access	In the object library, click the *Formats* tab, then open the Nested Schemas category.

Characteristic	Description
Description	When used as a source, an XML message object translates incoming XML-formatted messages into an internal SAP Data Services data set. When used as a target, an XML message object translates the data produced by a job, including nested data, into an XML-formatted message and sends the message to the Access Server. When a real-time job contains an XML message source, it must also contain an XML message target. The data read into or written out of an XML message must have a single row at the top-level table. When writing out an empty nested table, the software includes a single row of the nested table, with null values in each column of the table. To produce the metadata that describes the data that an XML message handles, the software reads the format for the XML message. The metadata is stored in the repository as an XML Schema or DTD. XML message properties are the same as those for its DTD and XML schema formats.

XML message: Source or target [page 229]
When you drop an XML message into the workspace, the software asks you if the resulting XML message is the source or target.

XML message test files [page 230]
Execute a real-time job in test mode to determine whether the XML message returns the expected results.

XML message mapping rules [page 230]
The software uses internal mapping rules to translate an XML Schema or DTD into the internal schema definition.

Related Information

DTD [page 58]
XML schema [page 231]
XML message source [page 143]
Target XML files, messages, and templates [page 207]
DTD Properties [page 58]
XML schema [page 231]

2.2.32.1 XML message: Source or target

When you drop an XML message into the workspace, the software asks you if the resulting XML message is the source or target.

You can insert an XML message into a real-time job by dragging an XML Schema or DTD format from the *Formats* tab of the object library into the workspace of a data flow.

Parent topic: XML message [page 228]

Related Information

XML message test files [page 230]
XML message mapping rules [page 230]

2.2.32.2 XML message test files

Execute a real-time job in test mode to determine whether the XML message returns the expected results.

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.

Parent topic: XML message [page 228]

Related Information

XML message: Source or target [page 229]
XML message mapping rules [page 230]

2.2.32.3 XML message mapping rules

The software uses internal mapping rules to translate an XML Schema or DTD into the internal schema definition.

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

See the Nested Data section of the *Designer Guide* for an introduction to the nested relational data model, which the software uses to generate an internal hierarchical schema.

Parent topic: XML message [page 228]

Related Information

XML message: Source or target [page 229]
XML message test files [page 230]
DTD [page 58]
XML schema [page 231]

2.2.33 XML schema

XML Schemas describe the data structure of an XML file or message.

XML schema object characteristics

Characteristic	Description
	XML schema icon.
Class	Reusable
Access	In the object library, click the *Formats* tab, then open the Nested Schemas category.
Description	SAP Data Services supports W3C XML Schemas Specification 1.0. Read about this XML Schema version on the following web site: https://www.w3.org/TR/xmlschema-1/ . 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. If you delete an XML Schema from the object library, XML sources or targets that are based on this format become invalid. The software marks the source or target objects with an icon that indicates the calls are no longer valid. Sample_order  To restore the invalid objects, delete the source or target and replace it with a source or target based on an existing XML Schema.

XML schema editor [page 232]
Open an XML schema editor by opening the schema in the object library.

XML schema properties [page 233]
View XML schema properties in the *Properties* dialog box.

XML schema supported attributes [page 234]
XML schema attributes include column and nested table attributes.

XML Schema elements [page 235]

SAP Data Services maps XML schema elements to attributes when they are imported as metadata.

XML Schema attributes [page 237]
When you import XML schema attributes as metadata, SAP Data Services maps the attributes to Data Services column attributes

XML schema error checking [page 238]
SAP Data Services enables you to control whether the software checks each incoming XML file or message for validity against the imported XML Schema.

2.2.33.1 XML schema editor

Open an XML schema editor by opening the schema in the object library.

The software stores XML schemas under the Nested Schemas node in the *Formats* tab of the object library.

The XML schema editor contains the related nested schema in the W3C XML schema format and an XML Formats tab.

The following is an example of a related nested schema like what you see when you open the XML schema editor.

Example

```
<?xml version="1.0"?>
<xs:schema targetNamespace="http://my-company.com/namespace"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Order">
   <xs:complexType>
    <xs:sequence>
        <xs:element name="OrderNo" type="xs:string" />
     <xs:element name="CustID" type="xs:string" />
     <xs:element name="ShipTo1" type="xs:string" />
     <xs:element maxOccurs="unbounded" name="LineItems">
       <xs:completType>
       <xs:sequence>
       <xs:element name="Item"    type="xs:string" />
       <xs:element name="ItemQty"   type="xs:string" />
       <xs:element name="ItemPrice" type="xs:string" />
         </xs:sequence>
       </xs:complexType>
     </xs:element>
    </xs:sequence>
   <xs:complexType>
  <xs:element>
</xs:schema>
```

The example-related nested schema contains the following information:

- The object name is **Order**.
- Columns at the top level are **OrderNo**, **CustID**, and **ShipTo1**.
- **LineItems** lists the nested columns down one level: **Item**, **ItemQty**, and **ItemPrice**.

The XML Format tab contains the following information:

XML Schema Format tab

Option	Description
Imported From	Specifies the full path to the XML Schema format file.
Root element name	Specifies the root element in XML schema format.
Namespace	Specifies the Namespace of the XML schema format.

Parent topic: XML schema [page 231]

Related Information

XML schema properties [page 233]
XML schema supported attributes [page 234]
XML Schema elements [page 235]
XML Schema attributes [page 237]
XML schema error checking [page 238]
Format XML documents

2.2.33.2 XML schema properties

View XML schema properties in the *Properties* dialog box.

The following table describes the XML schema properties.

XML schema property	Description
Name	The name of the format. This name appears in the object library under the Nested Schemas node of the *Formats* tab. The software displays the name for sources and targets that reference this format in data flows.
Description	Text that you enter to describe and document the XML Schema.
Imported from	The full path to the format. For example, `D:\data\test.xsd`.
Root element name	The name of the primary node you want to import. SAP Data Services imports elements of the format that only belong to this node or any related sub nodes.
Namespace	The Namespace URL of the root element. Namespace is optional.

Parent topic: XML schema [page 231]

Related Information

XML schema editor [page 232]
XML schema supported attributes [page 234]
XML Schema elements [page 235]
XML Schema attributes [page 237]
XML schema error checking [page 238]

2.2.33.3 XML schema supported attributes

XML schema attributes include column and nested table attributes.

The following table describes the XML schema column attributes.

XML schema column attributes

XML schema column attribute	Description
Required	Indicates whether you have to map this column. Value is Yes or 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	List of all possible values separated by pipes or vertical bars. For example: "Red | White | Blue Green | Magenta". The schema cuts the string off at 256 characters.
XML Type	Track whether the column was an element or attribute in the original XML Schema.
Length	Number of characters in this column.
Max Length	Maximum number of characters allowed in this column.
Min Length	Minimum number of characters allowed in this column.
Namespace	Column namespace.

XML schema nested table attributes

Nested table attribute	Description
Any One Column	Specifies how the software sets the value of Any One Column. • If choice, for example "white | black | almond", sets value to Yes. • If sequence, for example "first, last, street, city, state", sets value to No. • If the XML schema contains both choice and sequence, sets value to No.
Maximum Occurrence	Indicates maximum number of rows in the table. If maximum occurrence equals zero, the software indicates that the XML Schema structure is not valid.
Minimum Occurrence	Indicates minimum number of rows in the table.
Optional Table	Indicates whether the table is optional in an instance document. The software still accepts the document as input even when the table is not included in the instance document.

Parent topic: XML schema [page 231]

Related Information

XML schema editor [page 232]
XML schema properties [page 233]
XML Schema elements [page 235]
XML Schema attributes [page 237]
XML schema error checking [page 238]

2.2.33.4 XML Schema elements

SAP Data Services maps XML schema elements to attributes when they are imported as metadata.

The following table lists the XML schema element with the mapped attribute.

XML schema elements and mapped attributes

XML schema element	Attribute
All	All Elements occur in any order. For more information, see Choice.

XML schema element	Attribute
Choice	Any One Column If you specify the complex type for an element as Choice then the software creates an attribute called Any One Column and sets it to YES. If you define the complex type with sequence or several nesting levels that contain a mix of choice and sequence, then the software creates the Any One Column table attribute and sets it to NO. Data Services processes `<Sequence>`, `<choice>`, and `<all>` as follows: • `<Sequence>` becomes "Any One Column = NO". Attributes A, B, and C become columns A, B, and C. • `<Choice>` becomes "Any One Column = YES": Attributes A, B, and C become columns A or B or C. • `<all>` becomes "All": B, C, and A or any combination of the three.
Default	Default Value
Enumeration	Enumeration Maximum size is 256 characters. Therefore, the software cuts the string after 256 characters. So, you may not be able to view all of the enumerated values.
Fixed	Fixed Value
Length	Length
MinLength	Min Length
MaxLength	Max Length
MaxInclusive	Max Value Inclusive of
MinInclusive	Min Value Inclusive of
MinExclusive	Min Value Exclusive of
MaxExclusive	Max Value Exclusive of
MaxOccurs	Maximum Occurrence. Applies to tables only.
MinOccurs	Minimum Occurrence. Applies to tables only.
Name	Column name
Pattern	Pattern
TotalDigits and FractionDigits	None. The software handles digits as decimal data types.
Type	Saved as the Native Type attribute (string). The software also translates type element into a data type, usually `varchar`.

XML schema element	Attribute
Sequence	See Choice.

Parent topic: XML schema [page 231]

Related Information

XML schema editor [page 232]
XML schema properties [page 233]
XML schema supported attributes [page 234]
XML Schema attributes [page 237]
XML schema error checking [page 238]
Unsupported XML schema components

2.2.33.5 XML Schema attributes

When you import XML schema attributes as metadata, SAP Data Services maps the attributes to Data Services column attributes

The following table maps the XML schema attribute to the applicable column attribute.

XML schema attributes and mapped column attributes

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 software also translates the Type element into a data type, usually `varchar`.
Use	An XML Schema `<Use>` attribute with a value of OPTIONAL becomes the `<Required>` attribute with a value of NO. An XML Schema `<Use>` attribute with a value of REQUIRED becomes the `<Required>` attribute with a value of YES.

Parent topic: XML schema [page 231]

Related Information

XML schema editor [page 232]
XML schema properties [page 233]
XML schema supported attributes [page 234]
XML Schema elements [page 235]
XML schema error checking [page 238]
Unsupported XML schema components

2.2.33.6 XML schema error checking

SAP Data Services enables you to control whether the software checks each incoming XML file or message for validity against the imported XML Schema.

To control whether the software checks for validity, 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 an 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 example is true of both the source and target.

> **Example**
>
> You define an element in the XML schema with enumeration values of Black, White, StainlessSteel, and Almond. An error occurs when the mapping of the element from the XML document yields a value of Red. Whether *Enable Validation* is enabled or not, the software generates the XML with the value Red. However, if you enable validation, the software issues an error.

> **Example**
>
> An element whitespace attribute is set to `collapse`. Because the software does not support the whitespace attribute, it retains the data in sources or targets to respect the setting.

A good use for the validation is to enable it during your development phase. For example, validate all messages to test for error conditions with representative messages. Then during your production phase, you could set up 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, identify and test name conflicts 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.

Parent topic: XML schema [page 231]

Related Information

XML schema editor [page 232]
XML schema properties [page 233]
XML schema supported attributes [page 234]
XML Schema elements [page 235]
XML Schema attributes [page 237]

3 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 in Designer.

The transforms described in this section are available from the Local Object Library in Designer in the *Transforms* tab.

The transforms that you have available in the *Transforms* tab depend on your Data Services package. 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
- Data Quality
- Platform
- Text Data Processing

3.1 Transform reference

SAP Data Services has several groups of transform types, and each group contains individual transforms that provide you with the ability to transform your data in many ways.

Data Integrator transforms

Data Integrator transforms help ensure data integrity and maximize developer productivity for loading and updating in a data warehouse environment. The following table describes the Data Integrator transforms.

Transform	Description
Data_Transfer	Allows a data flow to split its processing into two sub data flows and push down resource-consuming operations to the database server.
Date_Generation	Generates a column filled with date values based on the start and end dates and increment that you provide.
Effective_Date	Generates an additional "effective to" column based on the primary key's "effective date."
Hierarchy_Flattening	Flattens hierarchical data into relational tables so that it can participate in a star schema. Hierarchy flattening can be both vertical and horizontal.
History_Preserving	Converts rows flagged as UPDATE to UPDATE plus INSERT, so that the original values are preserved in the target. You specify in which column to look for updated data.
Key_Generation	Generates new keys for source data, starting from a value based on existing keys in the table you specify.
Map_CDC_Operation	Sorts input data, maps output data, and resolves before- and after-images for UPDATE rows. While commonly used to support Oracle changed-data capture, this transform supports any data stream if its input requirements are met.
Pivot (Columns to Rows)	Rotates the values in specified columns to rows. (Also see Reverse Pivot.)
Reverse Pivot (Rows to Columns)	Rotates the values in specified rows to columns.
Table_Comparison	Compares two data sets and produces the difference between them as a data set with rows flagged as INSERT and UPDATE.
XML_Pipeline	Processes large XML inputs in small batches.

Data Quality transforms

The Data Quality transforms parse, standardize, correct, enrich, match, and consolidate your customer and operational information assets. The following table describes the 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.

Transform	Description
Country_ID	Parses input data and then identifies the country/region of destination for each record.
Data_Cleanse	Identifies and parses name, title, and firm data, phone numbers, Social Security numbers, dates, and e-mail addresses. It can assign gender, add prenames, generate Match standards, and convert input sources to a standard format. It can also parse and manipulate various forms of international data, as well as operational and product data.
DSF2 Walk Sequencer	Adds delivery sequence information to your data, which you can use with presorting software to qualify for walk-sequence discounts.
Geocoder	Identifies and appends geographic information to address data such as latitude and longitude.
Global_Address_Cleanse	Identifies, parses, validates, and corrects global address data. For example, 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).

Platform transforms

The Platform transforms enable you to perform general data movement such as merge rows from two or more sources, create SQL query operations, and mask personal data for anonymity.

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.

Transform	Description
Data_Mask	Uses data masking techniques to disguise or hide personal information contained in your databases. For example, bank account numbers, credit card numbers, and income. Data masking techniques include the following: • Character replacement • Number variance • Date variance • Pattern variance • Number generalization • Date generalization Data masking maintains data relevancy and relationships while keeping client information confidential and anonymous. Data masking also helps support your policies for business data protection.
Map_Operation	Modifies data based on current operation codes and mapping expressions. The operation codes can then be converted between data manipulation operations.
DQM_Microservices	Configures and executes a subset of DQM microservices services within Data Services.
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.
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.
Validation	Ensures that the data at any stage in the data flow meets your criteria. You can filter out or replace data that fails your criteria.

Text Data Processing transforms

The Text Data Processing group consists of the Entity_Extraction transform. It extracts information (entities and facts) from unstructured data and creates structured data that can be used by various business intelligence tools.

Parent topic: Transforms [page 240]

Related Information

Dynamic transform settings [page 244]
Embedded help for transform editors [page 249]
Data Integrator transforms [page 250]
Data Quality transforms [page 338]
Platform transforms [page 840]
Text Data Processing transforms [page 1025]

3.2 Dynamic transform settings

Dynamic transform settings allow you to change settings after the transform is initialized, without having to terminate and reinitialize the transform.

Pass each new setting through an input field to the transform. The transform gets an updated setting from the input field and adjusts to use the new setting, before processing the incoming record.

The settings that exist when you initialize the transform are considered the 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 transform processes the record with the default setting for that option. If the dynamic setting is invalid, the transform logs a warning and then uses the default settings.

The following table contains the transform name and the corresponding dynamic input fields.

Transform	Dynamic input field
Data Cleanse	`Option_Content_Domain_Sequence` Specify the domain using the ISO2 country code. Valid values are: AR, EN_US, IT, PT_PT CS, ES_ES, JA, RO DA, ES_MX, MS, RU DE, FR, NL, SK EN_AU, GLOBAL, NO, SV EN_GB, HU, PL, TR EN_IN, ID, PT_BR, ZH The GLOBAL domain is a special content domain, which contains all variations and their associated properties. If a variation is not associated with domain-specific information, the GLOBAL domain serves as the default domain. The GLOBAL domain is required for every content domain sequence. Be sure to add GLOBAL as the last domain in the sequence. The sequence input field of the content domain can hold more than one domain. If there is more than one domain, separate the domains with a pipe (|). For example, to specify the domain for Spain and then for Portugal, enter ES_ES|PT_PT|GLOBAL.

Transform	Dynamic input field
Data Cleanse	`Option_Country` Contains an ISO2 country code to help parse phone data and to determine the content domain sequence. Obtain ISO2 country code data in one of the following ways: • An input file that contains the codes. • An upstream transform like the Global Address Cleanse transform that you set to output ISO2 codes. For phone data parsing, the Data Cleanse transform uses the ISO2 country code to determine the country code when: • The `Option_Country` field contains the ISO2 country code. • The country is included in the cleansing package. Map the Global Address Cleanse field `ISO_Country_Code_2Char` to the `Option_Country` input field. To set the domain sequence, Data Cleanse uses the ISO2 code to determine content domain sequence when the `Option_Country` field contains an ISO2 code.

Transform	Dynamic input field

Data Cleanse	`Option_Language`
	
	`Option_Region`
	
	Use these two fields with the `Option_Country` field to determine the content domain and output format.
	
	There are a few countries where `Option_Language` and `Option_Region` data is helpful to determine the most appropriate content domain and output format. The two fields are applicable only to certain countries. For example, Switzerland, Belgium, or Canada can be used for `Option_Language` and `Option_Region`.
	
	The transform uses these fields only to determine the most appropriate content domain and output format
	
	Map the `Option_Country`, `Option_Language`, and `Option_Region` input fields from the following Global Address Cleanse output fields in this order:
	
	• `ISO_Country_Code_2Char`
	• `Language`
	• `Region1`
	
	These three dynamic Data Cleanse input fields work together to determine the `Option_Content_Domain_Sequence` and `Option_Output_Format`.
	
	For more information and examples, read about domains in the *Designer Guide*.

Data Cleanse	`Option_Output_Format` Specify the output format with the ISO2 country code.
Entity Extraction	`Language`
Geocoder	`Option_Distance_Unit`
Geocoder	`Option_Max_Records`
Geocoder	`Option_Radius`
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`

Transform	Dynamic input field
Global Address Cleanse	`Option_Standardization_Character_Width_Style`
Global Address Cleanse	`Option_Standardization_Directional_Style`
Global Address Cleanse	`Option_Standardization_Include_Locality_Addition`
Global Address Cleanse	`Option_Standardization_Locality_Name_Style`
Global Address Cleanse	`Option_Standardization_Output_Country_Language`
Global Address Cleanse	`Option_Standardization_Postal_Phrase_Style`
Global Address Cleanse	`Option_Standardization_Primary_Type_Style`
Global Address Cleanse	`Option_Standardization_Region_Style`
Global Address Cleanse	`Option_Standardization_Secondary_Description_Style`
Global Address Cleanse	`Option_Standardization_Secondary_Number_Style`
Match	`Option_Field_Algorithm_Geo_Proximity_<logical_name>_Max_Distance`

Parent topic: Transforms [page 240]

Related Information

Transform reference [page 240]
Embedded help for transform editors [page 249]
Data Integrator transforms [page 250]
Data Quality transforms [page 338]
Platform transforms [page 840]
Text Data Processing transforms [page 1025]

3.3 Embedded help for transform editors

SAP Data Services Designer contains embedded documentation that opens with the transform editor so you can learn about the transform options while configuring the transform.

The embedded help content changes based on the transform pane you select. When you select a different pane or a new option group, the help information updates to reflect your selection.

Additionally, while in one embedded help topic, you can navigate to other topics by using hyperlinks within the open topic, or by accessing the table of contents.

SAP Data Services has embedded help for the following transforms:

- Associate
- Common
- Country ID
- Data Cleanse
- Data Mask
- DQM Microservices
- DSF2 Walk Sequencer
- Extraction
- Geocoder
- Global Address Cleanse
- Global Suggestion List
- Match
- Match Wizard
- USA Regulatory Address Cleanse
- User-Defined

> **i Note**
>
> Because of the complexity of the Associate, Match, and User-Defined transforms, perform the following extra steps to access embedded help:
>
> 1. Select the transform icon in the data flow.
> 2. Select *Tools* > `<transform>` *Editor* >.

Parent topic: Transforms [page 240]

Related Information

Transform reference [page 240]
Dynamic transform settings [page 244]
Data Integrator transforms [page 250]
Data Quality transforms [page 338]

Platform transforms [page 840]
Text Data Processing transforms [page 1025]

3.4 Data Integrator transforms

To extract, transform, and load data, use a Data Integrator transform.

Data Integrator transforms ensure data integrity and maximize developer productivity for loading and updating data into data warehouse environments.

Data_Transfer transform [page 251]
: Use the Data_Transfer transform to push down certain operations to the database server for more efficient processing.

Date_Generation transform [page 264]
: Use the Date_Generation transform in a data flow to produce a series of dates incremented as you specify.

Effective_Date transform [page 267]
: Use the Effective_Date transform to calculate an "effective-to" value for data that contains an effective date.

Hierarchy_Flattening transform [page 274]
: The Hierarchy_Flattening transform takes a hierarchical input and outputs a description of the hierarchy in vertical or horizontal, "flattened" format.

History_Preserving [page 287]
: Use the History_Preserving transform to flag rows so that the software preserves the original values in the target.

Key_Generation transform [page 296]
: Use the Key_Generation transform to generate new keys for source data that starts from a value based on an existing key in the table.

Map_CDC_Operation transform [page 300]
: Use the Map_CDC_Operation transform to sort input data, map output data, and resolve before- and after-images for UPDATE rows.

Pivot transform [page 308]
: Use the Pivot transform to rotate the values in specified columns to specified rows.

Reverse Pivot (Rows to Columns) [page 315]
: Use the Reverse Pivot to rotate the values in specified rows to columns.

Table_Comparison transform [page 318]
: To compare two data sets and produce the difference between them as a data set, use the Table_Comparison transform .

XML_Pipeline transform [page 334]
: Use the XML_Pipeline transform to process large XML input sources in small batches.

Parent topic: Transforms [page 240]

Related Information

Transform reference [page 240]
Dynamic transform settings [page 244]
Embedded help for transform editors [page 249]
Data Quality transforms [page 338]
Platform transforms [page 840]
Text Data Processing transforms [page 1025]

3.4.1 Data_Transfer transform

Use the Data_Transfer transform to push down certain operations to the database server for more efficient processing.

The following table describes the characteristics of the Data_Transfer transform.

Transform characteristic	Description
	Data_Transfer icon
Use	Writes the data from a source or the output from another transform into a transfer object and then reads data from the transfer object. The transfer type can be a relational database table or file. Pushes down resource-consuming operations to the database server. Transfer type is a database table. Example resource-consuming operations are joins, GROUP BY, and sorts.
Data inputs	Input can be 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 cannot contain hierarchical (nested) data.

Transform characteristic	Description
Data outputs	Output can be 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 operation 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 secondary data flows and executes them serially. The secondary data flow names use the following format: `<dataflowname_n>` The variable `<n>` is the number of the secondary data flow.

Data_Transfer transform editor [page 252]
: Open the Data_Transfer transform icon in the data flow and specify options in the transform editor.

Using Data_Transfer in a data flow [page 262]
: Increase the efficiency of a Group By operation in a dataflow by adding a Data_Transfer transform.

Parent topic: Data Integrator transforms [page 250]

Related Information

Date_Generation transform [page 264]
Effective_Date transform [page 267]
Hierarchy_Flattening transform [page 274]
History_Preserving [page 287]
Key_Generation transform [page 296]
Map_CDC_Operation transform [page 300]
Pivot transform [page 308]
Reverse Pivot (Rows to Columns) [page 315]
Table_Comparison transform [page 318]
XML_Pipeline transform [page 334]
Run as separate process

3.4.1.1 Data_Transfer transform editor

Open the Data_Transfer transform icon in the data flow and specify options in the transform editor.

The transform editor shows the input file schema in the upper pane. The lower pane contains the *General* tab. When you select *Table* for the *Transfer type*, and select a table from the *Table name* browse menu, the following additional tabs appear:

- *Options*
- *Bulk Loader Options*
- *Pre-Load Commands*
- *Post-Load Commands*

The additional tabs contain options for how the transform pushes down the processes to the transfer object.

Parent topic: Data_Transfer transform [page 251]

Related Information

Using Data_Transfer in a data flow [page 262]

3.4.1.1.1 General tab option descriptions

The options that you complete in the *General* tab in the target editor are based on the transfer type.

Transfer types determine the options that appear in the *General* tab. Types include:

- File
- Table
- Automatic
- Table, relational

The following table indicates the transfer type for each option.

General tab option descriptions

Transfer type	Option	Description
File, Table, or Automatic	*Enable transfer*	Enables or disables the execution of the Data_Transfer transform. It is selected by default. Disable this transform to compare performance with transfer enabled and disabled. **i Note** When you run the job in debug mode, Data Services automatically disables all Data_Transfer transforms.
File	*File options: File name*	Name of the flat file that you want to use as transfer for sub data flows. The file does not need to exist.

Transfer type	Option	Description
File	*File options: Root directory*	The name of the root directory that contains the file to use for transfer. If your default Job Server and Designer reside on the same computer, you can use the browse button to find the *Root directory*. If your default Job Server does not reside on your local computer, enter the path to your *Root directory* manually. You can use a global variable or parameter for the pathname.
File, Table, or Automatic	*Join rank*	Indicates the rank of the output data set relative to other tables and files joined in a data flow. The software joins sources with higher join ranks before joining sources with lower join ranks. Join rank specified in the Query transform editor FROM tab overrides any join rank specified in a source. For new jobs, specify the join rank only in the Query transform editor. Must be a non-negative integer. Default value is 0.
Table - relational	*Table options: Array fetch size*	Indicates the number of rows retrieved in a single request to a source database. The default value is 1000. Higher numbers reduce requests, lower network traffic, and improve performance. The maximum value is 5000. This option is available for source tables from DB2, Informix, ODBC, Oracle, and SQL Server datastores. When retrieving a column with an Oracle long data type, Data Services automatically sets *Array fetch size* to 1 because Data Services can retrieve only one row at a time for Oracle long data types.

Transfer type	Option	Description
Table - relational	*Table options: Database type*	Determines the content for additional tabs on the Data_Transfer transform editor. Allows you to quickly set target option values in data flows. If your target datastore has multiple configurations, the target editor lists database types that you define for your datastore configuration. To add or remove items in this list, edit the datastore configuration information in the datastore editor. Data Services allows you to use target table editor option values from any datastore configuration based on the following rules: • When the datastore has one configuration: The software sets target table editor values to the default values for the configuration. • When the datastore has more than one configuration with different database types or versions: The software determines the initial values for the additional database types or versions from the *Use values from* option in *Create New Configuration* dialog box. The *Create New Configuration* dialog box is in the datastore editor. If you select *Restore values if they already exist* in *Create New Configuration*, the software uses values that were previously defined for the database type or version. **i Note** 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. Other-

Transfer type	Option	Description
		wise, it gets the values from the configuration you select from the *Use values from* option. **Example** Suppose that you set a configuration for Oracle 8i. When you edit the target table editor options, you change the *Rows Per Commit* default value of 1000 to 500. Later you add a Microsoft SQL Server 2000 database datastore configuration to your original datastore and set the *Use values from* option to Oracle 8i. The target table editor settings for SQL Server inherit the value 500 for *Rows per Commit* because 500 was the value set in the Oracle 8i configuration.
Table - relational	*Table options: Table name*	Name of the database table that you want to use as transfer for sub data flows. Specify the table name with the following format: `<datastorename.ownername.tablename or <datastorename.ownername.schemaname.tablename>` You can click the browse button (ellipses) to display your datastores. Select a table name from the list or type in the name of a new table.

Transfer type	Option	Description

File, Table, or Automatic	*Transfer type*	Choose one of the following transfer types to temporarily store the data of each sub data flow:
		• *Table:* Database table from an existing datastore. Specify the Table options in *Table name*, *Database type*, and *Array fetch size*.
		• *File:* A flat file. Specify the File options in *Root directory* and *File name*.
		• *Automatic:* The optimizer, which is the optimization application inside the engine, chooses the transfer type from either:
		  • Your datastores that have the *Enable automatic data transfer* checkbox selected.
		  • The pageable cache directory that you specify in the Server Manager.
		The optimizer chooses the transfer type and location that could provide the optimal performance, based on subsequent operations that the data flow contains.
		Example
		If an ORDER BY follows the Data_Transfer transform in a data flow, the optimizer might pick the database datastore that contains the data so that the ORDER BY can be pushed down to the database.
		If the data flow does not contain an ORDER BY, GROUP BY, DISTINCT, join, or any expression that can be pushed down, the optimizer chooses the pageable cache directory. If multiple files are available (one on each job server in a server group), the optimizer chooses the directory that is local to the data flow process.

3.4.1.1.2 Data_Transfer transform editor: Options tab

The options that you complete in the *Options* tab in the Data_Transfer transform editor are based on the table that you select in the *General* tab.

The *Options* tab is available only after you select *Table* for the *Transfer type* and a table for the *Table name* in the *General* tab.

The following table describes the Data_Transfer options in the *Options* tab.

Option tab descriptions

Option	Description
Rows per commit	Specifies the transaction size in number of rows. If set to 1000, Data Services sends a commit to the underlying database every 1000 rows. This option is not available for targets in real time jobs.
Enable partitioning	Available only if the transfer table is either physically partitioned or logically partitioned. Enables Data Services to use the partition information in this transfer table. • Selected: Data Services transfers data using the number of partitions in the table as the maximum number of parallel instances. • Not selected: Disables this feature. If you select *Enable partitioning*, you cannot select *Number of Loaders*.
Delete data before loading	Deletes the existing data in the table before loading. • Selected: Deletes table data before loading new data. This is the default. • Not selected: Appends data to existing data in the table.

Option	Description
Number of loaders	Specify the number of loaders. • Loading with one loader is called "single loader loading." • Loading with more than one loader is called "parallel loading." The default number of loaders is 1. If you select *Number of Loaders*, you cannot select *Enable partitioning*. When parallel loading, each loader receives the number of rows indicated in the Rows per commit option, in turn, and applies the rows in parallel with the other loaders. **Example** If you set *Rows per commit* to 1000 and the *Number of loaders* to 3, Data Services: • Sends the first 1000 rows to the first loader. • Sends the second 1000 rows to the second loader. • Sends the third 1000 rows to the third loader. • Sends the forth 1000 rows to the first loader, and so on.
Drop and re-create before loading	Drops the existing table and creates a new table with the same name before loading. • Selected: Drops the existing table and creates a new table with the same name before loading. This is the default. • Not selected: Does not drop and re-create the table before loading. **Note** Unlike a template table, you can use bulk loading options for a transfer table even when the *Drop and re-create before loading* option is selected.
Data definition language (DDL)	Either have Data Services generate a default DDL by selecting *Generate default DDL*, or type your own DDL. Optionally add extra parameters such as table space name or extent size. Data Services saves the DDL and uses it at job execution time.
Generate default DDL	Select to display the default SQL CREATE TABLE statement that Data Services generates.

3.4.1.1.3 Data_Transfer transform editor: Bulk Loader Options descriptions

Use the options in the *Bulk Loader Options* tab when bulk loading is applicable for the specific relational database management system.

There are some supported database types that do not support bulk loading. If the database that you selected in the *General* tab supports bulk loading, complete options in the *Bulk Loader Options* tab. Keep in mind that the settings in the Data_Transfer editor affect data transfer at the point that you placed the transform in your data flow.

The following table describes the bulk loading options in general.

Bulk loader option	Description
Bulk load	Specifies the number of rows SAP Data Services loads before a commit takes place. Enter a positive integer. If you leave this option blank, the load utility uses the default value at runtime. The default is 1000.
Mode	Specifies the mode for loading data in the target table. Available modes depend on the bulk load method that you select. • *Append*: Adds new records to the table. • *Truncate*: Deletes the existing records and inserts the loaded data.
Rows per commit	Specifies the number of rows to load before Data Services commits the transaction. Enter a positive integer. The default setting is 1000 rows.
Maximum rejects	Specifies the maximum number of warnings issued before Data Services stops bulk loading. Enter a positive integer or leave blank. If you enter 0 or do not specify a value, Data Services continues bulk loading regardless of the number of warnings issued. When Data Services exceeds the value for *Maximum rejects*, the bulk loader stops.
Network Packet Size (kb)	Specifies the network packet size in KB. Enter an integer. The default setting is based on the database type. 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.

To determine whether your database supports bulk loading, and for specific bulk loading settings, find your database settings under Types of target tables [page 167].

3.4.1.1.4 Data_Transform editor: Pre- and Post-Load Commands tabs

To specify SQL commands that SAP Data Services executes before starting a load or after finishing a load into a transfer table, set options in the *Pre-Load Commands* and *Post-Load Commands* tabs.

The Pre- and Post-Load Commands tab contains a SQL Commands text box. When Data Services calls a data flow, it opens all objects in the data flow. For example, Data Services opens queries, transforms, sources, and targets. Next, Data Services runs the target preload script. As a result, Data Services executes any preload SQL commands before processing any transform.

> **i Note**
>
> Because Data Services executes the SQL commands as a unit of transaction, do not include transaction commands in preload or postload SQL statements.

You cannot use preload and postload SQL commands in a real-time job.

To edit SQL command lines, select the line in *SQL Commands* to display the line in the *Value* box. Enter or edit the line text in the *Value* box. Click in the SQL Commands text box to add the text.

Editing pre- and post-load SQL commands [page 261]
To add pre- and post-load instructions to the Data_Transfer transform, enter SQL commands in the Pre- and Post-Load Commands tabs.

3.4.1.1.4.1 Editing pre- and post-load SQL commands

To add pre- and post-load instructions to the Data_Transfer transform, enter SQL commands in the Pre- and Post-Load Commands tabs.

You must select Table as the Transfer type in the *General* tab so the *Pre-Load Commands* and the *Post-Load Commands* tabs appear.

To add a line, delete an existing line, and include variables and parameters in the SQL Commands box in the *Pre-Load Commands* or *Post-Load Commands* tab, perform the following steps:

1. To add the first line and subsequent lines to the SQL Commands box:
   a. Right-click over the *SQL Commands* box and select *Insert After*.
   b. Type the first line of the SQL command in the *Value* text box.
   c. Click in the *SQL Commands* box to add the line.
   d. Select the line you just added, or the line before or after which to add a new line.
   e. Right-click and select *Insert Before* or *Insert After* as applicable.
   f. Type the SQL command in the *Value* text box.
   g. Click in the *SQL Commands* box to add the line.

2. To delete a line:
   a. Select the line in the *SQL Commands* box
   b. Right click, and choose *Delete*.
3. Include variables and parameters in preload or postload SQL statements:
   a. Click in the existing line in the *SQL Commands* box.

      The line appears in the *Value* box.
   b. Place your cursor in the line in the *Value* box where you want to add the variable or parameter and use either brackets, braces, or quotes as applicable.
   c. Click in the *SQL Commands* box to update the line with your changes.

   Data Services translates each statement differently, writing a statement that depends on the variable or parameter type.

Example

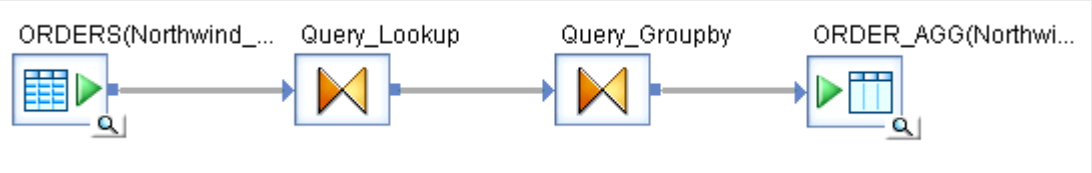
Entered statement	Variable value	Written statement
[$X]	5	5
[$X]	John Smith	John Smith
{$X}	5	5
{$X}	John Smith	John Smith
'$X'	5	5
'$X'	John Smith	John Smith

3.4.1.2 Using Data_Transfer in a data flow

Increase the efficiency of a Group By operation in a dataflow by adding a Data_Transfer transform.

Example

The following example shows a simple data flow that contains a Query transform. The Query transform looks up sales subtotals and groups the results by country and region.

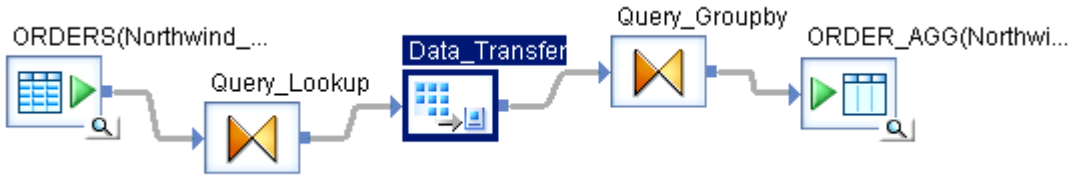


The GROUP BY operation processes millions of rows, which are slowing down processing. Data Services cannot push the GROUP BY operation down to the database because the Query transform that comes before the GROUP BY operation contains a lookup_ext function. Data Services cannot push down the lookup_ext function.

To make this data flow more efficient, add a Data_Transfer transform to split the lookup_ext function and the GROUP BY operation into two sub data flows. Then Data Services can push the GROUP BY operation to the target database.

The following steps show you how to restructure the example data flow with the Data_Transfer transform to speed up processing time:

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 place it between the two Query transforms. Reconnect the objects.



3. Open the Data_Transfer transform editor, the *General* tab, and select *Table* for *Transfer type*.

   The *Options*, *Bulk Loader Options*, *Pre-Load Commands*, and *Post-Load Commands* tabs appear.
4. In the *General* tab, Table options group, click the browse button (ellipses button) for *Table name*.

   The *Input table for Data_Transfer* dialog box opens.
5. Select an existing table or enter a name in *Table name* for a new table.

   This is the table that will contain the data to transfer. If you entered a new table name, Data Services creates the table.
6. Select *OK*.

   The *Input table for Data_Transfer* dialog box closes.
7. Accept the default setting for *Array fetch size* in the Table options group.
8. Complete any remaining settings for the job as applicable, save, and execute the job.

During job execution, Data Services displays messages for each sub data flow during execution.

Example

For the example 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.
```

Task overview: Data_Transfer transform [page 251]

Related Information

Data_Transfer transform editor [page 252]
Data_Transfer transform for push-down operations

3.4.2 Date_Generation transform

Use the Date_Generation transform in a data flow to produce a series of dates incremented as you specify.

Date_Generation information

Characteristic	Description
	Date_Generation icon
Use	Produces the key values for a time dimension target. From this generated sequence, populate other fields in the time dimension, such as day_of_week, using functions in a query.
Data inputs	N/A
Data outputs	A data set with a single column named `DI_GENERATED_DATE` that contains the date sequence. The software flags the generated rows as INSERT. The Date_Generation transform does not generate hierarchical data. Generated dates can range from 1900.01.01 through 9999.12.31.

Date_Generation transform editor options [page 265]
Set options in the Date_Generation transform for SAP Data Services to generate key values for a time dimension target.

Example: Create time dimension target [page 266]
An example showing how to use the Date_Generation transform to create a time dimension target.

Parent topic: Data Integrator transforms [page 250]

Related Information

Data_Transfer transform [page 251]
Effective_Date transform [page 267]
Hierarchy_Flattening transform [page 274]
History_Preserving [page 287]
Key_Generation transform [page 296]
Map_CDC_Operation transform [page 300]
Pivot transform [page 308]
Reverse Pivot (Rows to Columns) [page 315]
Table_Comparison transform [page 318]
XML_Pipeline transform [page 334]

3.4.2.1 Date_Generation transform editor options

Set options in the Date_Generation transform for SAP Data Services to generate key values for a time dimension target.

The following table describes the options to complete in the *Date Generation* tab of the Date_Generation transform editor.

Date_Generation option descriptions

Date_Generation option	Description
Start date	The first date in the output sequence. Specify this date using the following format: YYYY.MM.DD `YYYY` is a year value, `MM` is a month value, and `DD` is a day value.
End date	The last date in the output sequence. Use the same format as the *Start date* to specify this date.
Increment	The interval between dates in the output sequence. Select *Daily*, *Monthly*, or *Weekly*.

Date_Generation option	Description
Join rank	Indicates the rank of the output data set relative to other tables and files joined in a data flow. Must be a non-negative integer. The default is 0. Data Services joins sources with higher join ranks before joining sources with lower join ranks. Join rank specified in the Query transform editor *FROM* tab overrides any join rank specified in a source. For new jobs, specify the join rank in the Query transform editor only.
Cache	Indicates whether Data Services reads the required data from the source and loads it into memory or pageable cache. Because Data Services must read an inner source of a join for each row of an outer source, select to cache the source when it is an inner source in a join. • *Yes*: The source is always cached unless it is the outermost source in a join. Yes is the default setting. • *No*: The source is never cached. Cache specified in the Query transform editor *FROM* tab overrides any cache specified in a source. For new jobs, specify the cache in the Query transform editor only.

Parent topic: Date_Generation transform [page 264]

Related Information

Example: Create time dimension target [page 266]
Date_Generation transform [page 264]

3.4.2.2 Example: Create time dimension target

An example showing how to use the Date_Generation transform to create a time dimension target.

Example

Follow these steps to create a time dimension target with dates from the beginning of the year 1997 to the end of the year 2000:

1. Place a Date_Generation transform, a query, and a target in a data flow.
2. Connect the output of the Date_Generation transform to the query, and the output of the query to the target.

3. Specify the following values in the Date_Generation transform:
   - *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.)

   You can also specify a variable for these options.
4. 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)

Parent topic: Date_Generation transform [page 264]

Related Information

Date_Generation transform editor options [page 265]
Date_Generation transform [page 264]

3.4.3 Effective_Date transform

Use the Effective_Date transform to calculate an "effective-to" value for data that contains an effective date.

The following table describes the characteristics of the Effective_Date transform.

Characteristic	Description
	Effective_Date icon
Use	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. The queries based on the effective dates produce meaningful results.
Data inputs	Data that has an effective date column. Effective dates allow you to indicate changes to information over time.

Characteristic	Description
Data outputs	Includes all of the columns from the source schema and the calculated effective-to date column.

Effective_Date transform editor [page 268]
When you include the Effective_Date transform in a data flow, complete the applicable options in the transform editor.

Effective_Date transform editor options [page 269]
To calculate an effective-to value for data with an effective date, complete the options in the Effective_Date transform editor.

Example: Input data for Effective_Date transform [page 271]
Effective dates allow you to indicate changes to information over time.

Example: Output data for Effective_Date transform [page 272]
The output from the Effective_Date transform includes all of the columns from the source schema and the calculated effective-to date column.

Parent topic: Data Integrator transforms [page 250]

Related Information

Data_Transfer transform [page 251]
Date_Generation transform [page 264]
Hierarchy_Flattening transform [page 274]
History_Preserving [page 287]
Key_Generation transform [page 296]
Map_CDC_Operation transform [page 300]
Pivot transform [page 308]
Reverse Pivot (Rows to Columns) [page 315]
Table_Comparison transform [page 318]
XML_Pipeline transform [page 334]

3.4.3.1 Effective_Date transform editor

When you include the Effective_Date transform in a data flow, complete the applicable options in the transform editor.

Open the Effective_Date transform after you configure it as part of a data flow. The Effective_Date transform editor includes the following:

- A *Schema In* pane on the upper left that shows the source schema

- A *Schema Out* pane on the upper right that shows the target schema
- An *Effective Date* tab with the transform options.

The software generates the target schema based on the values that you choose in the transform options. To refresh the target schema after you change the options, choose *View* > *Refresh* or press *F5*.

Parent topic: Effective_Date transform [page 267]

Related Information

Effective_Date transform editor options [page 269]
Example: Input data for Effective_Date transform [page 271]
Example: Output data for Effective_Date transform [page 272]

3.4.3.2 Effective_Date transform editor options

To calculate an effective-to value for data with an effective date, complete the options in the Effective_Date transform editor.

The following table contains the option descriptions for the *Effective Date* tab in the Effective_Date transform editor.

Effective_Date option	Description
Effective date column	Lists a column from the input data set with the data type of `date`. The column contains the effective date. SAP Data Services enters the column name automatically when the input data has a column named `EFFDT` in the source. The column appears in the output schema with the name `EFFDT`. Alternately, drag the applicable column from the *Schema In* pane to the *Effective date* column text box, or select a column name from the dropdown list. This field is required.

Effective_Date option	Description
Effective sequence column	Lists a column from the input data set that indicates the order in time of related rows that have duplicate effective dates. Alternately, drag the applicable column from the *Schema In* pane to the *Effective sequence column* text box, or select a column name from the dropdown list. If there is no related row that shares effective dates, the sequence numbers are the same, for example, 0. If related rows share the same effective dates, Data Services increments the sequence numbers as it adds rows with conflicting effective dates. The transform returns only the row containing the maximum sequence number if there are related rows with the same effective date. This field is required only if the input data set allows duplicate effective dates.
Effective to column	Lists the output column added to the output schema that contains the effective-to date, with a `date` data type. 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, Data Services uses the *Default effective to date* value.
Default effective to date value	A date assigned as the effective-to date for rows with the highest effective date among related rows. For convenience, specify a variable for this option.

Parent topic: Effective_Date transform [page 267]

Related Information

Effective_Date transform editor [page 268]
Example: Input data for Effective_Date transform [page 271]
Example: Output data for Effective_Date transform [page 272]

3.4.3.3 Example: Input data for Effective_Date transform

Effective dates allow you to indicate changes to information over time.

Input data for an Effective_Date transform has an effective date column. 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. The effective date distinguishes each row that describes the set of information.

Example

An input data set includes the following columns:

- **Project:** Name that identifies the information being described.
- **Effective date:** The date from which the data in the row is valid.
- **Status:** Data that changes over time.

The following table contains an example portion of the input data set.

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

The 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, described by the values in the **Project** column.

In the following table, an additional **Effective sequence** column contains a value that helps distinguish between rows with duplicate effective dates. The Hetch Hetchy Reservoir project contains a row with an **Effective sequence** of 0 and a row with an **Effective sequence** of 1. The different **Effective sequence** values indicate that the two rows have the same **Effective date** of 2002.10.17.

Project	Effective date	Effective sequence	Status
Cherry Lake	1999.06.22	0	Proposal
Cherry Lake	2002.01.12	0	Case
Hetch Hetchy Reservoir	1999.08.02	0	Proposal
Hetch Hetchy Reservoir	2002.10.17	0[a]	Proposal
Hetch Hetchy Reservoir	2002.10.17	1[a]	Case
Hetch Hetchy Reservoir	2003.05.06	0	Case

Project	Effective date	Effective sequence	Status

[a] The effective sequence distinguishes the project status for two related rows with the same Effective date.

Only the row with the largest sequence number of 1 is effective-dated by the transform.

If you construct a query that selects the status of the "Hetch Hetchy Reservoir" project on "2002.12.31", the query returns the status from the last row, which is "Case". The last row has an effective date of 2003.05.06. The **Effective date** for this row indicates a date that falls after the stated date of 2002.12.31. The other Hetch Hetchy Reservoir rows contain an **Effective date** that falls before the stated date.

Considerations

The input data set can contain only rows flagged as NORMAL. The input data set can contain hierarchical data. However, the transform operates only on the rows at the top level of the input data set. The transform passes nested data through to the output without change. Therefore, columns that contain nested schemas cannot be used as transform parameters.

Parent topic: Effective_Date transform [page 267]

Related Information

Effective_Date transform editor [page 268]
Effective_Date transform editor options [page 269]
Example: Output data for Effective_Date transform [page 272]

3.4.3.4 Example: Output data for Effective_Date transform

The output from the Effective_Date transform includes all of the columns from the source schema and the calculated effective-to date column.

Example

Given a default "effective-to" date of January 1, 2999, SAP Data Services transforms the input described in the data input section as follows:

Project	Effective date	Effective-to date	Status
Cherry Lake	1999.06.22	2003.01.12	Proposal
Cherry Lake	2003.01.12	2999.01.01[a]	Case
Hetch Hetchy Reservoir	1999.08.02	2002.10.17	Proposal

Project	Effective date	Effective-to date	Status
Hetch Hetchy Reservoir	2002.10.17	2003.05.06	Case
Hetch Hetchy Reservoir	2003.05.06	2999.01.01[a]	Case

[a] The default effective-to date closes the effective date range.

An *Effective sequence column* is necessary to produce a unique key for related rows that contain the same Effective date. Therefore, the output from the Effective_Date transform includes a single row where the input had more than one row with the same Effective date. The row returned contains the largest Effective sequence number of 1:

Project	Effective date	Effective to date	Effective sequence	Status
Cherry Lake	1999.06.22	2003.01.12	0	Proposal
Cherry Lake	2003.01.12	2999.01.01	0	Case
Hetch Hetchy Reservoir	1999.08.02	2002.10.17	0	Proposal
Hetch Hetchy Reservoir	2002.10.17	2003.05.06	1[a]	Case
Hetch Hetchy Reservoir	2003.05.06	2999.01.01	0	Case

[a] The transform in this example omits data from the row with the same Effective date and with the Effective sequence = 0 from the output.

After Data Services generates a range of effective dates for a set of data, use the **Effective to date** value to filter appropriate records. For example, extract the subset of records valid as of today by selecting only those records whose **Effective to date** column is later than the current date and **Effective date** column is earlier than the current date.

The Effective_Date transform passes through nested schemas in the input without change.

Parent topic: Effective_Date transform [page 267]

Related Information

Effective_Date transform editor [page 268]
Effective_Date transform editor options [page 269]
Example: Input data for Effective_Date transform [page 271]

3.4.4 Hierarchy_Flattening transform

The Hierarchy_Flattening transform takes a hierarchical input and outputs a description of the hierarchy in vertical or horizontal, “flattened” format.

The following table describes the characteristics of the Hierarchy_Flattening transform.

Characteristic	Description
	Hierarchy_Flattening icon
Use	Constructs a complete hierarchy from parent/child relationships, then produces a description of the hierarchy in vertically or horizontally flattened format.
Data inputs	Rows that describe individual parent-child relationships.
Data outputs	Columns that represent each level in the hierarchy. The root is listed in the first column and the outermost leaf is listed in the last column.

Hierarchy_Flattening transform editor [page 275]
When you include the Hierarchy_Flattening transform in a data flow, you configure the options in the editor.

Hierarchy_Flattening transform editor options [page 276]
To describe input parent child structure and the flattening type to use, specify options in the *Hierarchy Flattening* tab.

Hierarchy_Flattening transform cyclic run-time error [page 280]
If the hierarchy represented by the input data set is cyclic, SAP Data Services produces a run-time error.

Hierarchy_Flattening transform data inputs [page 281]
Data input for the Hierarchy_Flattening transform includes rows that describe individual parent to child relationships.

Hierarchy_Flattening data output: Horizontal flattening [page 282]
Use horizontal flattening so that each row of the output describes a single node in the hierarchy and the path to that node from the root.

Hierarchy_Flattening data outputs: Vertical flattening [page 284]
Use vertical flattening so that each row of the output describes a single node in the hierarchy and the path to that node from the root.

Parent topic: Data Integrator transforms [page 250]

Related Information

Data_Transfer transform [page 251]
Date_Generation transform [page 264]
Effective_Date transform [page 267]
History_Preserving [page 287]
Key_Generation transform [page 296]
Map_CDC_Operation transform [page 300]
Pivot transform [page 308]
Reverse Pivot (Rows to Columns) [page 315]
Table_Comparison transform [page 318]
XML_Pipeline transform [page 334]

3.4.4.1 Hierarchy_Flattening transform editor

When you include the Hierarchy_Flattening transform in a data flow, you configure the options in the editor.

The Hierarchy_Flattening transform editor includes the following sections:

- *Schema In* pane contains the source schema
- *Schema Out* pane contains the target schema
- Hierarchy_Flattening transform options
  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 based on the values that you choose in the transform options.
  To refresh the target schema after you change the options, choose *View* > *Refresh* or press *F5*.

Parent topic: Hierarchy_Flattening transform [page 274]

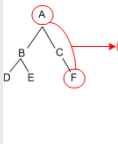
Related Information

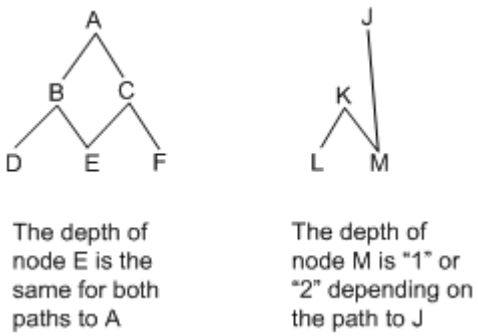
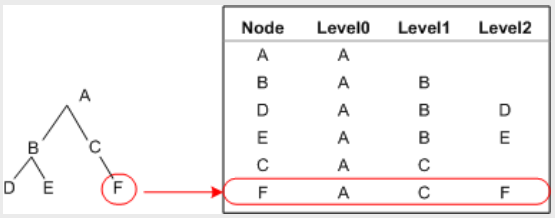
Hierarchy_Flattening transform editor options [page 276]
Hierarchy_Flattening transform cyclic run-time error [page 280]
Hierarchy_Flattening transform data inputs [page 281]
Hierarchy_Flattening data output: Horizontal flattening [page 282]
Hierarchy_Flattening data outputs: Vertical flattening [page 284]

3.4.4.2 Hierarchy_Flattening transform editor options

To describe input parent child structure and the flattening type to use, specify options in the *Hierarchy Flattening* tab.

The following table describes the options in the *Hierarchy Flattening* tab in the transform editor.

Option	Description
Flattening type group:	
Vertical	Describes the hierarchical relationship in the output in a vertical manner. Each row of the output: • Describes a single relationship between ancestor and descendent • Describes 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. Example  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

Option	Description
Use maximum length paths	Applicable when you select Vertical for flattening. Indicates whether the longest or the 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. 
Horizontal	Describes the hierarchical relationship in the output in a horizontal manner. Each row of the output: • Describes a single node in the hierarchy • Describes the path to that node from the root The horizontal mode requires that you specify the maximum path length through the tree as the *Maximum depth*. Example  Node Level0 Level1 Level2 A A B A B D A B D E A B E C A C F A C F

Option	Description
Maximum depth	For horizontal flattening only. Indicates the maximum depth of the hierarchy. The root node (level 0) has a depth of 0; the first level has a depth of 1, and so on. If you do not know the number of levels in your hierarchy, set *Maximum depth* to 1. When you execute the job, SAP Data Services issues a warning in the execution log indicating that the *Maximum depth* is less than the actual depth of the hierarchy. Reset *Maximum depth* to the actual value reported in the warning message.
Parent—Child groups:	
Parent column	Identifies the column in the source data that contains the parent identifier in each parent-child relationship. Drag this column from the source schema into the *Parent column* box. You cannot specify columns that include nested schemas as the parent attribute.
Parent attribute list	Identifies a column or columns that are associated with the parent column. Drag columns from the source schema into the *Parent attribute list*. The column name appears in the target schema with a prefix that identifies the column as a parent attribute. The *Parent attribute* list can contain columns that include nested schemas as the parent attribute. **Example** The following table shows the result of adding a column named `POPULATION` to the parent attribute list. **Flattening mode** / **Source column** / **Target column** Vertical / POPULATION / P_POPULATION Horizontal / POPULATION / P_L1_POPULATION, P_L2_POPULATION There is one column for each hierarchy level.

Option	Description
Child column	Identifies the column in the source data that contains the child identifier in each parent-child relationship. Drag this column from the source schema into the *Child column* box. You cannot specify columns that include nested schemas as the child.
Child attribute list	Identifies a column or columns that are associated with the child column. Drag columns from the source schema into the *Child attribute list*. The column name appears in the target schema with a prefix that identifies the column as a child attribute. Specify columns, including nested schemas, as the child attribute. **Example** The following table shows the result of adding a column named `POPULATION` to the child attribute list for vertical and horizontal flattening. **Flattening mode** / **Source column** / **Target column** Vertical / POPULATION / C_POPULATION Horizontal / POPULATION / C_L1_POPULATION C_L2_POPULATION There is one column for each hierarchy level.
Run as separate process	Creates a separate data flow process for the Hierarchy_Flattening transform when Data Services executes the data flow. Select to enable. The default setting is not selected.
Do not abort in case of cycle	Specifies not to abort a job if a cycle (circular dependency) is detected. If a cycle is detected, Data Services writes warnings to the log file. If not selected, Data Services aborts jobs when the transform encounters a cycle. The default setting is not selected.

Option	Description
Generate cycle rows	When a cycle is encountered, the circular node is the last in the hierarchy tree, and the tree itself carries a negative value for Leaf Level (Horizontal Flattening) and Depth (Vertical Flattening). Select to get more information about which nodes are causing the cycles. The default setting is not selected. **Example** Insert a Validation transform after the Hierarchy_Flattening transform to check for negative values. If Data Services encounters negative values, it sends the data to another path for further analysis.

Parent topic: Hierarchy_Flattening transform [page 274]

Related Information

Hierarchy_Flattening transform editor [page 275]
Hierarchy_Flattening transform cyclic run-time error [page 280]
Hierarchy_Flattening transform data inputs [page 281]
Hierarchy_Flattening data output: Horizontal flattening [page 282]
Hierarchy_Flattening data outputs: Vertical flattening [page 284]
Run as separate process

3.4.4.3 Hierarchy_Flattening transform cyclic run-time error

If the hierarchy represented by the input data set is cyclic, SAP Data Services produces a run-time error.

A cyclic error occurs when a hierarchy in the input data set has a node that is its own ancestor.

Data Services does not produce the run-time error under the following circumstances:

- If you select the *Do not abort in case of cycle* option
- If the input data source describes multiple root nodes

Parent topic: Hierarchy_Flattening transform [page 274]

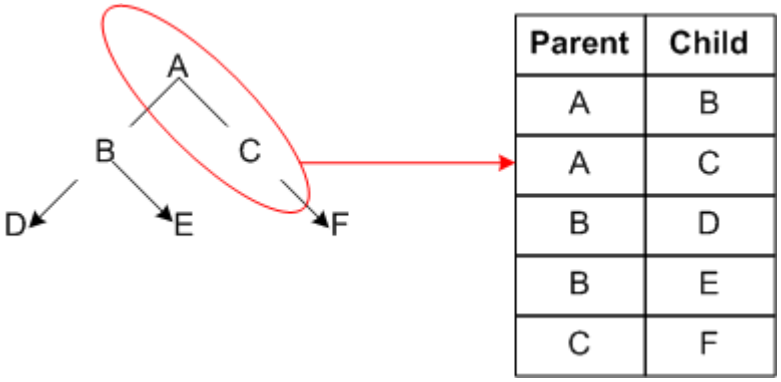
Related Information

Hierarchy_Flattening transform editor [page 275]
Hierarchy_Flattening transform editor options [page 276]
Hierarchy_Flattening transform data inputs [page 281]
Hierarchy_Flattening data output: Horizontal flattening [page 282]
Hierarchy_Flattening data outputs: Vertical flattening [page 284]

3.4.4.4 Hierarchy_Flattening transform data inputs

Data input for the Hierarchy_Flattening transform includes rows that describe individual parent to child relationships.

The following illustration shows rows in an input table that describe 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 that describe the parent and, or child.

Example

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 manager (parent node). The following table shows the columns in the input data set.

Employee	Node/Key
Employ_ID	Child node Primary key
Dept	
Salary	
Mgr_ID	Parent node

The following lists other characteristics of an input data set:

- The input data set includes only rows with operation code of NORMAL.
- The input data set can contain hierarchical data.

Parent topic: Hierarchy_Flattening transform [page 274]

Related Information

Hierarchy_Flattening transform editor [page 275]
Hierarchy_Flattening transform editor options [page 276]
Hierarchy_Flattening transform cyclic run-time error [page 280]
Hierarchy_Flattening data output: Horizontal flattening [page 282]
Hierarchy_Flattening data outputs: Vertical flattening [page 284]

3.4.4.5 Hierarchy_Flattening data output: Horizontal flattening

Use horizontal flattening so that each row of the output describes a single node in the hierarchy and the path to that node from the root.

Horizontal flattening shows the number of levels in the hierarchy and the distance between a give node and the root node. With horizontal flattening, the root is listed in the first column and the outermost leaf is listed in the last column.

The following table describes the target columns that horizontal flattening produces.

Column name	Description
CURRENT_LEAF	The end node described.
LEAF_LEVEL	The number of levels down from the root node where Current_leaf is found The root node has Leaf_level of 0.
LEVEL0	The descriptor for the top level node.
LEVEL1	The descriptor for the first level node If Leaf_level is 0, this value is NULL.
LEVEL <n>	The descriptor for the <n>th level node where <n> is the number of levels in the hierarchy If Leaf_level is <n> -1 or less, this value is NULL.

Column name	Description
P_L0_ <attribute_column>	Parent attribute column associated with the node described in Level0.
C_L1_ <attribute_column>	Child 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>	Parent 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>	Child 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>	Parent 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>	Child 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.

Parent topic: Hierarchy_Flattening transform [page 274]

Related Information

Hierarchy_Flattening transform editor [page 275]
Hierarchy_Flattening transform editor options [page 276]
Hierarchy_Flattening transform cyclic run-time error [page 280]
Hierarchy_Flattening transform data inputs [page 281]
Hierarchy_Flattening data outputs: Vertical flattening [page 284]

3.4.4.5.1 Example of horizontal flattening

Use this example to help you understand horizontal flattening.

Example

The following table shows an example of the target schema and data for horizontal flattening. The target schema contains the following:

- Two level hierarchy
- Country as the root node
- State at level 1 (leaf node)
- City at level 2 (leaf node)

The parent and child attributes are both 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 that includes a Hierarchy_Flattening transform with the same attribute for parent and child nodes, you may follow the transform with a query that filters the duplicated attribute values. The query passes the P_L0_ `<attribute_column>` and the C_L `<n>` _ `<attribute_column>` through as they are. The query keeps either parent or child attribute for the intermediate levels.

3.4.4.6 Hierarchy_Flattening data outputs: Vertical flattening

Use vertical flattening so that each row of the output describes a single node in the hierarchy and the path to that node from the root.

The following table shows the target columns that vertical flattening produces.

Column name	Description
ANCESTOR	The node closer to the root node in the relationship described by this row.
DESCENDENT	The node farther from the root node in the relationship described by this row.
DEPTH	Number of levels between the Ancestor and Descendent.
ROOT_FLAG	Identifies the value in the Ancestor column as the top node of the hierarchy. Root_flag is 1 if Ancestor is the root node. Otherwise, Root_flag is 0.
LEAF_FLAG	Identifies the value in the Descendent column as the bottom node of the hierarchy. Leaf_flag is 1 if Descendent is the leaf node. Otherwise, Leaf_flag is 0.
P_ `<attribute_column>`	Column from the source that you associate with the parent (can be more than one P_ `<attribute_column>`). This value is NULL for a row describing the relationship between a leaf node and itself, which means that the Leaf_flag is 1, and the parent and child are the same node.
C_ `<attribute_column>`	Column from the source that you associate with the child (can be more than one C_ `<attribute_column>`). This value is NULL for a row describing the relationship between a root node and itself, which means that the Root_flag is 1, and the parent and child are the same node.

Parent topic: Hierarchy_Flattening transform [page 274]

Related Information

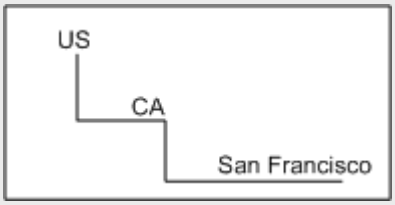
Hierarchy_Flattening transform editor [page 275]
Hierarchy_Flattening transform editor options [page 276]
Hierarchy_Flattening transform cyclic run-time error [page 280]
Hierarchy_Flattening transform data inputs [page 281]
Hierarchy_Flattening data output: Horizontal flattening [page 282]

3.4.4.6.1 Example of vertical flattening

Use this example to help you understand vertical flattening.

Example

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 both a population value.

ANCESTOR	DESCENDENT	DEPTH	ROOT _FLAG	LEAF _FLAG	P_POP	C_POP
US	CA	1	1	0	272,583,805	30,866,851
US	SanFrancisco	2	1	1	272,583,805	723,959
CA	SanFrancisco	1	0	1	30,866,851	723,959
US	US	0	1	0	272,583,805	NULL
CA	CA	0	0	0	30,866,851	30,866,851
SanFrancisco	SanFrancisco	0	0	1	NULL	723,959

Each node is listed one time as both ancestor and descendent. The Parent attribute is null for a row describing the relationship between a leaf node and itself. Likewise the Child attribute is null for a row describing the relationship between a root node and itself.

The transform ignores any hierarchical data unless a nested schema is identified as a parent or child attribute. An attribute column containing nested data is passed through the transform without change.

3.4.5 History_Preserving

Use the History_Preserving transform to flag rows so that the software preserves the original values in the target.

History_Preserving information

Characteristic	Description
	History_Preserving icon
Use	Produces a new row in your target instead of updating an existing row. You indicate in which columns the transform identifies changes to be preserved. If the value of certain columns change, the transform creates a new row for each row flagged as UPDATE in the input data set.
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.
Data outputs	A data set with rows flagged as INSERT or UPDATE.

History_Preserving transform editor [page 288]
When you include the History_Preserving transform in a data flow, complete the applicable options in the editor.

History_Preserving transform editor options [page 289]
Specify options in the History_Preserving section to set the software behavior for this transform.

Example: History_Preserving data inputs [page 292]
Data that is input to the History_Preserving transform contains the result of a comparison of two images of the same data where rows are flagged as UPDATE or INSERT.

Example: History_Preserving data outputs [page 293]
Output from the History_Preserving transform contains a data set with rows flagged as INSERT or UPDATE.

Parent topic: Data Integrator transforms [page 250]

Related Information

Data_Transfer transform [page 251]
Date_Generation transform [page 264]

Effective_Date transform [page 267]
Hierarchy_Flattening transform [page 274]
Key_Generation transform [page 296]
Map_CDC_Operation transform [page 300]
Pivot transform [page 308]
Reverse Pivot (Rows to Columns) [page 315]
Table_Comparison transform [page 318]
XML_Pipeline transform [page 334]

3.4.5.1 History_Preserving transform editor

When you include the History_Preserving transform in a data flow, complete the applicable options in the editor.

The History_Preserving transform editor includes the *Schema In* pane, the *Schema Out* pane, and transform options section.

Drag column names from the *Schema In* pane to fill values for the following columns:

- Valid from
- Valid to
- Compare columns
- 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 change the options, choose *View* > *Refresh* or press *F5*.

Parent topic: History_Preserving [page 287]

Related Information

History_Preserving transform editor options [page 289]
Example: History_Preserving data inputs [page 292]
Example: History_Preserving data outputs [page 293]

3.4.5.2 History_Preserving transform editor options

Specify options in the History_Preserving section to set the software behavior for this transform.

History_Preserving option descriptions

History_Preserving option	Description
Compare columns	Specifies the column or columns in the input data set for which this transform compares the before and after images to determine if there are changes. • If the values in the specified columns in each image of the data match, the transform flags the row as UPDATE. The result updates the warehouse row with values from the new row. The row from the before image is included in the output as UPDATE to effectively update the date and flag information. • If the values in the specified columns 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. The *Compare columns* cannot contain nested schemas.
Current flag—Column	Specifies a column from the source schema that identifies the current valid row from a set of rows with the same primary key. Use this flag to indicate whether a row is the most current data in the warehouse for a given primary key. The *Current flag—Column* value cannot be the same as the *Date columns—Valid from* option or the *Date columns—Valid to* option. The *Current flag—Column* option cannot contain a nested schema. If you specify any one of the above three options, Data Services validates that you specify a *Current flag—Column* value, *Current flag—Set value* value, and *Current flag—Reset value* value.
Current flag—Reset value	Specifies an expression that evaluates to a value with the same data type as the *Current flag—Column* column. Data Services uses this value to update the *Current flag—Column* in an existing row in the warehouse that includes changes in one or more of the compare columns. Enter a value for this option to enable the *Current flag—reset value* option. For added flexibility, you can enter a variable for this option.

History_Preserving option	Description
Current flag—Set value	Specifies an expression that evaluates to a value with the same data type as the *Current flag—Column* column. Data Services uses this value to update the *Current flag—Column* in the new row of the warehouse added to preserve history of an existing row. Enter a value in the *Current flag—Column* box to enable this option. For added flexibility, you can enter a variable for this option.
Date columns—Valid from	Specifies a date or datetime column from the source schema. If the warehouse uses an effective date to track changes in data, specify a *Date columns—Valid from* 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 *Date columns—Valid to* date column in the previously current row in the warehouse. Data Services validates that you have specified a value for both the *Date columns—Valid from* and the *Date columns—Valid to* options.
Date columns—Valid to	Specifies a date or datetime column from the source schema. Specify if the warehouse uses an effective date to track changes in data and if you specified a *Date columns—Valid from* value. This value is used as the new value in the *Date columns—Valid to* column in the new row added to the warehouse to preserve history of an existing row. The *Date columns—Valid to* value cannot be the same as the *Date columns—Valid from* value.

History_Preserving option	Description

Date columns—Valid to date value	Specify the values to use in the *Date columns—Valid to* option in the old record and the new record added to the warehouse to preserve history of an existing row.
	• *New record*: Specify one of the following values:
	  • A date value specified as yyyy.mm.dd. The default value is `9000.12.31`.
	  • A variable that contains a date value.
	• *Old record*: Specify one of the following values:
	  • Use the "valid from" date of new record.
	  Example
	  The following example shows that the new record, Key 2 *From_Date* contains `2006.01.31` and the old record, Key 1 *To_Date* contains the same value.
	  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
	  • Use a date that is one day before "valid from" date of new record.
	  Example
	  The following example shows that the new record, Key 2 *From_Date* contains `2006.01.31` and the old record, Key1 *To_Date* contains a date that is one day before that value, `2006.01.30`.
	  Key / **Empno** / **Name** / **Salary** / **From_Date** / **To_Date**
	  1 / 100 / Chang / 10000.00 / 2006.01.01 / 2006.01.30

History_Preserving option	Description
	<table><tr><th>Key</th><th>Empno</th><th>Name</th><th>Salary</th><th>From_Date</th><th>To_Date</th></tr><tr><td>2</td><td>100</td><td>Chang</td><td>20000.00</td><td>2006.01.31</td><td>9000.12.31</td></tr></table>
Preserve delete row(s) as update row(s)	Converts DELETE rows to UPDATE rows in the target warehouse. Also, if you set effective date values in *Date columns—Valid from* and *Date columns—Valid to*, sets the *Date columns—Valid to* value to the jog execution date. • *Yes*: Enables this behavior. • *No*: Does not enable this behavior. Use this option to maintain slowly changing dimensions by feeding a complete data set first through the Table Comparison transform with its *Detect deleted row(s) from comparison table* option selected.

Parent topic: History_Preserving [page 287]

Related Information

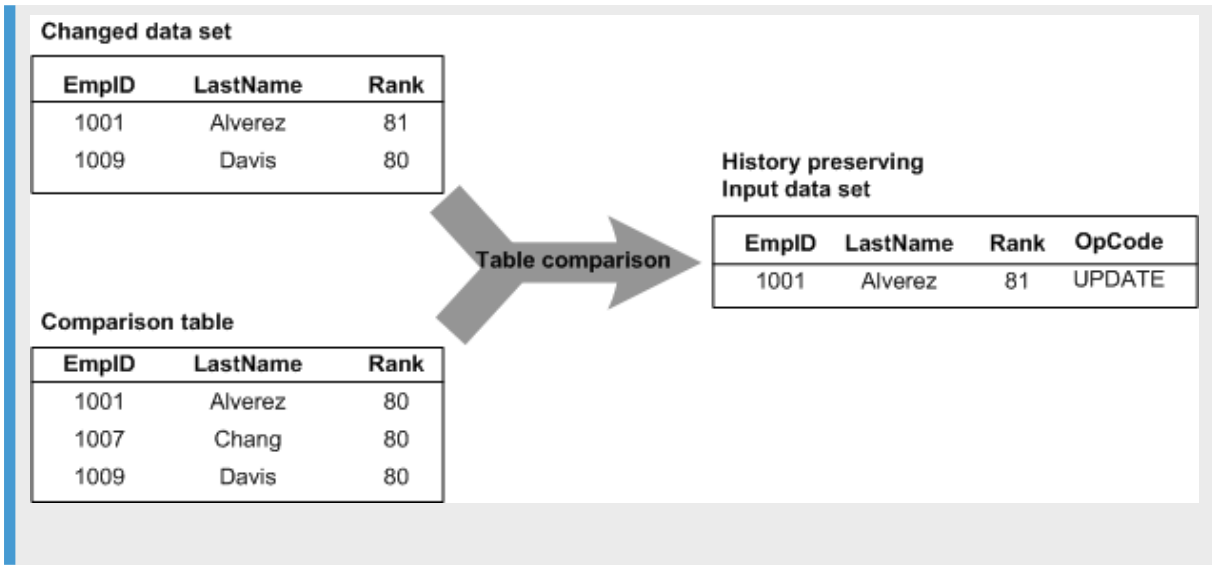
History_Preserving transform editor [page 288]
Example: History_Preserving data inputs [page 292]
Example: History_Preserving data outputs [page 293]

3.4.5.3 Example: History_Preserving data inputs

Data that is input to the History_Preserving transform contains the result of a comparison of two images of the same data where rows are flagged as UPDATE or INSERT.

Rows that contain changed data from the newer image of the data are flagged as UPDATE. Rows that contain new data from the newer image of the data are flagged as INSERT.

> Example
>
> You periodically update A target table that contains employee job rankings information from a source table. The table comparison flags changed data for employee Alverez and discards unchanged data for employee Davis. The result is a single row flagged with the UPDATE operation code.



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.

Parent topic: History_Preserving [page 287]

Related Information

History_Preserving transform editor [page 288]
History_Preserving transform editor options [page 289]
Example: History_Preserving data outputs [page 293]

3.4.5.4 Example: History_Preserving data outputs

Output from the History_Preserving transform contains 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:

- The software adds an INSERT row.
- The software adds an UPDATE row.

,

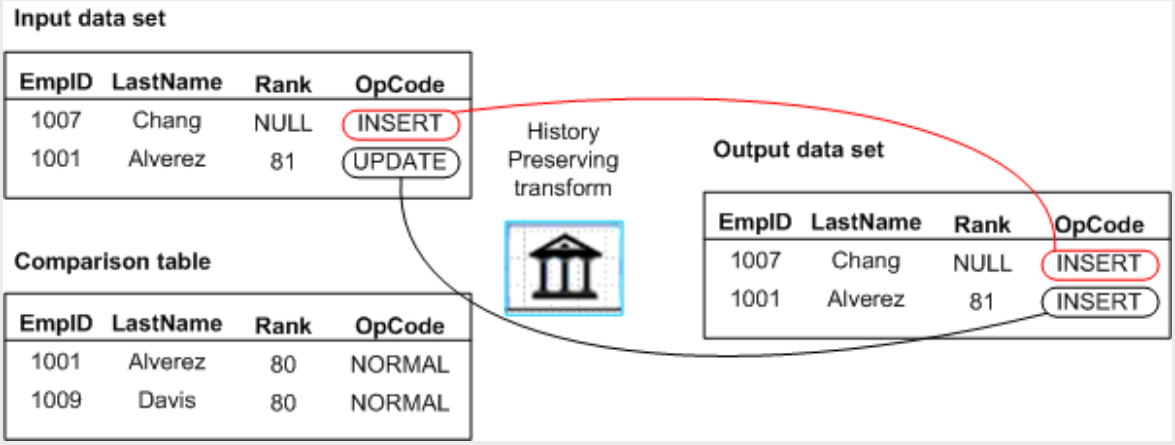
The software adds an INSERT row

An INSERT row happens when the software adds A new row to the comparison table. A new row is required under the following circumstances:

- An input row has an UPDATE flag because a value in a compare column from the input does not match a corresponding value in the comparison table.
- An input row has an INSERT flag because the primary key from the input does not appear in the comparison table.

Example

The transform produces an INSERT row with the values from the input data set row.

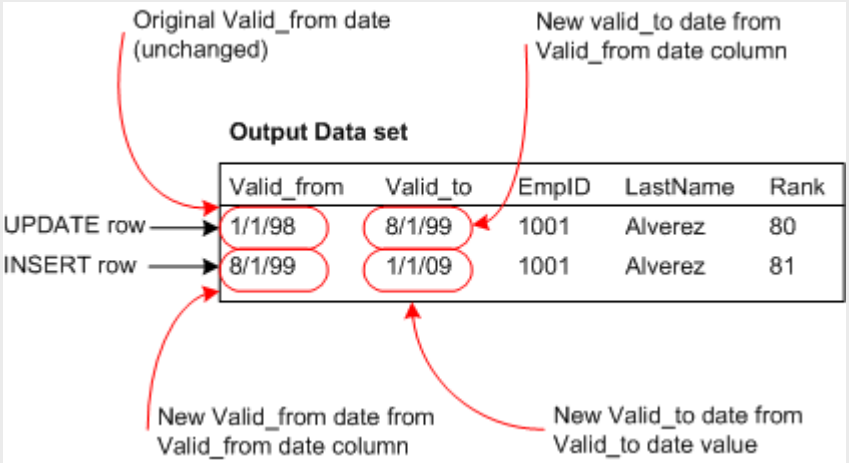


EmpID 1007, Chang is an INSERT row because it is not present in the comparison table.

EmpID 1001, Alverez is an INSERT row because the row in the Input data set has an UPDATE flag because the Rank changed from 80 to 81.

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.

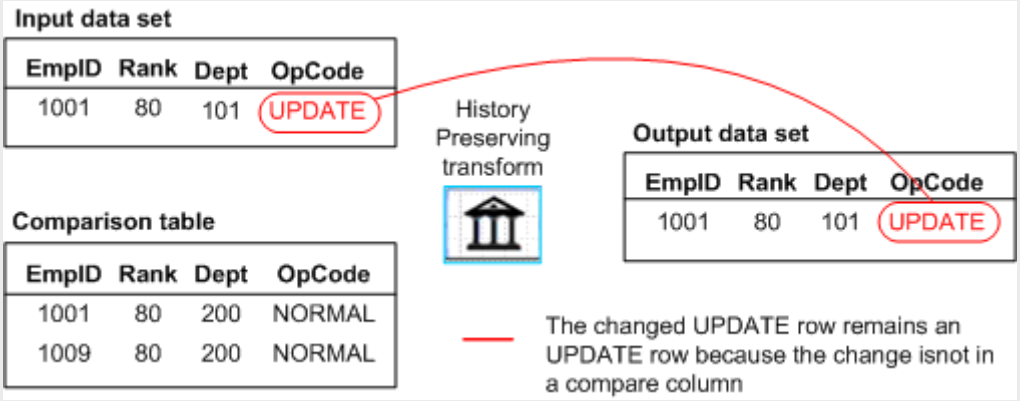


The software adds an UPDATE row

The software adds an UPDATE row under the following circumstances:

- An input row has an UPDATE flag because it contains changes, but not in the compare columns. The transform produces an UPDATE row with the values from the input data set row.

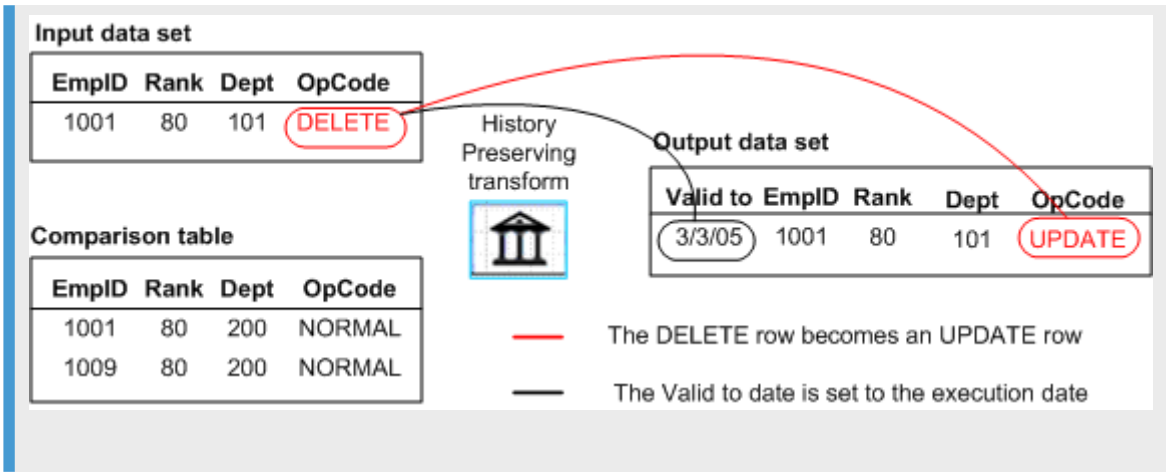
  Example

- The input row has a DELETE flag 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 job execution date, and includes the row in the output as an UPDATE row.

  Example

  Input rows flagged as DELETE contain changes, but not in the compare columns. The transform produces an UPDATE row. The update is that the row is only valid until the Valid to date.



The software passes through nested schemas in the input without change.

Parent topic: History_Preserving [page 287]

Related Information

History_Preserving transform editor [page 288]
History_Preserving transform editor options [page 289]
Example: History_Preserving data inputs [page 292]

3.4.6 Key_Generation transform

Use the Key_Generation transform to generate new keys for source data that starts from a value based on an existing key in the table.

Key_Generation information

Characteristic	Description
	Key_Generation icon
Use	Generates new keys for new rows in a data set. Generates artificial keys in a table by using the maximum existing key value in the input to use as a starting value for new keys in the output. The transform expects the generated key column to be part of the input schema.

Characteristic	Description
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.
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. **Example** 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, add a generated key to the warehouse table and fill new key values using the Key_Generation transform.

Key_Generation transform editor [page 298]
When you include the Key_Generation transform in a data flow, you configure the options in the editor.

Key_Generation transform editor options [page 298]
Specify options in the Key_Generation section to set the software behavior for this transform.

Parent topic: Data Integrator transforms [page 250]

Related Information

Data_Transfer transform [page 251]
Date_Generation transform [page 264]
Effective_Date transform [page 267]
Hierarchy_Flattening transform [page 274]
History_Preserving [page 287]
Map_CDC_Operation transform [page 300]

Pivot transform [page 308]
Reverse Pivot (Rows to Columns) [page 315]
Table_Comparison transform [page 318]
XML_Pipeline transform [page 334]

3.4.6.1 Key_Generation transform editor

When you include the Key_Generation transform in a data flow, you configure the options in the editor.

The Key_Generation transform editor includes the following sections:

- *Schema In* pane on the upper left
- *Schema Out* pane on the upper right
- Key_Generation transform options in the lower portion

To refresh the target schema after you change options, choose *View* > *Refresh* or press *F5*.

Parent topic: Key_Generation transform [page 296]

Related Information

Key_Generation transform editor options [page 298]

3.4.6.2 Key_Generation transform editor options

Specify options in the Key_Generation section to set the software behavior for this transform.

Key_Generation option descriptions

Option	Description
Generated key column	Key column in the source table that contains the existing key value. A column with the same name must exist in the input data set; the software inserts the new keys in this column on output.
Increment value	Interval between generated key values. For added flexibility, enter a variable for this option.

Option	Description
Table name	Fully qualified name of the source table from which the maximum existing key is determined (key source table). You import the table into the repository before you can use it here. *Table name* is represented as `<DATASTORE>.<OWNER>.<TABLE>` or, if you use Netezza 7.x, it is represented as `<DATASTORE>.<OWNER>.<SCHEMA>.<TABLE>`. Where: • `<DATASTORE>` is the name of the datastore Data Services uses to access the key source table. • `<OWNER>`, if required, depends on the associated database type. The following table lists database type and owner values: **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 SAP ASE / User name SAP Sybase IQ / User name

Parent topic: Key_Generation transform [page 296]

Related Information

Key_Generation transform editor [page 298]

3.4.7 Map_CDC_Operation transform

Use the Map_CDC_Operation transform to sort input data, map output data, and resolve before- and after-images for UPDATE rows.

Map_CDC_Operation information

Characteristic	Description
	Map_CDC_Operation icon
Use	Performs the following functions when you use the input requirements for the *Sequencing column* and *Row operation* column: • Sorts input data based on values in the *Sequencing column* dropdown list and optionally the *Additional grouping columns* box. • Maps output data based on values in the *Row operation column* dropdown list. The software maps source table rows 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 to filter columns and view UPDATE rows before running the job. Use the Map_CDC_Operation to support relational or mainframe changed-data capture (CDC). This transform also supports any data stream as long as you meet the input requirements. Relational CDC sources include Oracle and SQL Server. Place this transform at the end of the data flow but before the target because it produces INPUT, UPDATE, and DELETE operation codes. Data Services issues a warning if you use other objects.
Data inputs	All rows in the input data are set to NORMAL. NORMAL 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.

Characteristic	Description
Data outputs	A data set with rows flagged as specified by the values in the column that you 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 software assigns the DISCARD option under some conditions. The software does not pass through discarded rows to the output of the transform.

Parent topic: Data Integrator transforms [page 250]

Related Information

Import CDC data into tables

3.4.7.1 Map_CDC_Operation transform editor

When you include the Map_CDC_Operation transform in a data flow, you configure the options in the editor.

The Map_CDC_Operation editor contains the following sections:

- *Schema In* in the upper left
- *Schema Out* in the upper right
- Transform options in the lower pane

Select the required columns from your input schema to sequence the input rows and map operations to the output. Select the columns from the *CDC Columns* dropdown lists.

To sort the input using additional columns, click a column in the *Schema In* pane 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. The software automatically removes them in the output schema. To propagate the DI_SEQUENCE_NUMBER and the DI_OPERATION_TYPE columns to the output, create additional columns that map to them in the input data to the Map_CDC_Operation transform.

Parent topic: Map_CDC_Operation transform [page 300]

Related Information

Map_CDC_Operation transform editor options [page 303]
Map_CDC_Operation: Preserve row order on output [page 304]
Rules for resolving before and after image pairs [page 305]
Map_CDC_Operation transform filters out updated rows [page 307]

3.4.7.2 Map_CDC_Operation transform editor options

Specify values for the Map_CDC_Operation transform options to set the software behavior for this transform.

Map_CDC_Operation option descriptions

Option	Description
Additional grouping columns	In addition to the *Sequencing column*, you can sort input on additional columns by dragging them into this box from the input schema. Sorts are prioritized based first on the sequencing column and then on the order of the columns added to this box.
Define columns to filter updated rows	Drag and drop columns from the input schema to this box to compare CDC updated rows in the before and after images. If the values in the columns differ between the before and after images, Data Services generates an UPDATE row, otherwise the rows are filtered out.
Input already sorted by sequencing column	By default, the transform assumes that you have already sorted the input data based on the value selected in the *Sequencing column* box. If you deselect this checkbox, Data Services re sorts the input data using the value in the *Sequencing column* box. Use the re sort capability of this transform only when necessary as it impacts job performance.
Row operation column	Required. Specifies a column with one of the following output operation codes for each row: • `I` for INSERT • `B` for before-image of an UPDATE • `U` for after-image of an UPDATE • `D` for DELETE If you are using a relational or mainframe CDC source table, the `DI_OPERATION_TYPE` column is automatically selected as the *Row operation column*.
Sequencing column	(Required) Specifies an integer used to order table rows. If you are using a relational or mainframe CDC source table, the `DI_SEQUENCE_NUMBER` column is automatically selected as the *Sequencing column*.

Parent topic: Map_CDC_Operation transform [page 300]

Related Information

Map_CDC_Operation transform editor [page 302]
Map_CDC_Operation: Preserve row order on output [page 304]
Rules for resolving before and after image pairs [page 305]
Map_CDC_Operation transform filters out updated rows [page 307]

3.4.7.3 Map_CDC_Operation: Preserve row order on output

When you use change data capture (CDC) on target data, preserve row order for data consistency.

The following example illustrates why preserving row order is important.

Example

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` is 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, you preserve the order of the original set of operations.

The sequencing column values are also useful for resorting input coluns. If you use before and after images for update rows, it is possible that the pairs may be separated, multiplied, or lost depending on the design of your data flow. You can resort input columns as needed by using the sequencing column and any number of additional columns.

When the before and after images of an UPDATE row have the same sequence value the sorted before and after image rows are listed in pairs correctly.

Parent topic: Map_CDC_Operation transform [page 300]

Related Information

Map_CDC_Operation transform editor [page 302]
Map_CDC_Operation transform editor options [page 303]
Rules for resolving before and after image pairs [page 305]
Map_CDC_Operation transform filters out updated rows [page 307]

3.4.7.4 Rules for resolving before and after image pairs

The Map_CDC_Operation transform has specific rules to resolve 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 transform uses the value in *Row Operation Column*. If there are before images in the input stream, the transform combines the before (`B`) and after image (`U`) row pairs into one UPDATE row.

  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 transform outputs the following four rows:

Sequencing Column	Operation Column	Internal Operation Code
1	I	INSERT
2	U	UPDATE (before and after images)
3	D	DELETE
4	U	UPDATE (before and after images)

- If there are no before images (`B`) in the input stream, the after images (`U`) alone produce UPDATE rows.

  Example

  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 transform outputs the following four rows:

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.

Example

Given the following six input rows:

Sequencing Column	Operation Column	Internal Operation Code
1	U	NORMAL
1	B	NORMAL
2	B	NORMAL
3	B	NORMAL
4	B	NORMAL
2	U	NORMAL

The transform outputs the following two UPDATE rows:

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 rows 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.

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

The transform outputs the following two rows

Sequencing Column	Operation Column	Internal Operation Code
1	U	UPDATE (before and after images)
4	I	INSERT

Parent topic: Map_CDC_Operation transform [page 300]

Related Information

Map_CDC_Operation transform editor [page 302]
Map_CDC_Operation transform editor options [page 303]
Map_CDC_Operation: Preserve row order on output [page 304]
Map_CDC_Operation transform filters out updated rows [page 307]
Using before-images

3.4.7.5 Map_CDC_Operation transform filters out updated rows

It's important to know how to set the transform so that it excludes updated rows from your output file.

Use column filtering so that the transform filters out change data capture (CDC) UPDATE rows on output. To filter out updated rows, drag and drop columns from the input schema to the *Define columns to filter updated rows* box in the Map_CDC_Operation transform editor.

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 is filtered out. If there is no column filter, all updated rows coming from the input stream are 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.

> **i Note**
>
> Because the *Define columns to filter updated rows* option filters out CDC updated rows, a target table should have a primary key that identifies the row to update the change record.

Parent topic: Map_CDC_Operation transform [page 300]

Related Information

Map_CDC_Operation transform editor [page 302]
Map_CDC_Operation transform editor options [page 303]
Map_CDC_Operation: Preserve row order on output [page 304]
Rules for resolving before and after image pairs [page 305]

3.4.8 Pivot transform

Use the Pivot transform to rotate the values in specified columns to specified rows.

Pivot information

Characteristic	Description
	Pivot transform icon
Use	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.
Data inputs	A data set with rows flagged as NORMAL.

Characteristic	Description
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.

Pivot transform editor options [page 309]
Specify values in the Pivot transform editor options section to set the software behavior for this transform.

Pivot transform examples [page 311]
Examine the examples in this topic to help you learn how to use the Pivot transform.

Parent topic: Data Integrator transforms [page 250]

Related Information

Data_Transfer transform [page 251]
Date_Generation transform [page 264]
Effective_Date transform [page 267]
Hierarchy_Flattening transform [page 274]
History_Preserving [page 287]
Key_Generation transform [page 296]
Map_CDC_Operation transform [page 300]
Reverse Pivot (Rows to Columns) [page 315]
Table_Comparison transform [page 318]
XML_Pipeline transform [page 334]

3.4.8.1 Pivot transform editor options

Specify values in the Pivot transform editor options section to set the software behavior for this transform.

Pivot transform option descriptions

Pivot transform option	Description
Data field column	Name of the column that contains the pivoted data. This column contains all of the values in *Pivot columns*. The data type of this column is determined by the data type set in *Pivot columns*. If two or more *Pivot columns* contain different data types, Data Services converts the columns to a single data type; the data type of the first column you add to the pivot set.

Pivot transform option	Description
Header column	Name of the column that contains the pivoted column names. This column lists the names of the columns where the corresponding data originated.
Non-pivot columns	Columns in the source that are to appear in the target without modification.
Pivot columns	Set of columns to be rotated into rows. Describe these columns in the *Header column*. Describe the data in these columns in the *Data field column*.
Pivot sequence column	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. **Example** For example, if the row corresponds to the first column pivoted, the sequence number for the row is 1.
Pivot set	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 has a unique *Data field column* and *Header column*. Data Services automatically saves this information.

Parent topic: Pivot transform [page 308]

Related Information

Pivot transform examples [page 311]

3.4.8.2 Pivot transform examples

Examine the examples in this topic to help you learn how to use the Pivot transform.

Example: Rearrange expense table

Example

You keep your expenses in a table that contains a row for each employee, with a column for each expense type. However, because the expenses are scattered over several columns, you can't calculate expense summaries.

Original expense table

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

Use the Pivot transform to 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 so that all of the expenses are in the same column. You only need one pivot set in this case. Set the Pivot transform options as shown in the following table.

Pivot transform option	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

Note

To update the output schema for the Pivot transform, select *ViewRefresh* or press *F5*.

The resulting data set includes the following:

- The employee name, which you entered for the *Nonpivot columns* option.

- New columns: Expense_Type and Expense
  - *Header column* = Expense_Type
  - *Data field column* = Expense
- New values for the pivot sequence: *Pivot columns* = Internal_Expense Travel_Expense Misc_Expense. These were columns in the original table.

Because you don't need to include the manager ID (Mgr_ID) column, you do not include it in the Pivot transform options.

The result is a table with a single column of expense values that you can easily summarize.

New table

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

Example: Create two pivot sets

Example

Your expense table contains two types of data spread out over two months:

- Expenses
- Days traveling both domestically and internationally

Original expense table

Emp_name	Dom_Exp	Int_Exp	Dom_Day	Int_Day
AAA	2000.00	5000.00	10	5
BBB	3000.00	0.00	0	0

Emp_name	Dom_Exp	Int_Exp	Dom_Day	Int_Day
CCC	4800.00	800.00	15	1

To create a target table that has the data in two columns, expenses and days, you create two pivot sets:

- Pivot set 1: Pivot on the expense columns
- Pivot set 2: Pivot on the day columns

Pivot set 1

Create a pivot set to pivot on the expense columns by making the Pivot transform settings shown in the following table.

Pivot set 1 transform settings

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

Pivot set 2

Use a second Pivot transform to pivot on the day columns using the values in the following table.

Pivot set 2 transform settings

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

The output data set includes the employee name (not pivoted), and the following new columns:

- Pivot sequence
- Expense type
- Expense data
- Day type
- Day data

Because you linked the pivot data, domestic and international data is contained in unique rows.

Resulting data table

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

i Note

When working with multiple pivot sets, pivoted columns cannot contain a different number of rows.

Example: Add artificial column

Example

If your target expense table from the pervious example contained additional expenses (such as 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 add a new artificial column containing NULL values to the input data set, and associate the day columns with those additional expenses.

New output table with NULLs

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

Emp_name	Seq	Expense_Type	Expenses	Day_Type	Num_Days
BBB	1	Dom_Exp	3000.00	Dom_Day	12
BBB	2	Int_Exp	0.00	Int_Day	0
BBB	3	Internal_Exp	350.00	NULL	NULL
BBB	4	Misc_Exp	140.00	NULL	NULL
CCC	1	Dom_Exp	4800.00	Dom_Day	15
CCC	2	Int_Exp	800.00	Int_Day	1
CCC	3	Internal_Exp	1000.00	NULL	NULL
CCC	4	Misc_Exp	55.00	NULL	NULL

Parent topic: Pivot transform [page 308]

Related Information

Pivot transform editor options [page 309]

3.4.9 Reverse Pivot (Rows to Columns)

Use the Reverse Pivot to rotate the values in specified rows to columns.

Reverse Pivot information

Characteristic	Description
	Reverse Pivot icon
Use	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.

Characteristic	Description
Data inputs	A data set with rows flagged as NORMAL.
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.

Reverse Pivot transform options [page 316]
Specify options in the Reverse Pivot transform editor to set the software behavior for this transform.

Reverse_Pivot transform examples [page 317]
To help you learn more about Reverse_Pivot transform, we provide examples.

Parent topic: Data Integrator transforms [page 250]

Related Information

Data_Transfer transform [page 251]
Date_Generation transform [page 264]
Effective_Date transform [page 267]
Hierarchy_Flattening transform [page 274]
History_Preserving [page 287]
Key_Generation transform [page 296]
Map_CDC_Operation transform [page 300]
Pivot transform [page 308]
Table_Comparison transform [page 318]
XML_Pipeline transform [page 334]

3.4.9.1 Reverse Pivot transform options

Specify options in the Reverse Pivot transform editor to set the software behavior for this transform.

Reverse Pivot option descriptions

Option	Description
Axis value	The value of the pivot axis column that represents a particular set of output columns. A set of *Pivoted columns* is generated for each axis value. There should be one *Axis value* for each unique value in the *Pivot axis column*.
Column Prefix	Text added to the front of the *Pivoted column* names when creating new column names for the rotated data. An underscore separates the prefix name from the pivoted column name.

Option	Description
Default value	The value stored when a rotated column has no corresponding data. If you do not enter a value, the default is "null". Do not enter a blank.
Duplicate value	Action taken when a collision occurs. A collision occurs when there is more than one row with the same key and value in the *Pivot axis column*. In this case, you can store either the first row or the last row, or you can abort the transform process.
Input data is grouped	Select to indicate whether you have already sorted the input rows based on columns specified in the *Non pivot columns* box. Presorting can improve the performance of the transform.
Non-pivot columns	The columns in the source table that will appear in the target table without modification.
Pivot axis column	The column that determines what new columns are needed in the output table. At runtime, the transform creates a new column for each *Pivoted column* and each unique value in this column.
Pivoted columns	The columns containing data you want rotated into the same row. A set of columns is created for each unique value in the *Pivot axis column*.

Parent topic: Reverse Pivot (Rows to Columns) [page 315]

Related Information

Reverse_Pivot transform examples [page 317]

3.4.9.2 Reverse_Pivot transform examples

To help you learn more about Reverse_Pivot transform, we provide examples.

Example

You have a table that contains contact information for each employee. Each row in the table contains data for a particular employee and contact type.

Original employee data table

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 has multiple rows for each employee, finding information, such as a missing contact, for a particular employee is difficult. Use the Reverse_Pivot transform to rearrange your data into a more searchable form without losing the category information.

Set the Reverse_Pivot transform options to pivot the data so that all of the contact information for a particular employee is in the same row. The following table contains the Reverse_Pivot transform settings.

Reverse_Pivot transform settings

Reverse_Pivot 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 and two fields—name and phone—for each pivot axis. 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.

Resulting table

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

Parent topic: Reverse Pivot (Rows to Columns) [page 315]

Related Information

Reverse Pivot transform options [page 316]

3.4.10 Table_Comparison transform

To compare two data sets and produce the difference between them as a data set, use the Table_Comparison transform .

The resulting data set has rows flaggged as INSERT, UPDATE, or DELETE.

Table_Comparison information

Characteristic	Description
	Table_Comparison icon
Use	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. **i Note** To use the Table_Comparison transform with Teradata versions 13 and later as the comparison or target table, perform the followng tasks: • 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;`
Data inputs	There are two types of data inputs: • Input data set: The data set from a source or the output from another transform. Only rows flagged as NORMAL are considered by the transform. • Comparison table: The specification for a database table to compare to the input data set. 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.

Characteristic	Description
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. For more information about data outputs, see Table_Comparison transform data outputs [page 321].

Parent topic: Data Integrator transforms [page 250]

Related Information

Data_Transfer transform [page 251]
Date_Generation transform [page 264]
Effective_Date transform [page 267]
Hierarchy_Flattening transform [page 274]
History_Preserving [page 287]
Key_Generation transform [page 296]
Map_CDC_Operation transform [page 300]
Pivot transform [page 308]
Reverse Pivot (Rows to Columns) [page 315]
XML_Pipeline transform [page 334]

Distributed data flow execution
Teradata datastore

3.4.10.1 Table_Comparison transform data outputs

A data output from the Table_Comparison transform contains only the rows that are different between the source tables compared.

Set the Table Comparison transform in a data flow with two source tables as follows:

- An input data set
- A comparison table

The transform selects rows from the comparison table based on the primary key values in the input data set. The transform compares columns that exist in the schemas for both inputs.

Time and data data

If a column has a `date` data type in one table and a `datetime` data type in the other table, the transform compares only the date sections of the data. The columns can also be `time` and `datetime` data types, in which case SAP Data Services compares only the time sections of the data.

Resulting data set

The transform generates a data set consisting of rows with INSERT and UPDATE operation codes. However, if you use the *Detect Deleted row(s) from comparison table* option, Data Services produces DELETE rows. 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 outputs from the transform:

- INSERT row
- UPDATE row
- DELETE row
- Ignored row

Parent topic: Table_Comparison transform [page 318]

Related Information

Table_Comparison: INSERT row data [page 322]

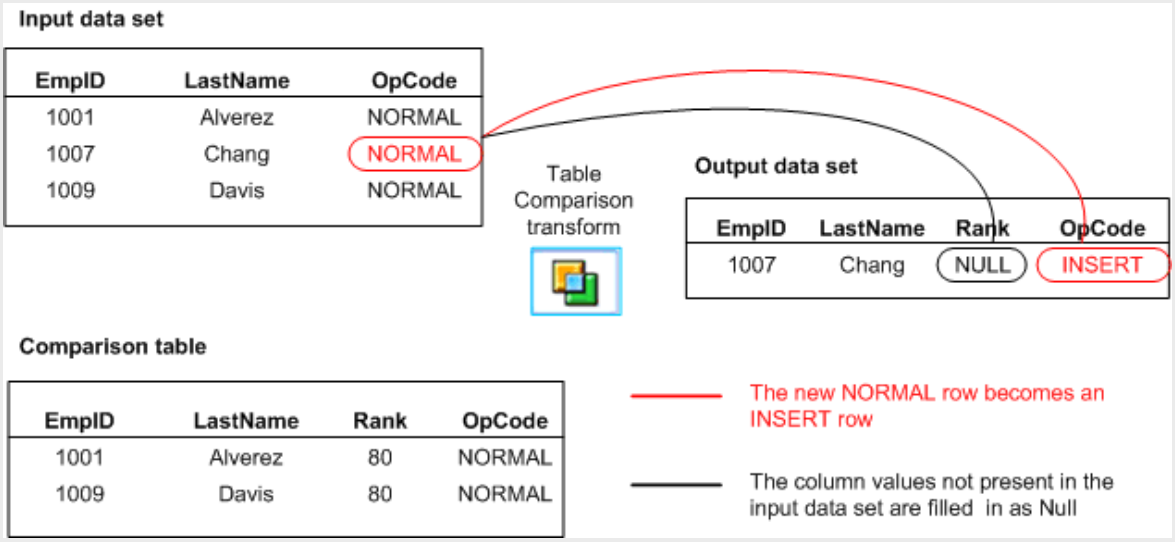
Table_Comparison: UPDATE row data [page 323]
Table_Comparison transform considerations [page 324]
Table_Comparison transform editor options [page 325]
Operation codes

3.4.10.2 Table_Comparison: INSERT row data

When the primary key value from the input data set does not match a value in the comparison table, the Table_Comparison transform produces an INSERT row.

The INSERT row contains 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.

Example



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, such as *Cached comparison table* or *Sorted input*.
- Add the `rtrim` or `rtrim_blank` function to remove trailing blanks from the input data.

Parent topic: Table_Comparison transform [page 318]

Related Information

Table_Comparison transform data outputs [page 321]
Table_Comparison: UPDATE row data [page 323]
Table_Comparison transform considerations [page 324]
Table_Comparison transform editor options [page 325]

3.4.10.3 Table_Comparison: UPDATE row data

Data that is not the same between the source table and the comparison table in non primary key columns is output as an UPDATE row.

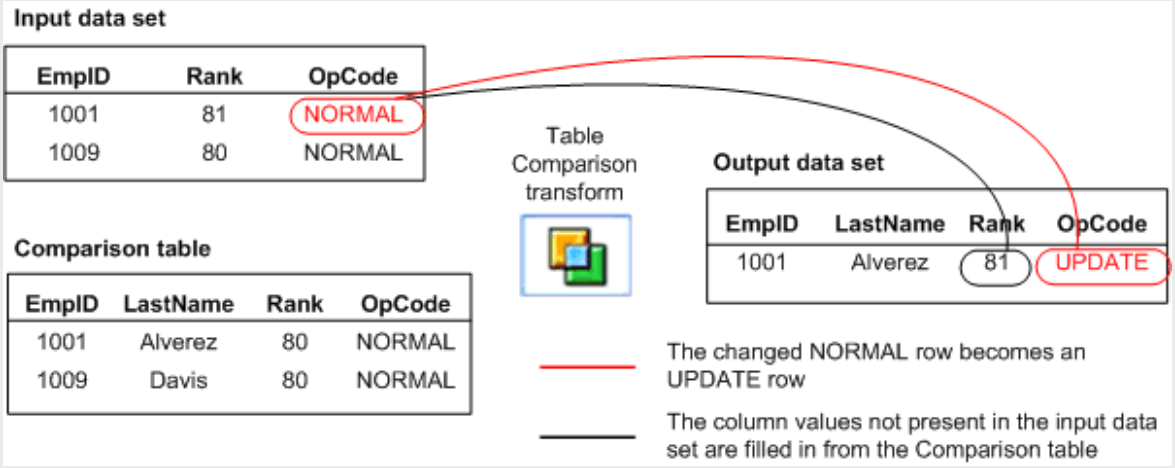
Data Services produces an UPDATE row under the following circumstances:

- The primary key value from the input data set matches the primary key value in the comparison table.
- The values from the comparison table in the non-key compare columns are different than the corresponding values in the input data set.

Data Services determines the content of the UPDATE row as follows:

- If there are differences between the non-key compare columns, the UPDATE row contains the values from the input data row set.
- If there are columns in the comparison table that are not present in the input data set, the UPDATE row contains the column name and values from the comparison table.

Example



> **! Restriction**
>
> 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.

Parent topic: Table_Comparison transform [page 318]

Related Information

Table_Comparison transform data outputs [page 321]
Table_Comparison: INSERT row data [page 322]
Table_Comparison transform considerations [page 324]
Table_Comparison transform editor options [page 325]

3.4.10.4 Table_Comparison transform considerations

Before you complete the options in the Table_Comparison transform editor, consider your primary keys and how to handle duplicate keys.

Consider the following factors to help you decide which Table_Comparison transform options to complete:

- Consider whether individual records in your data have changed more than once.

  > **Example**
  >
  > Do duplicate primary key rows exist, or will they be encountered during job processing?

- If records have changed more than once, decide if you want to record all changes, or just the latest change.

If you think that a record (row) will change more than once during job execution, consider a plan to handle duplicate keys. Duplicate keys can cause data corruption. The Table_Comparison transform has options to prevent this type of data corruption.

- *Input contains duplicate keys*: This option makes the Table_Comparison transform output all duplicate key rows from the input data set columns.
- *Generated Key column*: This option reads only the largest in a set of duplicated keys. When you select *Generated key column* with the *Detect Deleted row(s) from comparison table*, Data Services either detects deletes in all duplicate rows or in the row with the largest generated key.

For the *Table Name* option, you specify the table owner name as part of the table name. The value is based on the database type as described in the following table.

Database type	Table owner value
DB2	Data source dependent

Database type	Table owner value
Informix	Informix-defined user name
Microsoft SQL Server	User name
ODBC	Data source dependent
Oracle	User name
SAP Sybase SQL Anywhere	User Name
SAP Sybase	User name

Parent topic: Table_Comparison transform [page 318]

Related Information

Table_Comparison transform data outputs [page 321]
Table_Comparison: INSERT row data [page 322]
Table_Comparison: UPDATE row data [page 323]
Table_Comparison transform editor options [page 325]

3.4.10.5 Table_Comparison transform editor options

The Table_Comparison editor contains options that you complete based on the type of data that has changed since you last updated the target data.

The following table describes the options in the Table_Comparison transform editor.

> **i Note**
>
> Make sure to read Table_Comparison transform considerations [page 324] before you complete the options.

Table_Comparison option	Description
Table name	The fully qualified name of the source table from which SAP Data Services determines the maximum existing key (key source table). Select the table from the drop-down list. **i Note** The table must already exist in your repository. After you select the table, Data Services displays the table as follows: `<DATASTORE>.<OWNER>.<TABLE>`. If the table is Netezza 7.x, manually enter the table name as `<DATASTORE>.<OWNER>.<SCHEMA>.<TABLE>`. The elements of the table name are: • `<DATASTORE>`: Name of the datastore that Data Services uses to access the key source table • `<OWNER>`: Depends on the database type. For details, see the table in Table_Comparison transform considerations [page 324].

Table_Comparison option	Description
Generated Key Column	Optional. Provides a method of processing duplicate keys in the comparison table. **Example** Your employee data table includes a social security number as a primary key. However, there are multiple entries for some of these keys. Specify a column in the comparison table with unique keys as the generated key column. If you do not set this option, your comparison table contains rows with the same primary keys, and the transform chooses which of the rows to compare. A generated key column indicates which row is used in the comparison of a set that contains identical primary keys. Specify an existing column name that does not have duplicate keys. The Generated key column option insures that: • For an UPDATE, the output data set contains the largest generated key found for a given primary key. • For an INSERT, the output data set contains a NULL value for the generated key because Data Services omits this column from the Table Comparison transform and sets it by the Key_Generation transform later. • If an input column is used as generated key column, its value is preserved. • For a DELETE, the output data set includes all duplicate key rows or just the row with the largest key value. • If there is more than one row in the comparison table with the same primary key value and generated key value, the transform chooses the row to compare. • If the input data set and the comparison table have the column you specified in *Generated key column*, the transform does not compare the values for this column; it preserves the value.

Table_Comparison option	Description
Input contains duplicate keys	Optional. Provides a method of processing duplicate keys in the input data set. If you have more than one row with the same key in the *Input Primary Key* box, then select this checkbox. Data Services processes all DUPLICATE, INSERT, UPDATE, and DELETE rows. The INSERT, UPDATE, and DELETE rows occur in the same order as they occur in the input table. If you do not set this option, and your input columns have duplicate keys, the transform chooses the rows to compare during data flow processing. **i Note** Because this option uses additional memory to track rows with duplicate keys, use this option only when necessary. If the columns that 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.

Table_Comparison option	Description
Detect deleted row(s) from comparison table	Optional. Generates DELETE rows for all rows that are in the comparison table and not in the input set. Assumes that the input set represents the complete data set. By default this option is turned off. **i Note** The Table_Comparison transform flags rows as DELETE. However, rows actually become DELETE rows only after the data flow completes all other processing. If you select the following options, an additional section appears in the Table_Comparison editor. The section allows you to specify how to handle DELETE rows with duplicate keys: • *Generated key column* • *Detect deleted row(s) from comparison table* • *Row-by-row select* • *Sorted input* comparison method If you choose these options, also choose one of the following for deleted rows with the same key value: • *Detect all rows*: Outputs detected DELETE rows for all rows. • *Detect row with largest generated key value*: Outputs detected DELETE rows only for the generated key row with the largest value. If you select this option and the *Cached comparison table* option, Data Services always deletes the row with largest generated key value. When you leave the *Generated key column* option blank, then the secondary options with the *Detect Deleted row(s) from comparison table* are not enabled. If Data Services finds duplicate keys in the comparison table, then the DELETE output is corrupted. **i Note** If you choose the *Detect Deleted row(s) from comparison table* option, then the performance of the data flow is slower. The comparison methods most affected are the *Row-by-row select*, followed by *Cached compare* table, then *Sorted input* option. For *Row-by-row select* and *Cached compare* table, Data Services processes the deleted rows at the end of the data flow. For *Sorted input*, Data Services processes deleted rows as they are encountered in the data flow. *Run as a separate process*: Creates a separate data flow process for the Table_Comparison transform.

Table_Comparison option	Description
Comparison method group: Method for accessing the comparison table	
Row-by-row select	Looks up the target table using SQL every time it receives an input row. Consider this option when the target table is large compared to the number of rows input to the transform.
Cached comparison table	Loads the comparison table into memory. Queries the comparison table access memory rather than the actual table. Consider this option when you compare the entire target table. Data Services uses pageable cache as the default. If the table fits in the available memory, change the cache type to in-memory in the data flow.
Sorted input	Reads the comparison table in the order of the primary key columns using sequential read. Improves performance because Data Services reads the comparison table only once. To take advantage of this option, the order of the input data set must match the order of all primary key columns in the Table_Comparison transform. If the input data set order matches the order of all primary key columns, drag the primary key columns from the input schema into the Input primary key columns box. Using a sequential read, Data Services reads the comparison table in the order of the primary key columns. If you presort the input data, add a query between the source and the Table_Comparison transform. In the Query transform, drag the primary key columns from the *Schema In* pane into the *Order By* box. The 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. This set up explicitly adds a sort operation (ORDER BY) to the query input data set. ORDER BY ensures that the input data set order matches the order of the comparison table in the transform.

Table_Comparison option	Description
Input primary key column(s)	Specify the input data set columns that uniquely identify each row. These columns must be present in the comparison table with the same column names and data types. Drag each column from the *Schema In* pane into the *Input primary key column(s)* box. The transform selects rows from the comparison table that match the values from the primary key columns in the input data set. **Considerations:** • If values from more than one column are required to uniquely identify each row in the table, add more than one column to the *Input primary key columns* box. • If the columns that 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. **i Note** Do not include nested schemas in the *Input primary key columns* list.

Table_Comparison option	Description
Compare columns	Optional. Compares only a selected subset of columns between the input data set and the comparison table. Drag the columns to use from the input table in the *Compare columns* box. Comparing select columns improves performance. If you don't specify columns, Data Services compares each input column that is also in the comparison table as compare columns. **i Note** Not applicable for Long or Blob data types, or the Generated key column. Do not add primary key columns to the compare column list. The transform compares primary key columns before the compare columns. The compare columns apply only if the primary key value from the input data set matches a value in the comparison table. If the primary key value from the input data set does not match a value in the comparison table, Data Services generates an INSERT row and stops comparisons. Data Services generates an UPDATE row with the values from the input data set row when: • The primary key value from the input data set matches a value in the comparison table. • Values in the non key compare columns differ in the corresponding rows from the input data set and the comparison table.

Table_Comparison option	Description
Filter	Optional. SQL that limits the rows from the comparison table that are considered for comparison against the input data set. **⚠ Caution** Incorrect filtering can produce false `INSERT` rows. Construct your filter expression carefully to avoid unexpected results. Valid filter expressions elements: • Columns from the comparison table • Constants such as integers, strings, substitution parameters, global variables, and expressions with these combinations • Relational operators such as =, <, >, <=, >=, `LIKE`, `IN` • Logical operators such as `AND`, `OR` **i Note** Do not use Data Services functions, joins, and other SQL statements in filter expressions. Examples of valid filter expressions: • `TC.col1 IN (1, 2, 3)` • `TC.col1 >= 1 AND TC.col1 < 1000` • `TC.col1 = 1 AND TC.col2 = 2` • `col1 = 10 OR col1 = 20` • `TC.col1 = $V_DEPTNO`

Parent topic: Table_Comparison transform [page 318]

Related Information

Table_Comparison transform data outputs [page 321]
Table_Comparison: INSERT row data [page 322]
Table_Comparison: UPDATE row data [page 323]
Table_Comparison transform considerations [page 324]

3.4.11 XML_Pipeline transform

Use the XML_Pipeline transform to process large XML input sources in small batches.

The folowing table describes the characteristics of the XML_Pipeline transform.

XML_Pipeline information

Characteristic	Description
	XML_Pipeline icon
Use	With this transform, Data Services does not read the entire XML input into memory and 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 the transform 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. Limitation: Because the XML_Pipeline transform pushes operations to the XML source during execution, do not use a breakpoint between your XML source and an XML_Pipeline transform.
Data inputs	XML file or XML message. Connect more than one XML_Pipeline transform to an XML source.
Data outputs	N/A

Rules for XML_Pipeline transform input and output schemas [page 335]
There are specific rules to follow when you drag objects from the *Schema In* to the *Schema out* panes.

XML_Pipeline editor [page 336]
Specify input and output schemas in the XML_Pipeline transform editor.

XML_Pipeline transform example [page 336]
View the example data flow that contains the XML_Pipeline transform to help you understand how to use it in your jobs.

Parent topic: Data Integrator transforms [page 250]

Related Information

Data_Transfer transform [page 251]
Date_Generation transform [page 264]
Effective_Date transform [page 267]
Hierarchy_Flattening transform [page 274]
History_Preserving [page 287]
Key_Generation transform [page 296]
Map_CDC_Operation transform [page 300]
Pivot transform [page 308]
Reverse Pivot (Rows to Columns) [page 315]
Table_Comparison transform [page 318]

3.4.11.1 Rules for XML_Pipeline transform input and output schemas

There are specific rules to follow when you drag objects from the *Schema In* to the *Schema out* panes.

When you drag and drop objects from the input schema to the output schema in the XML Pipeline transform editor, consider the following requirements:

- Do not drag the root level schema from the input schema to the output schema.
- Drag the same child object as many times as necessary from the input schema to the output schema only when each instance of the object has a unique name.
  - Rename the mapped instance before attempting to drag an object to the output for the second and subsequent time.
- If you drag a column or sub schema from the input to the output schema, do not map the parent schema for that column or sub schema.
  Similarly, if you drag a parent schema from the input to the output schema, do not map an individual column or sub schema from under that parent.
- Do not map items from two sibling repeating sub schemas. This mappping causes a Cartesian product, which the XML_Pipeline transform does not support.

Parent topic: XML_Pipeline transform [page 334]

Related Information

XML_Pipeline editor [page 336]
XML_Pipeline transform example [page 336]

3.4.11.2 XML_Pipeline editor

Specify input and output schemas in the XML_Pipeline transform editor.

In addition to input and output schema panes, the *Mapping* tab displays how Data Services maps 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. The Designer shows this information with the icons described in the following table.

	Root schema and repeating sub schema
	Non-repeating sub schema

Within each sub schema, the Designer shows mapped and unmapped columns using the icons described in the following table.

	Column not used in output mapping
	Column used in output mappings

Parent topic: XML_Pipeline transform [page 334]

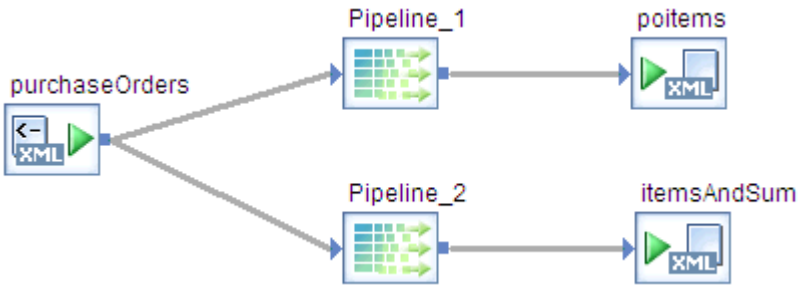
Related Information

Rules for XML_Pipeline transform input and output schemas [page 335]
XML_Pipeline transform example [page 336]

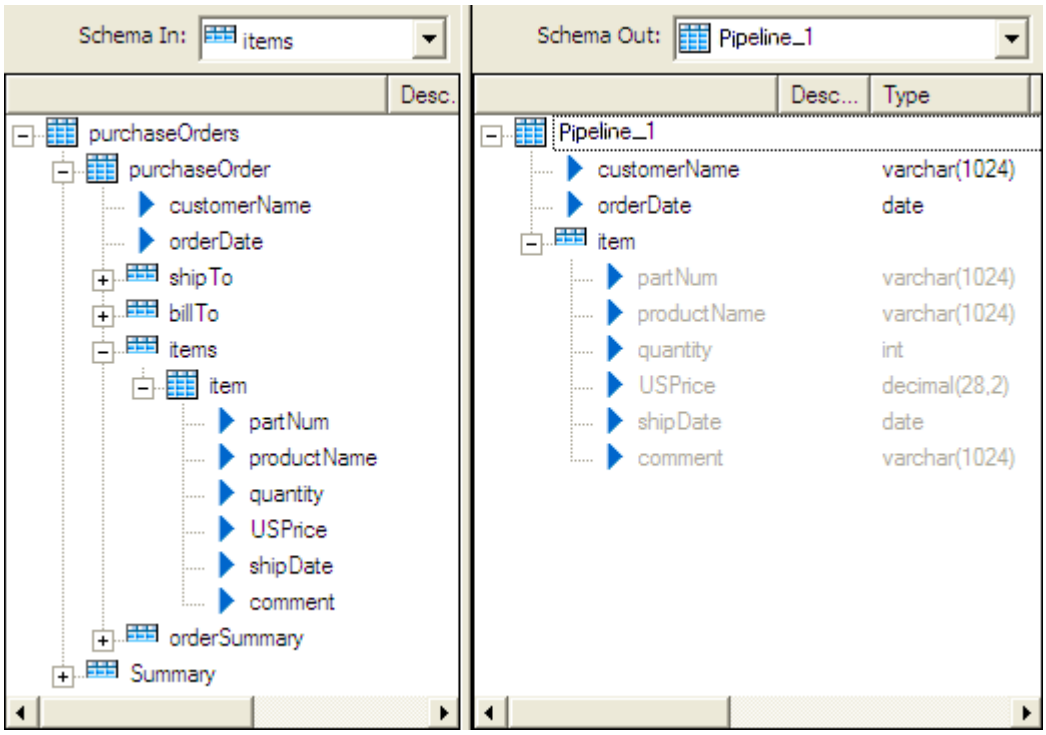
3.4.11.3 XML_Pipeline transform example

View the example data flow that contains the XML_Pipeline transform to help you understand how to use it in your jobs.

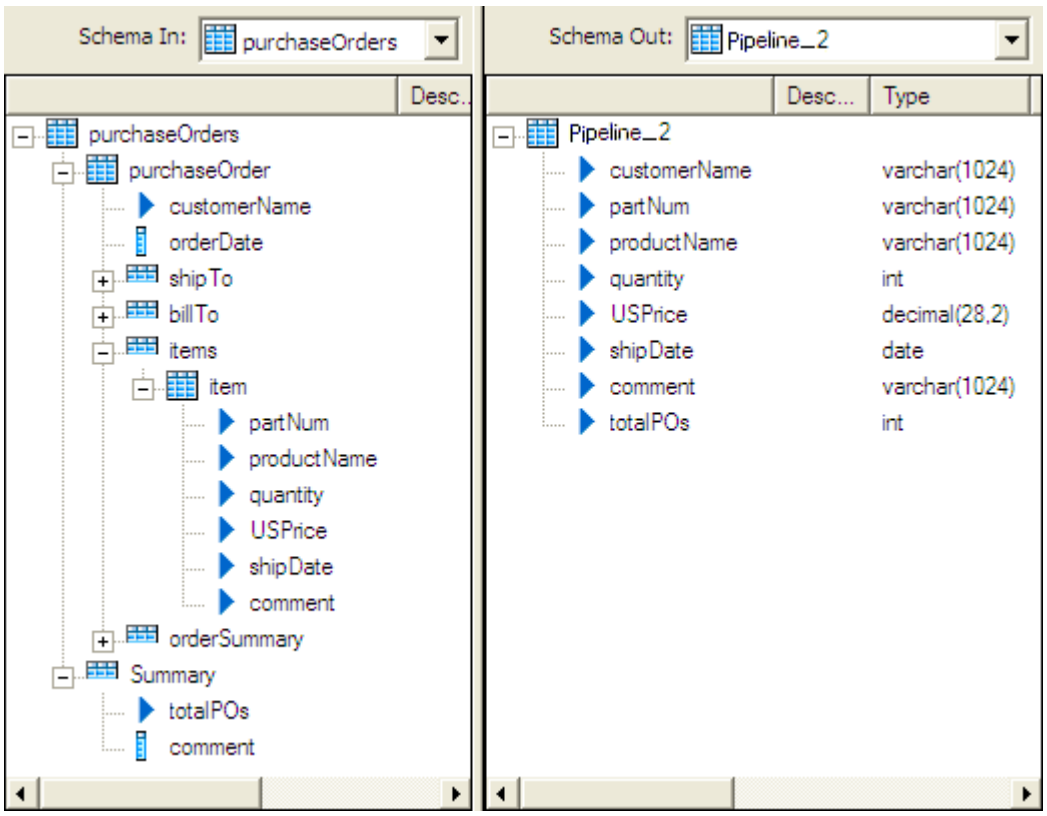
This simple data flow contains two XML_Pipeline transforms, but only one of them takes full advantage of the pipelining power.



- 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.

Parent topic: XML_Pipeline transform [page 334]

Related Information

Rules for XML_Pipeline transform input and output schemas [page 335]
XML_Pipeline editor [page 336]
Query transform [page 960]

3.5 Data Quality transforms

Transforms that help you improve the quality of your data.

The Data Quality transforms can parse, standardize, correct, enrich, match, and consolidate your customer and operational information assets.

Parent topic: Transforms [page 240]

Related Information

Transform reference [page 240]
Dynamic transform settings [page 244]
Embedded help for transform editors [page 249]
Data Integrator transforms [page 250]
Platform transforms [page 840]
Text Data Processing transforms [page 1025]

3.5.1 Download Data Quality blueprints and other content objects

We periodically post new and updated Data Quality blueprints and other SAP Data Services content and information to the SAP Community Network Web site for your use.

Data Quality blueprints

We create blueprints based on common scenarios that you are likely to handle with 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, use the blueprint by modifying the settings for your specific needs.

For each scenario, we include a blueprint that is already set to solve the business problem in that scenario. Run the blueprint data flows in your environment with only a few modifications.

Download all of the blueprints or only the blueprints that you find useful from the SAP Community Network Web site. Refer to this site frequently for updated content and use the forums to ask us any questions or to make requests. We have also provided the ability for you to share any content that you have developed with the rest of the development community.

Data Quality transform configurations

A transform configuration is a transform with preconfigured input fields, output fields, and options that can be used in multiple data flows. Transform configurations are useful if you repeatedly use a transform with specific options and input and output fields.

When you install Data Services, the installer includes read-only transform configurations for the Data Quality transforms.

You can use transform configurations in your data flows or use them as examples of a typical transform. After you place an instance of the transform configuration in a data flow, you can override the preset defaults. You can also create your own transform configuration, either by replicating an existing transform configuration or creating a new one.

Task localization file for data quality statistics

The task localization file translates information in the data quality statistics tables into recognizable terms and descriptions.

When you use the Data Cleanse, Geocoder, and Global Address Cleanse transforms and you generate the data quality statistics tables. The software sends data quality statistics to the corresponding tables in the repository based on the job set up. The tables contain lines of identification codes generated by the software about the various data quality processes performed on your data. The identification codes require a translation file that converts the identification codes into recognizable terms or descriptions.

The task localization file name is `dqs_task_localization_<language>.csv`. We provide the file in several languages based on the languages that are most commonly used in Data Services. Download the task localization file from the SAP Community Network website at: http://scn.sap.com/docs/DOC-68523.

Data Quality transforms [page 341]
Data Services provides read only Data Quality transform configurations that are partially configured for specific types of situations.

Downloading the data quality statistics task localization file [page 346]
When you query the data quality statistics files, you need the task localization file to translate the codes from the tables into understandable and useful information.

Downloading blueprints and other content objects [page 347]
Learn how to download and install blueprints and other content objects.

3.5.1.1 Data Quality transforms

Data Services provides read only Data Quality transform configurations that are partially configured for specific types of situations.

When you use a transform configuration, the software makes a copy of the transform configuration so that you can make changes based on the specific job. The following Data Quality transform configurations are available from the *Transforms* tab of the object library. When you use them in your data flows, complete the configuration based on your business case.

Associate transform

Transform configuration	Description
AssociateGroupStatistics_AssociateBatch	A sample Associate transform that generates group statistics.
Base_Associate	A sample base Associate transform that you can use to configure an Associate transform.
Wizard_AssociateBatch	A sample Associate transform configuration that you use with the Match Wizard. Do not edit this configuration.

Country ID transform

Transform configuration	Description
CountryID2Char	A sample Country ID transform that generates the two-character ISO country code.

Data Cleanse transform

Transform configuration	Description
Base_DataCleanse	A sample base Data Cleanse transform that you use to configured a base to a custom Data Cleanse transform.
Chinese_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, e-mail, and phone data using Chinese based data quality rules.
Dutch_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, e-mail, and phone data using Dutch based data quality rules.
EnglishNorthAmerica_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, e-mail, and phone data using English based data quality rules.
French_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, e-mail, and phone data using French based data quality rules.
German_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, e-mail, and phone data using German based data quality rules.
Italian_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, e-mail, and phone data using Italian based data quality rules.
Japanese_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, e-mail, and phone data using Japanese based data quality rules.
Portuguese_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, e-mail, and phone data using Portuguese based data quality rules.
Spanish_DataCleanse	A sample Data Cleanse transform configured to cleanse name, title, firm, date, e-mail, and phone data using Spanish based data quality rules.

Geocoder transform

Transform configuration	Description
Geocode	A sample Geocoder transform configured to assign latitude and 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 and longitude reference point.
ReverseGeocode	A sample Geocoder transform configured to assign an address based on a latitude and longitude reference point.

Global Address Cleanse transform

Transform configuration	Description
Australia_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Australia.
Brazil_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Brazil.
Canada_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Canada.
China_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in China.
Europe_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in multiple European countries.
France_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in France.
Germany_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Germany.
Global_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse Latin-1 address data in any supported country.
GlobalSuggestions_Address_Cleanse	A sample Global Address Cleanse transform configured to cleanse Latin-1 address data in any supported country using the Suggestion List feature.
Greece_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Greece when the address data consists of Greek data.
Italy_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Italy.
Japan_AddressCleanse	A sample Global Address Cleanse transform configured to cleanse address data in Japan when the address data consists of Japanese kanji, katakana, and hiragana.

Transform configuration	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.

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.

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.

Transform configuration	Description
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 e-mail data.
NameFirmAddress_MatchBatch	A sample Match transform configured to identify matching data records based on similar name, firm, and address data.
NameFirmAddressJapan_MatchBatch	A sample Match transform configured to identify matching data records based on similar name, firm, and address data when the data consists of Japanese kanji, katakana, and hiragana.
NameIndividualId_MatchBatch	A sample Match transform configured to identify matching data records based on similar name and individual identification number data.
NamePhone_MatchBatch	A sample Match transform configured to identify matching data records based on similar name and phone data.
ProductDescription_MatchBatch	A sample Match transform configured to identify matching data records based on similar product description data.
Wizard_MatchBatch	A sample Match configuration used by the Match Wizard. Do not edit this configuration.

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.

Transform configuration	Description
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.
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.

Parent topic: Download Data Quality blueprints and other content objects [page 340]

Related Information

Downloading the data quality statistics task localization file [page 346]
Downloading blueprints and other content objects [page 347]
Creating a custom transform configuration

3.5.1.2 Downloading the data quality statistics task localization file

When you query the data quality statistics files, you need the task localization file to translate the codes from the tables into understandable and useful information.

To download the SAP Data Services task localization file:

1. Open the Data Services SAP Community Network at http://scn.sap.com/docs/DOC-68523.
2. Find the applicable task localization file based on the language you need, and what is available. The file name is `dqs_task_localization_<language>.csv`.
3. Follow the download instructions in the SCN Data Quality Statistics Task Localization page.

Task overview: Download Data Quality blueprints and other content objects [page 340]

Related Information

Data Quality transforms [page 341]
Downloading blueprints and other content objects [page 347]
Download Data Quality blueprints and other content objects [page 340]

3.5.1.3 Downloading blueprints and other content objects

Learn how to download and install blueprints and other content objects.

Go the the SAP Community Network at https://wiki.scn.sap.com/wiki/display/EIM/SAP+Data+Services+Blueprints.

1. Open the *Content Objects User Guide* to view a list of all of the available blueprints and content objects and their descriptions.

   The *Content Objects User Guide* also contains instructions for downloading and setting up the blueprints.
2. Select the blueprint that you want to download.
3. Follow the instructions in the *Content Objects User Guide* to download the files to the appropriate location and make the necessary modifications in SAP Data Services to run the blueprints.

Task overview: Download Data Quality blueprints and other content objects [page 340]

Related Information

Data Quality transforms [page 341]
Downloading the data quality statistics task localization file [page 346]

3.5.2 About Data Quality fields

Many SAP Data Services transforms require mapped input and generated output fields.

Map Data Quality fields in the *Input* and *Output* tabs of the transform editor.

For detailed information about the input and output fields applicable for each transform, see the individual transform topics.

Data Quality field content types [page 348]
All input and output fields have an assigned content type that identifies the type of data in the field.

Changing default attribute values [page 350]
SAP Data Services has default settings for field attribute values that control how the software preserves data when you reimport table and columns.

3.5.2.1 Data Quality field content types

All input and output fields have an assigned content type that identifies the type of data in the field.

Setting the content type helps you map your fields when you set downstream transforms. The software 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.

> **Example**
>
> You have a column of data in your data source that is comprised mostly of first names called Given Name1. The software automatically maps the field to a Data Quality-recognized content type, Given_Name1.

You can change the automatic content type in the *Column Properties* window. Open the window by double-clicking the field name in the *Schema Out* grid of the transform editor. Select a different content type from the *Content Type* drop list.

You can specify the content type directly in your source data in the object Library, including source data from XML Schemas, COBOL Copybooks, flat files, Excel, and IDOC objects. If the data in a column cannot be automatically mapped, then you will see a blank content type. Content types in an XML Schema can be changed in the Designer object library, and the change is reflected in all data flows where the schema is used. You cannot change the content type in an XML Schema from within a single data flow.

A few situations to be aware of

If you are importing an XSD or DTD format, you can have the software automatically assign the content by selecting *Automatically Assign Content Type* in the *Import DTD Format* window.

If you attempt to merge the contents of two corresponding columns in the Merge transform, be certain that the content types match. If the content types do not match, the software issues 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 is overwritten.

If you reimport a column or table in Designer, the content types for all of the existing columns are preserved by default. For instructions to change the default setting for content type, see “Editing default content type preserve setting”.

Available content types

The following content types are available for each transform.

- <Blank>
- Address
- Address Primary Name
- Address Primary Number
- Address Primary Postfix
- Address Primary Prefix
- Address Primary Type
- Address Secondary Number
- Country
- Date
- Delivery Point
- DPV Status
- Email
- Family Name1
- Family Name1 Match Std
- Family Name2
- Family Name2 Match Std
- Firm
- Firm Location
- Firm Location Match Std
- Firm Match Std
- Given Name1
- Given Name1 Match Std
- Given Name2
- Given Name2 Match Std
- Group Number
- Locality
- Lot
- Lot Order
- Name
- Name And Firms
- Phone
- Postcode
- Postcode1
- Postcode2
- Postname
- Postname Match Std
- Prename
- Prename Match Std

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

Parent topic: About Data Quality fields [page 347]

Related Information

Changing default attribute values [page 350]
Changing default attribute values [page 350]

3.5.2.2 Changing default attribute values

SAP Data Services has default settings for field attribute values that control how the software preserves data when you reimport table and columns.

To change the default behavior when you re import a column and you want the software to clear existing attribute values:

1. In *Designer*, select *Tools* > *Options* > .
2. In *Options*, expand *Designer* and select *Attribute Values*.
3. Select either *Column* or *Table* from the *Object Type* dropdown list.
4. Make default settings for each applicable attribute.

> **Example**
>
> To change the default setting for *Content Type*, click the corresponding *Action* dropdown list and select *Clear*. When you reimport a column, the software clears existing content type attributes.

Task overview: About Data Quality fields [page 347]

Related Information

Data Quality field content types [page 348]

3.5.3 About data quality statistics

The data quality statistics tables enable you to analyze many aspects of the quality of your data.

The data quality statistics tables contain summary and record-level statistics about the cleansing and assignment processes performed on your data. Generate the tables by setting the Data Quality Statistics options in the applicable transforms: Data Cleanse, Geocoder, Global Address Cleanse.

3.5.3.1 Data quality statistics common requirements

When you generate data quality statistics, there are several requirements that apply to more than one type of statistics tables.

> **i Note**
>
> Generate data quality statistics tables in the Data Cleanse, Global Address Cleanse, and Geocoder transforms only.

Statistics tables and common requirements

Statistics table	Requirement
All data quality statistics tables: CLEANSE_ADDRESS_RECORD_INFO_TABLE CLEANSE_CHANGE_INFO_TABLE CLEANSE_COMPONENT_INFO_TABLE CLEANSE_INFO_CODES_TABLE CLEANSE_STATISTICS_TABLE GEOCODE_INFO_CODES_TABLE GEOCODE_STATISTICS_TABLE	Requires the Task Localization file, which you download from the SAP Customer Portal. The Task Localization file converts raw data that you extract from the data quality statistics tables into understandable language.
CLEANSE_STATISTICS_TABLE GEOCODE_STATISTICS_TABLE	Does not require the *ROW_ID* field output field. These statistic tables contain summary statistics and the *ROW_ID* field is required only for non-summary statistics tables. The software issues a warning when your job set up: • Includes the *ROW_ID* output field but is not configured to generate non-summary data quality statistics. • Excludes the *ROW_ID* output field but is configured to generate non-summary data quality statistics.
CLEANSE_INFO_CODES_TABLE CLEANSE_CHANGE_INFO_TABLE CLEANSE_COMPONENT_INFO_ TABLE CLEANSE_ADDRESS_RECORD_INFO_TABLE GEOCODE_INFO_CODES_TABLE	Requires the *ROW_ID* output field. These statistics tables contain non-summary statistics and the *ROW_ID* field is required to join the transform input or output with the output statistics. The software issues a warning when your job set up: • Excludes the *ROW_ID* output field but is configured to generate non-summary data quality statistics. • Includes the *ROW_ID* output field but is not configured to generate non-summary data quality statistics. You can use *ROW_ID* to join the transform input or output with the non-summary statistics table information. For example, use *ROW_ID* to drill into the before and after changes in a specific component that was cleansed.
CLEANSE_INFO_CODES_TABLE CLEANSE_CHANGE_INFO_TABLE CLEANSE_COMPONENT_INFO_TABLE CLEANSE_ADDRESS_RECORD_INFO_TABLE GEOCODE_INFO_CODES_TABLE	Requires that you map an input file primary key to the output file. Use the mapped primary key to join the input with any of the data quality statistics reports.

Statistics table	Requirement
Applies to all data quality statistics tables: CLEANSE_ADDRESS_RECORD_INFO_TABLE CLEANSE_CHANGE_INFO_TABLE CLEANSE_COMPONENT_INFO_TABLE CLEANSE_INFO_CODES_TABLE CLEANSE_STATISTICS_TABLE GEOCODE_INFO_CODES_TABLE GEOCODE_STATISTICS_TABLE	If you generate data cleansing statistics, requires that you set up a field for data cleansing. If you do not set up a field for data cleansing, the software does not generate data cleansing statistics for that field. **Example** An input file contains firm, address, and phone information. You set up the firm and address fields for data cleansing, but you do not set up the phone field for data cleansing. The software generates statistics that include the firm name and address information, but not the phone information.
CLEANSE_INFO_CODES_TABLE CLEANSE_CHANGE_INFO_TABLE CLEANSE_STATISTICS_TABLE CLEANSE_COMPONENT_INFO_TABLE	Configure to generate in both the Data Cleanse and Global Address Cleanse transforms. Not applicable for the Geocoder transform.

Parent topic: About data quality statistics [page 351]

Related Information

Data quality statistics transform set up requirements [page 353]
Data quality statistics input field requirements for Data Cleanse [page 354]
Requirements for CLEANSE_COMPONENT_INFO_TABLE [page 355]
Requirements for Geocoder and data quality statistics [page 357]
Requirements for Global Address Cleanse and the CLEANSE_COMPONENT_INFO_TABLE [page 357]
Download Data Quality blueprints and other content objects [page 340]
Data Cleanse Report and Analysis [page 383]
Data quality statistics tables and supplemental content information [page 1402]
Data quality statistics transform set up requirements [page 353]

3.5.3.2 Data quality statistics transform set up requirements

To successfully generate the data quality statistics tables for Data Cleanse, Geocoder, and Global Address Cleanse, you must adhere to specific job set up specifications or the software will not generate data quality statistics.

Parent topic: About data quality statistics [page 351]

Related Information

Data quality statistics common requirements [page 351]
Data quality statistics input field requirements for Data Cleanse [page 354]
Requirements for CLEANSE_COMPONENT_INFO_TABLE [page 355]
Requirements for Geocoder and data quality statistics [page 357]
Requirements for Global Address Cleanse and the CLEANSE_COMPONENT_INFO_TABLE [page 357]
Data quality statistics input field requirements for Data Cleanse [page 354]
Requirements for CLEANSE_COMPONENT_INFO_TABLE [page 355]
Requirements for Geocoder and data quality statistics [page 357]
Requirements for Global Address Cleanse and the CLEANSE_COMPONENT_INFO_TABLE [page 357]

3.5.3.3 Data quality statistics input field requirements for Data Cleanse

When you generate data quality statistics, the Data Cleanse transform requires that you configure specific input fields.

The following table lists the supported and unsupported input fields for generating data quality statistics in the Data Cleanse transform. It also includes specific input field configurations that are not supported for generating data quality statistics.

Input field requirements for data quality statistics in Data Cleanse

Supported input fields	Unsupported input fields	Unsupported input field configurations
• Email# • Firm_Line# • Discrete_Firm • Name_Line# • Name_or_Firm_Line# • Discrete_Person • Phone# • Title_Line#	• Date# • Multiline# • SSSN# • UDPM#	The software issues a warning when you enable data quality statistics and the following conditions exist: • You include multiline input fields. • You include fields associated with name or title. • You map *Discrete_Person* input fields and *Name_Line* input fields in the same job setup. • You map *Discrete_Firm* input fields and *Firm_Line* input fields in the same job setup.

Parent topic: About data quality statistics [page 351]

Related Information

Data quality statistics common requirements [page 351]
Data quality statistics transform set up requirements [page 353]
Requirements for CLEANSE_COMPONENT_INFO_TABLE [page 355]
Requirements for Geocoder and data quality statistics [page 357]
Requirements for Global Address Cleanse and the CLEANSE_COMPONENT_INFO_TABLE [page 357]

3.5.3.4 Requirements for CLEANSE_COMPONENT_INFO_TABLE

To generate the CLEANSE_COMPONENT_INFO_TABLE in Data Cleanse, ensure that you adhere to the output field configuration requirements for parsing person, firm, phone, and email fields.

The CLEANSE_COMPONENT_INFO statistics table contains the exact output field start position and length of individual data elements that the transform parsed during the cleansing process.

Person parsing

The software generates output field position data only for person-type output fields that have the following characteristics:

- Parent component is DUAL_NAME* or PERSON*
- FIELD_CLASS is STANDARDIZED

The software does not generate output field position data for person-type output fields that have the following field name values. These values are exceptions to the above-supported person field characteristics:

- Name_Special
- Gender
- Gender_ID
- Rule_Label
- Score
- *_Match_Std*
- Match_*

Firm parsing

The software generates output field position data for firm-type output fields that have the following characteristics:

- Parent component is FIRM*

- FIELD_CLASS is STANDARDIZED

The software does not generate output field position data for firm-type output fields that have the following field name values. These values are exceptions to the above-supported firm field characteristics:

- Rule_Label
- Score
- *_Match_Std*
- Match_*

Phone parsing

The software generates output field position data for phone-type output fields that have the following characteristics:

- Parent component is PHONE*
- FIELD_CLASS is STANDARDIZED

The software does not generate output field position data for phone-type output fields that have the following field name values. These values are exceptions to the above-supported phone field characteristics:

- *_Match_Std*
- Match_*

Email parsing

The software generates output field position data for e-mail-type output fields that have the following characteristics:

- Parent component is EMAIL*
- FIELD_NAME is EMAIL
- FIELD_CLASS is STANDARDIZED

Parent topic: About data quality statistics [page 351]

Related Information

Data quality statistics common requirements [page 351]
Data quality statistics transform set up requirements [page 353]
Data quality statistics input field requirements for Data Cleanse [page 354]
Requirements for Geocoder and data quality statistics [page 357]
Requirements for Global Address Cleanse and the CLEANSE_COMPONENT_INFO_TABLE [page 357]

3.5.3.5 Requirements for Geocoder and data quality statistics

To generate data quality statistics for the Geocoder transform, ensure that you set up the transform with the supported mode.

Applicable Geocoder modes

Supported mode	Unsupported mode
Address Geocoding mode	• Reverse Geocoding mode • Point of Interest (POI) Textual Search mode The software issues a warning when you are in this mode and you select to generate data quality statistics.

Parent topic: About data quality statistics [page 351]

Related Information

Data quality statistics common requirements [page 351]
Data quality statistics transform set up requirements [page 353]
Data quality statistics input field requirements for Data Cleanse [page 354]
Requirements for CLEANSE_COMPONENT_INFO_TABLE [page 355]
Requirements for Global Address Cleanse and the CLEANSE_COMPONENT_INFO_TABLE [page 357]
Geocoding

3.5.3.6 Requirements for Global Address Cleanse and the CLEANSE_COMPONENT_INFO_TABLE

To generate the CLEANSE_COMPONENT_INFO_TABLE in Global Address Cleanse, ensure that you adhere to the configuration requirements.

The CLEANSE_COMPONENT_INFO statistics table contains the exact output field location (start position and length) of individual data elements that the transform parses during the cleansing process. However, when you include any *NW_* input fields, the transform does not generate the input field start position and length.

The transform generates the output field start position and length for output fields with the following characteristics:

- FIELD_CLASS is BEST
- FIELD_CATEGORY is COMPONENT
- FIELD_ADDRCLASS is DELIVERY or DUAL.

Exceptions to the above-supported field characteristics include the following output fields:

- Multiline1-12
- Match_*

Parent topic: About data quality statistics [page 351]

Related Information

Data quality statistics common requirements [page 351]
Data quality statistics transform set up requirements [page 353]
Data quality statistics input field requirements for Data Cleanse [page 354]
Requirements for CLEANSE_COMPONENT_INFO_TABLE [page 355]
Requirements for Geocoder and data quality statistics [page 357]

3.5.4 Associate transform

The Associate transform combines or associates matching results generated from the Match transform.

Associate information

Characteristic	Description
	Associate icon

Characteristic	Description
Use	The Associate transform works downstream from the Match transform to provide a way to combine or associate match results by using the group number generated by the Match transform. Optionally add a Group Statistics operation to the Associate transform to gather match statistics. 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. **i Note** When using the Associate transform to associate match results from more than two Match transforms, we recommend that you associate only two Match group numbers at a time. Processing two match results at a time is more efficient than processing more than two match results together in one Associate transform. Use the Associate group number generated in a downstream Associate transform and associate it with the next match group number. For more information, see "Associating more than two match results" in the *Reference Guide*.
Content objects	Transform configurations, blueprints, and other content objects.

Related Information

Associating more than two match results [page 377]
Data Quality transforms [page 341]
Download Data Quality blueprints and other content objects [page 340]

3.5.4.1 Associate transform options

Transform level options for the Associate transform appear in the *Association Editor*, which you open using the *Tools* menu.

To open the *Association Editor*, select the Associate transform in your data flow and select *Tools* > *Associate Editor.*

The following table contains descriptions for the options that appear in the right pane when you click *Transform Options* in the left pane of the *Association Editor*.

Transform Options descriptions

Option	Description
Associate set name	Required. Type a name for your Associate set.
Generate report data	Required. Generate report data for this transform. *YES*: The default setting. Generates report data for this transform. *NO*: Turns off report data generation. You would disable report data generation if you do not need to generate reports. For example, if you work in production, you may not need report data. Turning off report data generation improves performance.
Logical source field	Optional. Select the field that contains the ID for the logical source from the dropdown list. The logical source is a group of records that span multiple input sources or a subset of records from a single input soure.
Physical source field	Optional. Select the field that contains the ID for the physical source (reader) from the dropdown list.
Run as a separate process	Specify whether to split the transform into a separate process. Running as a separate process can improve performance and throughput by using separate memory and computer resources. *YES*: Splits the transform into a separate process. *NO*: The default setting. Keeps the transform in the same process as the rest of the data flow.

Related Information

Run as separate process
Track physical and logical sources

3.5.4.2 Association editor options

Association options for the Associate transform appear in the *Association Editor*.

Open the *Association Editor* by selecting the Associate transform in your data flow and clicking *Tools* > *Associate Editor.*

The following table contains descriptions for the options that appear when you click an existing association in the left pane of the *Association Editor*. For example, if you have one association group, click *ASSOCIATION1*.

Add as many association groups as you want.

Associationn option descriptions

Option	Description
Association name	Type a name for this association. Ensure that the name is unique among other associations in this transform.
Group Number field	Click *Add Row* to add a line to the Group numbers table. Select the field that contains the group number data from the dropdown list. The field is from the Match transform. Click *Add Row* to add as many group number fields as desired. Click *Remove Row* to remove a row.

3.5.4.3 Associate transform editor post processing

The *Association Editor* contains a post processing page, which you use to set up any post processing operations.

Access the *Post associate* options by selecting the transform icon in the data flow and selecting *Tools* > *Association Editor.*

Add operations to the table by clicking *Add Operation*. Select one of the following operations from the *Add Operation* dropdown list.

- *Group Prioritization*
- *Group Statistics*
- *Best Record*
- *Unique ID*

Use the Post associate operations table to navigate to your operations by double-clicking the desired row in the table.

Group Prioritization

Use the Group prioritization operation to order records for processing by other post-match operations.

Group statistics

Use Group statistics to generate statistical information about your group of matching records. Statistics include the following information:

- Number of records within the match group
- Sequential group order number

- Group rank

> **Note**
>
> Group rank flags one record within each group of matching records as the master record. It flags all other records in the group as subordinate records.

- Records in a match group that belong to more than one source

Group statistics are essential for generating data for reports.

Best record

Use the Best record operation to salvage data from matching records that belong to match groups, and consolidate or post the data to a best record or to all matching records.

Unique ID

Use the Unique ID operation to have the software assign sequential identification numbers to each new record added to a data warehouse. The software carries the largest sequential number assigned in a project over to the next project and adds 1 to continue the sequential numbering. The new number is the beginning identification number for assigning the next group of sequential identification numbers. occurs when the software processes the next source against the data warehouse file.

> **Example**
>
> The largest sequencial number from project 1 is 155. The software starts the numbering for project 2 at 156 for the first unique ID.

Output record

Use the Output record options to flag certain types of records for potential processing downstream.

Related Information

Match transform Output Flag Selection options [page 668]

Best record

Unique ID

3.5.4.3.1 Match and Associate transforms Best Record tab

Use the best record post-match processing operation to specify how the software updates your records with information from other records in a match group.

The Best Record options appear in both the *Match Editor* and the *Association Editor* based on which transform you are working with.

Best Record post processing operation option descriptions

Option	Description
Best record name	Specifies a unique name for the Best Record operation. Make sure that you use a name that is unique within this Match transform. **i Note** This is the same option as the *Best record name* option in the *Destination Protection* tab. When you enter a name here, the software populates the *Best Record name* field in the *Destination Protection* tab.
Best record strategy	Specifies the strategy to determine whether any action is taken on records in a match group. After you choose the strategy, priority, and field that you want to work with, the transform automatically generates the Python code for you, except when you choose *Custom*. • *Custom:* Bases the best record strategy on Python code that you create. *Custom* enables you to open the Python Expression editor and create custom Python code. • *Date:* Bases the best record strategy on a date field. • *Length:* Bases the best record strategy on the length of data in a specified field. • *Non_Blank:* Bases the best record strategy on the completeness of data in a specified field. • *Priority_Number:* Bases the best record strategy on a number. • *Priority_String:* Bases the best record strategy on string data in a specified field.

Option	Description
Strategy priority	Specifies priorities for the best record strategy. *Strategy priority* does not control priority for the *Non_Blank* and *Custom* best record strategies. **Date** • *Newest*: The newest date in the field causes an action to take place. • *Oldest*: The oldest date in a field causes an action to take place. **Length** • *Longest:* The longest string in a field causes an action to take place. • *Shortest:* The shortest string in a field causes an action to take place. **Priority_Number** • *Highest:* The highest number in a field causes an action to take place. • *Lowest:* The lowest number in a field causes an action to take place. **Priority_String** • *Ascending:* The string with the most ascending string order causes an action to take place. • *Descending:* The string with the most descending string order causes an action to take place.
Strategy field	Specifies a field that contains data required to execute your strategy.
Posting destination	Specifies the destination record. • *Master*: Posts only to a master record. • *Subs*: Posts only to subordinate records. • *Master to Subs*: Pushes information from the master record and posts it to each subordinate record. • *All*: Posts to both the master and subordinate records.
Post only once per destination	Specifies to post once per destination. • *Yes:* Posts once per destination record. After data is posted to the destination record, the transform stops the operation. Select *Yes* when you use the *Non_Blank* best record strategy. • *No:* Posts more than once per destination. After data is posted to the destination record, the transform continues the operation and populates the destination record again with the next value. Select this option when accumulating values such as total sales. Select *No* when you use the Longest, Shortest, Newest, Oldest, Ascending, or Descending priorities. **i Note** The transform ignores the setting in this option when it uses the *Master to Subs* posting destination. With this posting destination, information can be posted only once.
View/Edit Python	Opens the Python Expression editor. Select the *Custom Best record strategy* to create custom python. If you choose any other strategy, the Python in the editor is read only.

Best record action fields table

Use the *Best record action fields* table to define actions on fields based on your selected best record strategy.

Best record action fields option descriptions

Option/Option group	Description
Source field	Specifies the name of the source field in the input record.
Destination field	Specifies the name of the destination field in the output record, or the destination of this best record action. Either set the action to post to the same input field or set the action to post to a different field.
Custom	Specifies whether you create custom Python code for the action to perform on the destination field. • *Yes:* Specifies to create custom Python code to perform an action on the destination field. • *No:* Specifies to use the same source and destination fields. When this option is set to *No*, the transform copies the contents of the source field to the destination field.
Editor	When you choose *Yes* in the *Custom* column, the word "PYTHON" appears in the *Editor* column and the software activates the *Edit Python* button.
Edit Python	Opens the *Python Editor for Best Record* dialog box. Compose your custom Python code using the options and tools in the editor.

Related Information

Best record

3.5.4.3.2 Match and Associate transforms Destination Protection tab

Use the options in the *Destination Protection* tab to protect data in particular records or data in records from particular input sources from being overwritten.

Destination Protection tab option descriptions

Option	Description
Best record name	Specifies a name for the best record operation. Make sure that you use a name that is unique within this Match transform. **i Note** This is the same option as the *Best record name* option in the *Best Record* tab. If you entered a name for *Best record name* in the *Best Records* tab, the name also appears here. If you change the name here, the software changes the name shown in the *Best Record* tab.
Enable destination protection	Enables the transform to protect records from best record operations that may overwrite the record contents or data in records from particular input sources.
Default destination protection	When you set destination protection for records based on fields: Specifies whether a destination is protected when the field that you specify in the *Destination protection field* does not have a valid value. • *Yes*: Protects a destination when the destination protection field does not have a valid value of Y or N. • *No*: Disables this type of protection. When you set destination protection for records based on input source membership: Specifies whether a destination (input source) is protected when you do not specifically define the source in the table.
Specify destination protection by field	Specifies to enable destination protection through a value in a field. The value must be Y or N. When you select this option, select a field from the *Destination Protection field* dropdown list, or *Unique ID protected field*.
Destination protection field	Specifies the field that is protected. The field must contain a value of Y or N. Any other value, including blank, causes the transform to use the *Default destination protection* value.
Specify destination protection by source	Specifies to enable destination protection through membership in a particular source. When you select this option, complete the table with source names and destination protection settings.

Option	Description
Source Name	Choose the name of the source from the dropdown list. The transform populates the list with defined sources and source groups from the *Input Sources Editor* window of the Match Editor. If there are no names listed in the dropdown list, or you do not see a source that you expected to see, add an input source through the *Input Sources Editor*. To access the *Input Sources Editor*: 1. Click the *Transform Options* node. 2. Click *Add* at the bottom of the pane at left. 3. Click *Input Sources*.
Destination protected	Specifies whether to protect records from the source in the *Source Name* column. • *Yes*: Protects records from the source from being overwritten. • *No*: Does not protect records from the source from being overwritten. If you do not specify an option for every source, the transform uses the default value in the *Default Destination Protection* option

Related Information

3.5.4.3.3 Match and Associate transforms Group Statistics Editor

Use the options in the *Group Statistics Editor* to set up an operation for post association or post match processing.

Create group statistics in the post match processing portion of a match or associate level. Open the *Group Statistics Editor* in either the Match Editor or the Associate Editor. For instructions to create group statistics, see the *Designer Guide*.

Group Statistics Editor option descriptions

Option	Description
Group statistics name	Specifies a name for the group statistics. Make sure that you use a name that is unique within this Match transform.
Generate only basic statistics	Generates statistics about match groups, such as how many records in each group. Basic statistics don't include statistics about input sources.
Generate source statistics from input sources	Generates basic statistics and statistics about input sources. To enable this option, define one or more input sources in the *Match Editor*.

Option	Description
Generate source statistics from source values	Generates source statistics based on source values in a field instead of defined input sources. When you select this option, the following options become editable: • *Logical source field* • *Default logical source value* • *Count all sources* • *Choose sources to count*
Logical source field	Specifies the input field that holds the value for your logical sources. **i Note** Only mapped input fields appear in the list.
Default logical source value	Specifies a value to use if the field specified for the *Logical source field* option is blank. If a record has a blank value in the logical source field, the transform uses the value from the *Default logical source value* option.
Count all sources	Counts all sources, regardless of the value in the field specified in *Logical source field*.
Choose sources to count	Specifies the sources to count based on values in the field specified in *Logical source field*. When you select this option, the following options become editable: • *Default count flag* • *Auto generate sources* • *Predefined count flag field* (editable only when you also select *Auto generate sources*) • *Manually define logical source count flags*
Default count flag	Specifies the value to use when the field specified in *Predefined count flag field* is invalid. For example, if the value in the field specified for *Predefined count flag field* isn't Y, N, or blank. • *YES*: Counts all of the records in the source. • *NO*: Doesn't count any of the records in the source.
Auto generate sources	Generates sources based on the value in the field specified for *Predefined count flag field*.

Option	Description
Predefined count flag field	Specifies the field that contains the Y or N indicator value. The transform behaves as follows when the specified field contains either a Y or an N: • If the specified field contains a Y, the transform counts the source for statistics. • If the specified field contains an N, the transform doesn't count the source. You must check *Auto generate sources* to enable this field.
Manually define logical source count flags	Complete the columns in this table when you select *Choose sources to count*. If you don't complete both columns, the transform disregards the selection of *Choose sources to count*.
Source value	Specifies the value in the field that identifies the logical source. This value is case-sensitive: You must enter the value using the same casing as in the field.
Count	Specifies whether the transform uses the source, based on the value you entered in the *Source value* column, in the count. • *YES*: If the transform finds the value specified in the *Source Value* column, include the source in the count. • *NO*: If the transform finds the value specified in the *Source Value* column, don't include the source in the count.

3.5.4.3.4 Match and Associate transforms Priority Order options

To control group priority order in post processing, set options in the *Priority Order* group.

The *Priority Order* options appear in both the *Match Editor* and the *Association Editor* based on which transform you are working with.

In the *Post Association Processing* or *Post Matching Processing* dialog of the `<transform_name>` *Editor*, select to add a *Group Prioritization* operation to control group order in post processing.

For more information about priority order, see the *Designer Guide*. Search for “priority order” or “group prioritization”.

Group forming prioritization

Use the *Group Prioritization* operation to order records within each break group. This order 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. Ordering records controls 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. A *Group Prioritization* operation controls the destination of data that is being propagated from other records to form a best record.

Group Prioritization Editor option description

Option	Description
Prioritization name	Specifies the name for this Group Prioritization operation. If you have multiple operations in this Match or Associate 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, such as a dollar amount or a date. Use the buttons to add, remove, and order rows. Place the primary sort field at the top of the list. Keep the rest of the fields in the order that they are positioned to determine the sub-sort that occurs.

Option	Description
Input field	Specifies a field on which to sort your records.
Field order	Specifies the order in which records are sorted.

3.5.4.3.5 Match and Associate transforms Record Completeness

To set the priority of output data based on record completeness, configure options in the *Record Completeness* tab.

Record Completeness option descriptions

Option	Description
Prioritization name	Specifies the name for this *Group Prioritization* operation. If you have multiple operations in the Match transform, use a unique name.
Order records based on completeness of data	Specifies whether to apply priority and blank penalty points to records to help control the order of your records.
Define only field penalties	Specifies whether the transform assesses penalties based on blank fields.
Define priority and penalty fields	Indicates that you have specific fields that contain the actual integer values for priority and blank penalty.

Option	Description
Record priority field	Specifies the field that contains priority values. This field must contain an integer.
Apply blank penalty field	Specifies the field that contains the indicator Y or N for applying blank penalty points to a record.
Define priority and penalty based on input source	Specifies whether to have your record priority and blank penalty indicator Y or N determined by membership in a given source.
Source Name	Specifies an input source. Select an input source from the list of sources in the *Source Name* column. The sources listed here are defined in the *Input Source* operation.
Priority	Specifies a priority value (an integer) in the *Priority* column. The lower the priority score, the higher the priority.
Apply Blank Penalty	Specifies whether to apply a blank penalty to a record based on membership to this source.
Default record priority	Specifies the default value for the record priority if: • The record does not contain a field with this value. • The field is blank for a record. • A record does not belong to any of the sources specified. The lower the priority score, the higher the priority.
Default apply blank penalty	Specifies the default indicator to add blank penalty points to records with blank fields. The transform uses the indicator under the following circumstances: • When a record does not have a field that carries this indicator. • If that field is blank or has invalid data. • If a record does not belong to any of the sources specified. Options are: • *Yes*: The transform adds each blank penalty for a record to the record priority to generate an adjusted record priority score. The lower the score, the higher the priority. • *No*: The transform does not apply a penalty when the fields are blank.
Input field	Displays the input fields that are available to assign a blank penalty score to.
Blank penalty	Specifies a penalty value (an integer) to apply to a field when the specified field is blank in a record.

3.5.4.3.6 Match and Associate transforms Unique ID options

Use the Unique ID options to assign unique sequential identification numbers to each new record that you add to a data warehouse.

The Unique ID post-match processing operation enables you to begin numbering where the highest unique ID from the previous run ended.

> **Example**
>
> The transform can carry over the largest number assigned in a particular project as the beginning identification number plus 1. The transform uses the new beginning identification number in the assignment of new sequential ID numbers. The transform assigns new sequential numbers when the software processes the next source against the data warehouse file.

In addition to configuring the transform to assign unique ID numbers to individual records, configure the transform to assign the same unique ID to every record in a match group.

Add the *Unique ID* operation as a *Post Match Processing* operation in the *Match Editor*.

Unique ID options descriptions

Option	Description
Unique ID name	Specifies a name for the Unique ID operation. Make sure that you use a name that is unique within this Match transform. For example, use the match transform name and the match level in the unique ID name. **Note** This field is the same as the *Unique ID name* field in the *Destination Protection* tab.
Processing operation	Specifies the type of unique ID processing operation to perform. Valid values include the following: • *Assign*: Assigns a new ID to unique records that need one, or assigns a new ID to all members of a group that don't have an ID. In addition, the assign operation uses an existing ID when a member of a match group already has an ID. All match group members must appear consecutively in one collection and must be in priority order (high to low). • *AssignCombine*: Performs both an assign operation and a combine operation. In addition to the Assign process, the transform combines the ID numbers 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). • *Combine*: Combines the IDs of a match group when more than one ID is represented. All match group members must appear consecutively in one collection and must be in priority order (high to low). • *Delete*: Removes unique ID numbers from records that already have an ID, unless the record is protected. • *Split*: Splits the ID numbers 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).

Option	Description
Recycle unique IDs	Specifies whether the software should reuse the unique IDs that were discarded during the delete operation in different records. If you have a limited amount of unique ID numbers available, you may want to recycle them. Valid values include: *Yes*: Recycle discarded unique IDs. *No*: Do not recycle discarded unique IDs.
ID field	Specifies a field that holds a previously assigned unique ID. If you do not specify a field, then the transform assumes that no records have a unique ID.
Starting unique ID source group	
Field	Specifies an input field from which the transform obtains the starting unique ID. For the field to appear in the dropdown list, map it from an upstream transform.
Starting unique ID field	Specifies the field that passes the starting unique ID. If the transform does not receive a Unique ID, it uses 1 as the starting number.
Constant value	Specifies that the starting ID is a positive whole number in the *Starting value* option.
Starting value	Indicates the starting unique ID value. Valid values range from 1 to UINT_MAX (unsigned integer max). The default value is 1.
Value from file	Specifies to read the starting Unique ID from the file specified in the *File* option.
File	Specifies the path and name of the file that manages unique ID numbers. Specify a value for this option only when the *Value from file* option is selected.
GUID	Specifies to use the Globally Unique Identifier (GUID) as the unique ID. GUID is also known as the Universal Unique Identifier (UUID). The UUID variation used for unique ID is a time-based 36-character string with the format: `TimeLow-TimeMid-TimeHighAndVersion-ClockSeqAndReservedClockSeqLow-Node`.
Save ending ID to file and reclaim recycled IDs	Specifies whether to save the last ID that was assigned to a file. Additionally, specifies whether to reclaim recycled IDs.
File	Specifies the file to which the transform writes the last assigned ID.

Option	Description
Allow multiple Match transforms to access unique ID file	Specifies to allow multiple Match transforms to access a shared unique ID file. When enabled, multiple data flows access the same unique ID file, and single Match transforms run in more than one process when the DOP setting is greater than 1. In addition, this option allows multiple Match transforms within a single data flow to share a single unique ID file.
Number of IDs to get when accessing file	Specifies the number of IDs to retrieve from the unique ID file during each access. For example, with a setting of 100, the first process accesses the file and retrieve IDs numbered 1-100. The next process with retrieve IDs numbered 101-200. If a process uses less than the number of retrieved IDs, the remaining IDs are written back to the unique ID file as recycled IDs. **i Note** A setting greater than 1 improves performance when sharing a unique ID file between multiple processes by reducing the number of times the file must be accessed. However, integer numbers may not be assigned in sequential order.
Group number field	Specifies the field that holds a group number. The group number is used to assign the same unique ID to more than one record. If this field is omitted, then it is assumed that each record is unique and should have its own number.

Related Information

Unique ID
Assigning unique IDs using a file

3.5.4.3.7 Match and Associate transforms Destination Protection tab

Use the *Destination Protection* tab to control whether a unique ID for a record is protected based on the source that the record belongs to.

Destination protection helps prevent IDs from being assigned to a suppression or a rented source.

Destination Protection table option descriptions

Option	Description
Unique ID name	Specifies a name for this unique ID operation. Make sure that the name is unique within this match transform. If you use other unique ID operations in this Match transform, use both the match transform name and the match level in the Unique ID name to distinguish it from other unique ID operations. **i Note** This field is the same as the *Unique ID name* field in the *Unique ID* tab.
Enable destination protection	Enables the protection of the source record unique ID numbers from being overwritten with the unique ID numbers of matching records.
Default destination protection	When you set destination protection for records based on fields: Specifies whether a destination is protected when the field that you specify in *Unique ID protected field* option does not have a valid value. • *Yes*: Protects a destination when the destination protection field does not have a valid value of Y or N. • *No*: Disables this type of protection When you set destination protection for records based on input source membership: Specifies whether a destination (input source) is protected when you do not specifically define the source in the table.
Specify destination protection by field	Specifies to enable destination protection through a value in a field.
Unique ID protected field	Specifies an input field 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) causes the transform to use the setting in *Default destination protection* option.
Specify destination protection by source	Specifies to control destination protection through membership in a particular source. Complete the table with source names and destination protection settings.

Option	Description
Source name	Choose the name of the source from the dropdown list. The transform populates the list with defined sources and source groups from the *Input Sources Editor* window of the *Match Editor*. If the dropdown list is empty, or you don't see the specific source in the list, add the source in the *Input Sources Editor*. To access the *Input Sources Editor*: 1. Click the *Transform Options* node at left. 2. Click *Add* at the bottom of the pane at left. 3. Click *Input Sources*.
Unique ID protected	Specifies whether to protect the source in the *Source name* column. • *Yes*: The source is protected. • *No*: The source is not protected.

3.5.4.4 Associate transform output fields

The Associate transform requires that you map one field on output: Group_Number.

The following table describes the Associate transform output fields and provides the default content type.

Output field name	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

Output field name	Description
Priority_Value	Record priority value assigned to record. If you did not include a priority value, this field outputs 0.

3.5.4.5 Associating more than two match results

When using the Associate transform to associate match results from more than two Match transforms, we recommend that you associate only two Match group numbers at a time.

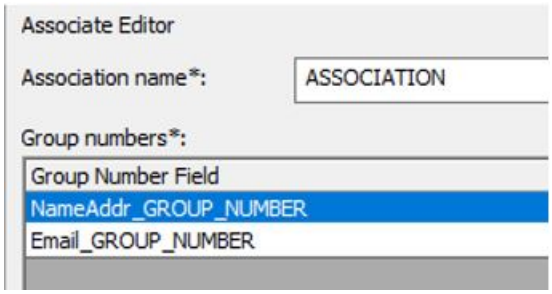
Processing two match results at a time is more efficient than processing more than two match results together in one Associate transform. Use the Associate group number generated in a downstream Associate transform and associate it with the next match group number.

For example, you may want to use the Associate transform to associate match results from three Match transforms: one matching on name and address, a second matching on email, and a third matching on SSN.

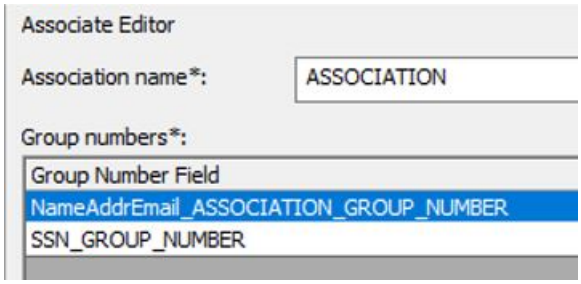
To do this, you first need to associate the match groups from the name and address with the match groups from email. Use the Associate group number generated in a downstream Associate transform with the SSN match groups.

Here is how the job should be set up using this recommendation:

The first Associate transform associates the match results from the name and address, and the email. This Associate transform generates NameAddrEmail_ASSOCIATION_GROUP_NUMBER.



Use the association results from the NameAddr_Email Associate transform to associate with the SSN match results.



Related Information

3.5.5 Country ID transform

The Country ID transform parses your input data and then identifies the country of destination for each record.

> **i Note**
>
> Reference to “country” or “countries” in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

Country ID information

Characteristic	Description
	Country ID icon
Use	The Country ID transform parses your input data and then identifies the country of destination for each record. After identifying the country, the transform outputs the country name, any of three different ISO country codes, an ISO script code, and a percentage of confidence in the assignment. Though you can use the Country ID transform before any transform in a data flow, it’s 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 outputs. It is not necessary to use the Country ID transform before the Global Address Cleanse transform in a data flow because the Global Address Cleanse transform contains its own Country ID processing. It’s also not necessary to use the Country ID transform before the USA Regulatory Address Cleanse transform because the input data contains U.S. addresses only.
Content objects	Data Quality provides you with a sample transform that can help you get started creating a Country ID transform useful to you.

Related Information

Data Quality transforms [page 341]

3.5.5.1 Coutry ID options

The Country ID options include options that are common to most transforms.

Country ID options

Option	Description
Common	
Run as Separate Process	Specifies to split the transform into separate processes for efficiency. • *Yes*: Splits the transform into separate processes. • *No*: Keeps transform in the same process as the rest of the data flow.
Reference Files	
Directory Path	Specifies the location of your directories. The directories you use depend on the country that you are processing. Select Browse to go to the directory path, or use the substitution parameter `$$RefFilesAddressCleanse` for convenience. Configure the substitution parameter before you use it in the transform set up.

3.5.5.2 Country ID transform input fields

To map input fields for the Country ID transform, use the Input_Fields option group in the transform editor.

The following table describes each Country_ID input field.

Input field name (Country ID)	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.

Input field name (Country ID)	Description
Postcode	The postal code for the address.
Region1	The state, province, territory, or region of the address.

3.5.5.3 Country ID transform output fields

Map output fields for the Country ID transform in the *Output* tab of the Country_ID transform editor.

> **i Note**
>
> Reference to "country" or "countries" in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

The following table describes the Country ID transform output fields.

Output field name (Country ID)	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 can't determine a country. *1010*: Indicates a tie in identifying the country. *1005*: Indicates that no country was identified.
Country_Name	The identified country name.
ISO_Country_Code_2Char	The 2-character ISO code for the identified country.
ISO_Country_Code_3Char	The 3-character ISO code for the identified country.
ISO_Country_Code_3Digit	The 3-digit ISO code for the identified country.
ISO_Script_Code	The 4-character script code to use for the identified country, such as LATN or KANA.

Related Information

Country ID transform input fields [page 379]

3.5.6 Data Cleanse transform

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.

Data Cleanse information

Characteristic	Description
	Data Cleanse icon
Use	The Data Cleanse transform identifies and isolates specific parts of mixed data. It then parses and formats the data based on the referenced cleansing package and options that you set directly in the transform. Use Data Cleanse to assign gender and prenames to name data and to generate Match standards for all types of data. The transform has options to enable generation of data quality statistics tables in the repository. These statistics provide insight into how the software cleansed and assigned your data by generating aggregated summary statistics and specific record-level statistics. You specify a cleansing package in the transform. The cleansing package that you specify in the transform defines how the Data Cleanse transform parses and standardizes your data. **i Note** The Social Security number algorithm is no longer updated by the Social Security Administration. The SSN validation performed in Data Cleanse is based on data through July, 2011. SAP builds the Social Security data into the cleansing package. Within a data flow, place the Data Cleanse transform after the address cleansing process and before the matching process.
Content objects	Transform configurations, blueprints, and other content objects.

Data Quality transform common option [page 383]
The Data Quality transforms have one common option that you'll see in many of the transform configurations: *Run as Separate Process*.

Data Cleanse Report and Analysis [page 383]
Select options for generating data quality statistics and select to generate report data for the Data Cleanse transform.

3.5.6.1 Data Quality transform common option

The Data Quality transforms have one common option that you'll see in many of the transform configurations: *Run as Separate Process*.

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

3.5.6.2 Data Cleanse Report and Analysis

Select options for generating data quality statistics and select to generate report data for the Data Cleanse transform.

Option	Description
Data Quality Statistics Settings	Settings that control the generation of data quality statistics tables.
Generate Report Data	Specifies whether to generate report data for this transform. *YES*: Generates report data for this transform. *NO*: Does not generate report data for this transform. Select *NO* when you don't need reports, such as during testing. Setting this option to *NO* improves performance.

3.5.6.3 Data Quality Statistics for Data Cleanse, Global Address Cleanse, and Geocoder transforms

To enable the generation of data quality statistics, make settings in the Data Quality Statistics Settings group.

The Data Quality Statistics Settings group appears in the Data Cleanse, Global Address Cleanse, and the Geocoder transforms.

The following table contains the options for the *Data Quality Statistics Settings* group. Because the group is applicable for more than one transform, the table lists the applicable transform for each option.

The software generates the following types of statistics for each transform:

- The Data Cleanse and the Global Address Cleanse transforms generate additional statistics and info codes about address cleansing results on the data.
- The Geocoder transform generates statistics and info codes that are unique to the Geocoding processes.

Option	Transform	Description
Generate Cleanse Address Record Info Table	Global Address Cleanse	Specifies whether the software generates the Cleanse Address Record Information Table. • *YES*: Generates the Cleanse Address Record Information Table. • *NO*: Does not generate the Cleanse Address Record Information Table. *NO* is the default setting. The Cleanse Address Record Information Table contains non-summary information about the results of the address cleanse process on the data. The information includes assignment information and assignment type for each address. The software generates this information only when you include the Global Address Cleanse transform for address data.
Generate Cleanse Change Info Table	Data Cleanse Global Address Cleanse	Specifies whether the software generates the Cleanse Change Information Table. • *YES*: Generates the Cleanse Change Information Table. • *NO*: Does not generate the Cleanse Change Information Table. *NO* is the default setting. The Cleanse Change Information Table contains non-summary information. It identifies areas of concern from your source data by the number of significant changes.

Option	Transform	Description
Generate Cleanse Component Info Table	Data Cleanse Global Address Cleanse	Specifies whether the software generates the Cleanse Component Information Table. • *YES*: Generates the Cleanse Component Information Table. • *NO*: Does not generate the Cleanse Component Information Table. *NO* is the default setting. **i Note** This option is available in SAP Data Services version 4.2.6 and later. The Cleanse Component Information Table contains non-summary information. It contains position information for each specified data element when one or both of the following occurs: • The transform parses the data element from an input field. • The transform writes the data element to an output field.
Generate Cleanse Info Codes Table	Data Cleanse Global Address Cleanse	Specifies whether the software generates Cleanse Information Codes Table. • *YES*: Generates the Cleanse Information Codes Table. • *NO*: Does not generate the Cleanse Information Codes Table. *NO* is the default setting. The Cleanse Information Codes table contains non-summary information. The information includes the distribution of information codes or a count of missing, suspect, or blank data.

Option	Transform	Description
Generate Cleanse Statistics Table	Data Cleanse Global Address Cleanse	Specifies whether the software generates the Cleanse Statistics Table. • *YES*: Generates the Cleanse Statistics Table. • *NO*: Does not generate the Cleanse Statistics Table. *NO* is the default setting. The Cleanse Statistics Table contains summarized information for each unique Entity ID. It provides a high-level insight into what the software changed during the cleansing process.
Generate Geocode Info Codes Table	Geocoder	Specifies whether to generate the Geocode Information Codes Table. • *YES*: Generates the Geocode Information Codes Table. • *NO*: Does not generate the Geocode Information Codes Table. *NO* is the default setting. The Geocode Information Codes Table contains non-summary information about why records were not assigned the highest level of latitude and longitude. The statistics are based on the information code data. **i Note** Not all existing information codes apply to the data quality statistics.

Option	Transform	Description
Generate Geocode Statistics Table	Geocoder	Specifies whether the software generates the Geocode Statistics Table. • *YES*: Generates the Geocode Statistics Table. • *NO*: Does not generate the Geocode Statistics Table. *NO* is the default setting. The Geocode Statistics Table contains totals for the following: • The number of rows in the data • The number of rows that were assigned a latitude and longitude.

Related Information

Data quality statistics common requirements [page 351]
Data quality statistics transform set up requirements [page 353]
Content information for data quality statistics tables [page 1404]

3.5.6.4 Data Cleanse Cleansing Package

Specify a cleansing package and set related options in the Cleansing Package group.

Data Cleanse cleansing package option descriptions

Option	Description
Cleansing Package Name	Specifies the cleansing package to use for processing. Enter the name of the cleansing package. The transform uses the location that you set in the *DS Common Dir* option in the *System Group* as the location for the cleansing package.

Option	Description
Content Domain Sequence	Specifies the content domain to use for processing. Select the content domains to include. Use the arrows to change the order of the content domains if necessary. Be sure to add GLOBAL as the last domain in the sequence. Content domains include the following: GLOBAL - Global AR - Arabic ZH - Chinese CS - Czech DA - Danish NL - Dutch EN_US - English (United States & Canada) EN_GB - English (United Kingdom & Ireland) EN_AU - English (Australia & New Zealand) EN_IN - English (India) FR - French DE - German HU - Hungarian ID - Indonesian IT - Italian JA - Japanese MS - Malay NO - Norwegian PL - Polish PT_BR - Portuguese (Brazil) PT_PT - Portuguese (Portugal) RO - Romanian RU - Russian SK - Slovak ES_MX -Spanish (Latin America) ES_ES - Spanish (Spain) SV - Swedish TR - Turkish The Global domain is a special content domain that 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. **i Note** You can set this option as a dynamic input field.

Option	Description
Output Format	Specifies the format for output. Data Cleanse uses certain output fields based on the specified domain in the output format. The transform formats the data in those fields according to the regional standards. **i Note** You can set this option as a dynamic input field. Valid values for this option are: AR Arabic ZH Chinese CS Czech DA Danish NL Dutch EN_US English (United States & Canada) EN_GB English (United Kingdom & Ireland) EN_AU English (Australia & New Zealand) EN_IN English (India) FR French DE German HU Hungarian ID Indonesian IT Italian JA Japanese MS Malay NO Norwegian PL Polish PT_BR Portuguese (Brazil) PT_PT Portuguese (Portugal) RO Romanian RU Russian SK Slovak ES_MX Spanish (Latin America) ES_ES Spanish (Spain) SV Swedish TR Turkish

3.5.6.5 Data Cleanse editor options

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

Option	Description
Filter Output Fields	Specifies which output fields are displayed in the Output tab. *Show_Relevant_Fields:* The fields available in the Output tab are based on the mapped input fields and the selected parser sequence multiline options. Includes all output fields that could possibly contain parsed data. The `<Extra>` fields are always available. *Show_All_Fields:* All Data Cleanse transform fields are available.
Memory in KB for Cache	(Optional) Enter a value of memory to allocate per thread for processing the Data Cleanse transform. Setting this option to **0** indicates that the internal default value (65536) should be used.

3.5.6.6 Data Cleanse Input Word Breaker

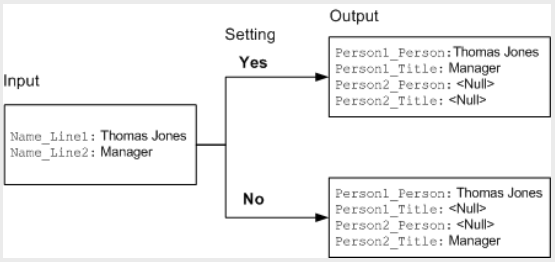
The option in the Input Word Breaker group controls how the parser breaks input data.

Option	Description
Break on Whitespace Only	Specifies whether the Data Cleanse transform breaks input data on white space only or on white space, punctuation, alphanumeric transitions, and script transitions. *YES*: Breaks input data on white space only. *NO*: Breaks input data on white space, punctuation, and alphanumeric transitions, and script transitions. For data in the CJK and Kana scripts, input data breaks on each character. *NO* is the default setting. Enables the Data Cleanse transform to recognize alphanumeric strings, such as product codes, as entries in a custom cleansing package. **Example** **Input data:** AF302 *Break on Whitespace Only* = *NO* • Parser breaks input into two tokens: AF and 302. *Break on Whitespace Only* = *YES* • Parser recognizes input as a single entry AF302 **Note** This option typically applies to custom cleansing packages. The out-of-the-box person and firm cleansing packages are designed to use the parsing strategy that breaks data on white space, punctuation, alphanumeric transitions, and script transitions.

3.5.6.7 Data Cleanse person standardization options

The options in the Person group control how the transform standardizes person-related output.

Person standardization options

Option	Description
Assign Prenames	Specifies whether the transform includes assigned prenames, such as Mr., Mrs., Miss, or Ms., in the *PRENAME* output field. • *YES*: Assigns prenames based on the gender of the value in the *GIVEN_NAME1* field, or on gender option settings. • *NO*: Does not assign prenames. The *PRENAME* output field always includes prenames that are part of the name input data. Additionally, the Data Cleanse transform adds 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, the transform assigns prenames based on the gender option settings in *Use Given Name2 to Assign Gender* and *Use Family Name to Assign Gender.*.
Associate Name Title	Specifies whether the transform should consider data in separate name and occupational title fields as one person. • *YES*: Consider data in separate name and occupational title fields as one person. • *NO*: Do not consider data in separate name and occupational title fields as one person. **Example** The following diagram shows the difference in the output based on the setting you choose for *Associate Name Title*.  **Restriction** If you generate data quality statistics tables in the Data Cleanse transform, set this option to *NO*.

Option	Description
Combine Compound Names	Specifies how the software standardizes compound family names when the family name includes a *PRE_FAMILY_NAME* that is also a *PRE_FAMILY_NAME_COMBINE*. • *YES*: Combines compound family names. For example, the transform combines the family name Mc Donald to McDonald. • *NO*: Preserves the input format of compound family names. For example, the family name Mc Donald remains Mc Donald. **i Note** This option has no impact on a *PRE_FAMILY_NAME* that is not classified as a *PRE_FAMILY_NAME_COMBINE*. **Example** The software does not combine the name Van Helsing into VanHelsing because "Van" is not classified with both *PRE_FAMILY_NAME* and *PRE_FAMILY_NAME_COMBINE*.
Enable Presumptive Name Parsing	Specifies whether to use presumptive name parsing on *NAME_LINE* input fields. • *YES*: Turns on presumptive name parsing. The transform treats data in the *NAME_LINE* input field as a name. • *NO*: Turns off presumptive name parsing. Data in the *NAME_LINE* input field that does not parse as a name remains unparsed and the transform outputs it to the *EXTRA* field. **Example** If data contains an automobile brand and model in a *NAME_LINE* input field, the 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 cannot assign the data, it outputs the unparsed data to the *EXTRA* field. • If the option is set to *YES*, Data Cleanse assigns the data as a name.

Option	Description
Name Order	Defines how the transform applies parsing rules to determine the content of the *GIVEN_NAME* and *FAMILY_NAME* output fields. • *GIVEN_FAMILY_NAME_STRICT*: Specifies the order of given name and family name in the input file. The transform considers only parsing rules that follow the strict name definition order to determine the rule to apply to the input string. Select when the order of given name and family name is consistent in the input data. • *FAMILY_GIVEN_NAME_STRICT*: Specifies the order of given name and family name in the input file. The transform considers only parsing rules that follow the strict name definition order to determine the rule to apply to the input string. Select when the order of family name and given name is consistent in the intput data. • *GIVEN_FAMILY_NAME_SUGGEST*: Specifies the parsing order of given name and family name in the input file. The transform uses this order as a tie breaker when two rules have the same confidence score. • *FAMILY_GIVEN_NAME_SUGGEST*: Specifies the parsing order of family name, given name in the input file. The transform uses this order as a tie breaker when two rules have the same confidence score. • *UNKNOWN*: Specifies to use the order of the rule with the highest confidence score based on information in the dictionary and rule file.

Option	Description
Parse Discrete Input	Specifies how the transform maps discrete person data to output fields. • *NO*: Maps discrete input fields directly to the corresponding output fields without parsing. • *YES*: Combines discrete input fields to one field, parses the data, and outputs data to discrete output fields. **Example** Input data: PERSON1_GIVEN_NAME1 = Mr John T PERSON1_FAMILY_NAME1 = Smith III Output: Option = *NO* PERSON1.PRENAME = *blank* PERSON1.GIVEN_NAME1 = Mr John T PERSON1.GIVEN_NAME2 = *blank* PERSON1.FAMILY_NAME1 = Smith III PERSON1.MATURITY_POSTNAME = *blank* Output: Option = *YES* PERSON1.PRENAME = Mr PERSON1.GIVEN_NAME1 = John PERSON1.GIVEN_NAME2 = T PERSON1.FAMILY_NAME1 = Smith PERSON1.MATURITY_POSTNAME = III

3.5.6.8 Data Cleanse gender options

The gender options control what input fields Data Cleanse uses to assign gender.

Find gender standardization options 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)	Specifies how the transform assigns gender when the input gender in *PRENAME* or *GIVEN_NAME1* is unassigned or ambiguous. *YES*: Assigns ambiguous gender data using the gender of the parsed *PRENAME* or *GIVEN_NAME2* fields. *NO*: Does not assign gender data using the gender of the parsed *PRENAME* or *GIVEN_NAME2* field. **Example** **Input:** Pat Robert Smith **Parsed:** • *GIVEN_NAME1* = Pat • *GIVEN_NAME2* = Robert *Use Given Name2 To Assign Gender (Gender Options)* = *YES*. **Gender:** *GIVEN_NAME1* is ambiguous. Therefore, Data Cleanse uses the data in *GIVEN_NAME2*, which is not ambiguous, to determine that the gender is *STRONG_MALE*.

Option	Description

Use Family Name To Assign Gender (Gender Options)	Specifies how the transform assigns gender when the input gender in *PRENAME*, *GIVEN_NAME1*, or *GIVEN_NAME2* is unassigned or ambiguous.
	• *YES*: Assigns gender based on the parsed *FAMILY_NAME1* or *FAMILY_NAME2* fields when it can't determine gender using the *PRENAME* or *GIVEN_NAME1 GIVEN_NAME2* fields.
	  • Uses *FAMILY_NAME1* when gender is assigned and is not ambiguous.
	  • Uses *FAMILY_NAME2* when unable to use *FAMILY_NAME1*.
	• *NO*: Does not assign gender of the parsed *FAMILY_NAME1* or *FAMILY_NAME2* fields.
	Example
	Input data: N. Albiantsev
	Parsed: Given_Name1 = N.
	Use Family Name To Assign Gender (Gender Options) = *NO*
	Results: No gender assigned because *GIVEN_NAME1* is ambiguous.
	Parsed: Family_Name1 = Albiantsev
	Use Family Name To Assign Gender (Gender Options) = *YES*
	Results: Assigned gender is *STRONG_MALE* because value of *FAMILY_NAME1*, Albiantsev, is *STRONG_MALE*

3.5.6.9 Data Cleanse 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	Specifies whether the transform uses presumptive firm parsing on *FIRM_LINE* input fields. • *YES*: Uses presumptive firm parsing. Assigns data in the *FIRM_LINE* input field as firm data regardless of the content. • *NO*: Does not use presumptive firm parsing. Assigns data in the *FIRM_LINE* input field as firm data. If it cannot assign data as firm data, outputs unparsed data to the *EXTRA* field. Presumptive firm parsing tells the transform to process any information in a Firm field as a firm, regardless of the type of information in the field. **Example** If your input 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 *Enable Presumptive Firm Parsing* is set to *NO* and Data Cleanse is not able to assign the data, it outputs the unparsed data to the *EXTRA* field. • If the option is set to *YES*, Data Cleanse assigns the data as a firm.

3.5.6.10 Data Cleanse Other Standardization options

To control how the Data Cleanse transform standardizes other data types in your input data, set options in the Standardization group.

Option	Description
Capitalization	Specifies the casing of your output. • *LOWER*: Converts the output to lowercase. For example, john mckay. • *MIXED*: Preserves the casing for the standard form as defined within the cleansing package. If a standard form is not defined, the output is converted to mixed case. **Example** If you define the standard form as JOHN MCKAY, the transform preserves the form. If you do not define a standard form, the transform converts the output to mixed case as in John McKay. • *PRESERVE*: Preserves the input casing. Input is JOHN MCKAY, stays as JOHN MCKAY. • *UPPER*: Converts the output to uppercase. For example, converts john mckay to JOHN MCKAY.

Option	Description
Character Width Style	Specifies the character width used in output fields. Useful when you process Japanese or mixed language data. • *NORMAL_WIDTH*: Output field width reflects the normalized character width based on the script type. Some output columns contain halfwidth characters and other columns contain fullwidth characters. *NORMAL_WIDTH* does not require special processing and therefore is the most efficient setting. **Example** For *NORMAL_WIDTH*, the transform standardizes all fullwidth Latin characters to their halfwidth forms. The transform standardizes all halfwidth katakana characters to their fullwidth forms. • *FULL_WIDTH*: Converts characters from halfwidth form to fullwidth form in all output fields. If characters do not have fullwidth forms, leaves characters in halfwidth form. • *HALF_WIDTH*: Converts characters from fullwidth forms to halfwidth forms in all output fields. If characters do not have halfwidth forms, leaves the characters in fullwidth form. **→ Tip** Since the output width is based on the normalized width for the character type, the output data may be larger than the input data. Consider increasing the column width in the target table. For template tables, selecting *Use NVARCHAR for VARCHAR columns in supported databases* changes the VARCHAR column type to NVARCHAR and allows for increased data size.

Option	Description
One-to-One Mapping	Specifies whether the transform places the input data into the corresponding output field for the following parsers: Phone, Email, Date. • *YES*: Places the parsed data into the corresponding output field. • *NO*: Places the parsed data into the first available output field in the category. **Example** **Input:** Date1 = *blank* Date2 = *blank* Date3 = 10/2018 Output when you set *One-to-one mapping* to *YES*: The output is the same as input. Output when you set *One-to-one mapping* to *NO*: Date1 = 10/2018 Date2 = *blank* Date3 = *blank*
Remove Diacritical Characters	Specifies whether the software removes diacritical characters and replaces them with the ASCII equivalents. • *YES*: Replaces diacritical characters such as accent marks and umlauts, with the ASCII equivalent. • *NO*: Standardizes and retains diacritical characters. **Example** *Remove Diacritical Characters* = *NO* • **Input:** María Hernández • **Output:** María Hernández *Remove Diacritical Characters* = *YES* • **Input:** María Hernández • **Output:** Maria Hernandez

Option	Description
Remove Punctuation	Specifies whether the transform removes all punctuation from standardized data, with the exception of hyphens between names. • *YES*: Removes punctuation except for hyphens between names. • *NO*: Does not remove punctuation. **Example** *Remove Punctuation* = *YES* • **Input:** X.L. • **Output:** XL *Remove Punctuation* = *NO* • **Input:** X.L. • **Output:** X.L.
Ssn Delimiter	Specifies the character to use for standard U.S. Social Security number (SSN) output delimiters. • *BACKSLASH*: Uses backward slashes as the delimiter in the SSN. For example, 799\45\6789. • *DASH*: Uses dashes as the delimiter in the SSN. For example, 799-45-6789. • *SLASH*: Uses forward slashes as the delimiter in the SSN. For example, 799/45/6789. • *NONE*: Does not add a delimiter to the SSN. For example, 799456789. • *PERIOD*: Uses periods as the delimiter in the SSN. For example, 799.45.6789. • *SPACE*: Uses spaces as the delimiter in the SSN. For example, 799 45 6789.

3.5.6.11 Data Cleanse Date Options

To control how the Data Cleanse transform standardizes date data, set options in the Date Options group.

Date option descriptions

Option group	Description
Century Threshold	Specifies a number for which the transform considers a two-digit date as part of the 20th or 21st century. 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). **Example** *Century Threshold* = 11: • Input = 8 • Output = 2008 • Input = 11 • Output = 2011 • Input = 73 Output = 1973

Option group	Description

Date Delimiter	Specifies what character to use for standard date output delimiters.
	• *BACKSLASH*: Uses backward slashes as the delimiter for the date. For example, 04\01\2010.
	• *DASH*: Uses dashes as the delimiter for the date. For example, 04-01-2010.
	• *SLASH*: Uses forward slashes as the delimiter for the date. For example, 04/01/2010.
	• *NONE*: Does not add a delimiter to the date. For example, 04012010
	• *PERIOD*: Uses periods as the delimiter for the date. For example, 04.01.2010.
	• *SPACE*: Uses spaces as the delimiter for the date. For example, 04 01 2010.
	• *CHINESE_JAPANESE*: Uses the following Chinese or Japanese characters as delimiters:
	  • 月 always follows the month
	  • 日 always follows the day
	  • 年 always follows the year
	Example
	Arabic numbers with Chinese or Japanese delimiters: 04 月 01 日 2010 年
	Example
	Chinese or Japanese Numbers with Chinese or Japanese delimiters: 四 月 一 日 二千零一 十 年

Date Format	Specifies how the transform standardizes dates on output. • *YEAR_MONTH_DAY*: For Example, August 16, 2012 is 2012-08-16 • *YEAR_DAY_MONTH*: For Example, August 16, 2012 is 2012-16-08 • *MONTH_DAY_YEAR*: For Example, August 16, 2012 is 08-16-2012 • *DAY_MONTH_YEAR*: For Example, August 16, 2012 is 16-08-2012
Enable Zero Pad	Specifies whether the transform places a zero in front of one-digit days and months on input. • *YES*: Places a zero in front of one-digit day and month data. For example, outputs July 4 as 07/04. • *NO*: Does not place a zero in front of one-digit day and month date. For example, outputs July 4 as 7/4.

Option group	Description
Input Month Before Day	Specifies whether input dates use a pattern with the month listed first. • *YES*: Lists the month first in a date string. For example, 11/12/2004 is November 12, 2004. • *NO*: Does not list the month first in a date string.
Input Year First	Specifies whether input dates use a pattern with the year listed first. • *YES*: Lists the year first in a date string. For example, 03/02/04 is 2003 February 4. • *NO*: Does not list the year first in a date string.
Month Format	Specifies how the transform standardizes month output components in a date output. • *FULL_TEXT*: Outputs month component using the full form of the spelled word. For example, outputs Aug as August. Outputs 12 as December. • *NUMERIC*: Outputs month component using the numeric form of the month. For example, outputs September as 9. Outputs December as 12. • *SHORT_TEXT*: Outputs month component using the standard abbreviation of the month. For example, outputs December as Dec Outputs March as Mar.
Numeric Format	Specifies what numeric format the transform uses for date values. • *ARABIC_NUMBERS*: Returns numeric date values in Arabic. • *CHINESE_JAPANESE_NUMBERS*: Returns numeric date values in Chinese or Japanese.
Year Format	Specifies how the transform standardizes year components in a date output. • *FULL*: Outputs the year using four digits. For example, outputs 04 as 2004. • *SHORT*: Outputs the year using 2 digits. For example, outputs 2004 as 04.

3.5.6.12 Data Cleanse Phone Options group

To control how the Data Cleanse transform standardizes phone data on output, set options in the Phone Options group.

> **i Note**
>
> Reference to “country” or “countries” in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

Phone Options group descriptions

Option	Description
ISO2 Country Code Sequence	Specifies a sequence of countries using the ISO two-character code. Enter the two-character ISO country code for each country in the search. Separate the country codes with a pipe character. • The codes you include determine the countries in which the transform searches for phone matches. • The order in which you place the codes in the sequence determines the order in which Data Cleanse searches for phone information. *GLOBAL* is the default value. It's optional. *GLOBAL* contains the international regular expressions that are set in the cleansing package. If you include *GLOBAL* in your sequence, place it last in the sequence. **i Note** Even if you place *GLOBAL* first in the sequence, the transform always searches *GLOBAL* last. **Example** If most of your data contains records from Germany and Australia, make sure that the ISO2 code for Germany comes first, followed by Australia: *DE|AU*. If you're sure that there are no other countries represented in your data, exclude *GLOBAL* to save processing time. If you don't know what countries are represented in your data, include only *GLOBAL* in your sequence. If you know that some countries are represented in your data, but not all countries, include the countries that you're sure of and then include Global: *DE|AU|GLOBAL*. For this example sequence, the transform searches for phone matches in the following way: • Searches Germany phone data to find a match. • If the data doesn't parse as German phone data, the transform searches Australia phone data to find a match. • If the data doesn't match either Germany or Australia phone data, the transform searches the global phone data.

Option	Description
	• If there's still no match, the transform searches for a match in the North American Phone Parens Area Plan (NANP).
North American Phone Parens Area	Specifies whether the transform includes parentheses around the area code of phone numbers belonging to the NANP countries. • *YES*: Includes the parentheses. For example, (123) 555-1234. • *NO*: Omits the parentheses. For example, 123 555-1234.
North American Phone Delimiter After Area	Specifies whether the transform follows the rule for delimiter placement after the area code following the NANP. To use this option, also set a delimiter in the *North American Phone Delimiter* option. • *YES*: Includes the specified delimiter after the area code in the phone data output. For example, 123-555-1234. • *NO*: Doesn't include a delimiter after the area code in the phone data output. For example, 123 555-1234.
North American Phone Delimiter	Specifies what character the transform uses to delimit phone output following the NANP. **i Note** To use this option, set the *North American Phone Delimiter After Area* to *YES*. • *BACKSLASH*: Uses backward slashes as the delimiter in the phone number. For example, 123\555\1234. • *DASH*: Uses dashes as the delimiter in the phone number. For example, 123-555-1234. • *SLASH*: Uses forward slashes as the delimiter in the phone number. For example, 123/555/1234. • *NONE*: Doesn't add a delimiter to the phone number. For example, 1235551234. • *PERIOD*: Uses periods as the delimiter in the phone number. For example, 123.555.1234. • *SPACE*: Uses spaces as the delimiter in the phone number. For example, 123 555 1234.
Phone Extension Text	Specifies the standard text the transform outputs as the phone extension indicator. For example, Ext.

3.5.6.13 Data Cleanse Parser configuration

Parser configuration options control the parsing engines that the Data Cleanse transform uses for parsing multiline fields and the order in which they are applied.

If you do not include a particular parser, Data Cleanse does not look for that type of data in the input field.

Data Cleanse Parser options

Option	Description
Parser Sequence Multiline1 *Parser Sequence Multiline2* *Parser Sequence Multiline3* *Parser Sequence Multiline4* *Parser Sequence Multiline5* *Parser Sequence Multiline6* *Parser Sequence Multiline7* *Parser Sequence Multiline8* *Parser Sequence Multiline9* *Parser Sequence Multiline10* *Parser Sequence Multiline11* *Parser Sequence Multiline12*	Specifies the parser or parsers the transform uses, and the order in which to use them. • *ADDRESS*: Parses data as an address. • *DATE*: Parses data as a date. • *EMAIL*: Parses data as an e-mail address. • *FIRM*: Parses data as a firm name. • *LAST_LINE*: Parses data as a last line. • *PERSON*: Parses data as a person name. • *PERSON_OR_FIRM*: Parses data as a person or a firm name. • *PHONE*: Parses data as a phone number. **i Note** The transform parses any non-North American numbers first. Phone numbers may be output to the *PHONE* and *INTERNATIONAL_PHONE* fields. The transform parses North American numbers later using the North American Numbering Plan (NANP). The data may be output to *PHONE* and *NORTH_AMERICAN_PHONE* output fields. Define phone number patterns that do not follow the NANP in Cleansing Package Builder. • *SSN*: Parses data as U.S. Social Security number. • *UDPM*: Parses data using user-defined patterns created in Cleansing Package Builder. **i Note** Order is important. To order your parsers, enter the parsers separated with a pipe character in the applicable order in which the transform should parse your data.

3.5.6.14 Data Cleanse input fields

Map fields from upstream objects to Data Cleanse input fields so the transform knows the type of data to expect on input.

> **i Note**
>
> Reference to "country" or "countries" in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

The following table contains an alphabetical list of the recognized input fields to use in the input mapping for the Data Cleanse transform.

Name	Description
DATA_SOURCE_ID	Specifies the name of the data source that the record comes from. **→ Tip** If you choose to generate data quality statistics tables, use this field as part of your mapping strategy, along with primary key fields and the output field *ROW_ID*. This mapping identifies the source of individual records in any of the non-summary data quality statistics tables, thus enabling you to trace problem data to the data source. This input field is applicable for the following transforms: • Data Cleanse • Global Address Cleanse • Geocoder
DATE1-6	Contains date data. For example, 08/16/2004. Use up to six date fields for mapping multiple dates.
EMAIL1-6	Contains e-mail address data. Use up to six e-mail fields for mapping multiple e-mail addresses.
FIRM_LINE1-6	Contains firm name, firm location, or both. Use up to six firm lines for mapping multiple firm names.
FIRM_LOCATION1-2	Contains location within a company or organization, such as a department, mail stop, room, or building. Use up to two firm location fields.

Name	Description
FIRM_NAME1-2	Contains the name of a company or organization. Use up to two firm name fields.
MULTILINE1-12	Contains multiline data. The transform parses data from this input field in the order that you set in the *Parser Sequence Multiline* option, including parsers from custom cleansing packages. Use up to 12 multiline fields.
NAME_LINE1-6	Contains a whole name such as first and last name, or multiple whole names. Can include job title. Use up to six name line fields.
NAME_OR_FIRM_LINE1-6	Contains the name of a person or organization. Use up to six name or firm line fields.
OPTION_COUNTRY (Dynamic input field)	Contains an ISO2 country code to help parse phone data and to determine the content domain sequence. Obtain ISO2 country code data either from an input file that contains the codes, or from an upstream transform such as Global Address Cleanse that is set up to output ISO2 codes. **For phone parsing:** Data Cleanse uses the ISO2 country code to help determine the country code for parsing phone data when: • The transform populates the *OPTION_COUNTRY* dynamic input field with an ISO2 country code. • The country is included in the cleansing package. Map the *ISO_COUNTRY_CODE_2CHAR* field from the Global Address Cleanse transform to the *OPTION_COUNTRY* input field. **For setting domain sequence:** Data Cleanse uses the ISO2 code to determine content domain sequence when the *OPTION_COUNTRY* dynamic input field is populated with an ISO2 code. The transform automatically generates the output format based on the ISO2 country code and values in the *OPTION_LANGUAGE* and *OPTION_REGION* fields.

Name	Description
OPTION_LANGUAGE *OPTION_REGION* (Dynamic input fields)	Contains the language and region. The transform uses the values in *OPTION_LANGUAGE* and *OPTION_REGION*, along with the value in the *OPTION_COUNTRY* field, to determine the content domain and output format. There are a few countries where *OPTION_LANGUAGE* and *OPTION_REGION* data is helpful to determine the most appropriate content domain and output format. The two fields are applicable only to certain countries. For example, use *OPTION_LANGUAGE* and *OPTION_REGION* for Switzerland, Belgium, or Canada. The transform uses the *OPTION_LANGUAGE* and *OPTION_REGION* fields only to determine the most appropriate content domain and output format. Map the *OPTION_COUNTRY*, *OPTION_LANGUAGE*, and *OPTION_REGION* input fields from the following Global Address Cleanse output fields in this order: • *ISO_COUNTRY_CODE_2CHAR* • *LANGUAGE* • *REGION1* These input fields work together to determine the *OPTION_CONTENT_DOMAIN_SEQUENCE* and-or *OPTION_OUTPUT_FORMAT*.
OPTION_CONTENT_DOMAIN_SEQUENCE (Dynamic input field)	Contains the content domain sequence. The valid predefined values are: AR, CS, DA, DE, EN_AU, EN_GB, EN_IN, EN_US, ES_MX, ES_ES, FR, HU, ID, IT, JA, MS, NL, NO, PL, PT_BR, PT_PT, RO, RU, SK, SV, ZH, and GLOBAL.
OPTION_OUTPUT_FORMAT (Dynamic input field)	Contains the format for output specified as an abbreviation of the domain. The valid predefined values are: AR, CS, DA, DE, EN_AU, EN_GB, EN_IN, EN_US, ES_MX, ES_ES, FR, HU, ID, IT, JA, MS, NL, NO, PL, PT_BR, PT_PT, RO, RU, SK, SV, and ZH.
PERSON1_FAMILY_NAME1 *PERSON2_FAMILY_NAME1*	Contains a discrete family name or last name such as Smith. Map up to two family names.

Name	Description
PERSON1_FAMILY_NAME2 *PERSON2_FAMILY_NAME2*	Contains a second discrete family name. Can be useful for cultures where people are known by both paternal and maternal family names. If your input data contains two family names, map the first to *PERSON1_FAMILY_NAME2* and the second to *PERSON2_FAMILY_NAME2*.
PERSON1_GIVEN_NAME1-2 *PERSON2_GIVEN_NAME1-2*	Contains discrete given names, for example, "John" or "B". Map up to two given names for person 1, and map up to two given names for person 2.
PERSON1_HONORARY_POSTNAME *PERSON2_HONORARY_POSTNAME*	Contains an honorary postname indicating certification, academic degree, or affiliation, such as CPA. Map an honorary postname for up to two people.
PERSON1_MATURITY_POSTNAME *PERSON2_MATURITY_POSTNAME*	Contains a maturity postname indicating heritage, such as Jr., Sr., III. Map maturity postnames for up to two people.
PERSON1_PRENAME *PERSON2_PRENAME*	Contains a discrete prename, such as Mr., Mrs., Dr., or Lt. Col. Map a discrete prename for up to two people.
PERSON1_TITLE *PERSON2_TITLE*	Contains a discrete job title, such as Accountant or Software Engineer. Map a discrete job title for up to two people.
PHONE1-6	Contains a phone number. Map up to six phone numbers.
SSN1-6	Contains a U.S. Social Security Number. Map up to six Social Security Numbers.
TITLE_LINE1-6	Contains a job title, such as Accountant, or Assistant. Map up to six title lines.
UDPM1-4	Contains information associated with patterns and rules that you define in the user-defined type of reference Data in the Cleansing Package Builder utility. Map up to four user-defined fields.

Related Information

Domains
Dynamic transform settings [page 244]

3.5.6.15 Data Cleanse output fields

Map Data Cleanse output fields to output Standardized and corrected data, generated data, and additional data.

The following table contains an alphabetical list of recognized output fields that you use in output mapping for the Data Cleanse transform. By default, the *EXTRA*, *INFO_CODE*, and *STATUS_CODE* fields are always available. Map other fields based on the mapped input fields and the selected parser sequence multiline options. Additionally, the *IGNORE* field may be available.

You can use the *Filter Output Fields* option to display a complete list of output fields.

Field name	Description	FIELD_CLASS	Content Type
DATE	Contains a date.	Parsed Standardized	Date
DATE_DAY	Contains the day that the transform parses from the date.	Parsed Standardized	Date
DATE_MONTH	Contains the month that the transform parses from the date.	Parsed Standardized	Date
DATE_YEAR	Contains the year that the transform parses from the date.	Parsed Standardized	Date
DUAL_NAME	Contains name components from an input string that contains two names separated by a connecting word such as "and" or "or." **Example** **Input:** Terry and Kris Johnson **Output:** Terry Johnson and Kris Johnson **Example** **Input:** Terry Johnson or Kris Adams **Output:** Terry Johnson or Kris Adams	Parsed Standardized	None
EMAIL	Contains a full e-mail address.	Parsed Standardized	Email

Field name	Description	FIELD_CLASS	Content Type
EMAIL_DOMAIN_ALL	Contains the domain portion of an e-mail address. The domain portion includes all information after the @ symbol. For example, in "joex@sap.com", “sap.com” is the domain.	Parsed Standardized	Email
EMAIL_DOMAIN_FIFTH	Contains the Parsed fifth to last domain from an e-mail address with multiple domains listed.	Parsed Standardized	Email
EMAIL_DOMAIN_FOURTH	Contains the Parsed fourth to last domain from an e-mail address with multiple domains listed.	Parsed Standardized	Email
EMAIL_DOMAIN_HOST	Contains the host portion of an e-mail address. The host portion is the first string after the @ symbol. For example, in "joex@sap.com", "sap" is the host.	Parsed Standardized	Email
EMAIL_DOMAIN_SECOND	Contains the Parsed second to last domain in an e-mail address with multiple domains listed.	Parsed Standardized	Email
EMAIL_DOMAIN_THIRD	Contains the Parsed third to last domain in an e-mail address with multiple domains listed.	Parsed Standardized	Email
EMAIL_DOMAIN_TOP	Contains the last domain in an e-mail address with multiple domains listed. For example, in jsmith@one.two.three.com, “.com” is the last domain.	Parsed Standardized	Email
EMAIL_IS_ISP	Indicates whether the e-mail address is a known ISP (internet service provider) or e-mail domain name listed in the e-mail data type of Reference Data in Cleansing Package Builder.	Standardized	Email

Field name	Description	FIELD_CLASS	Content Type
EMAIL_USER	Contains the user name of the e-mail address. For example, in "joex@sap.com", "joex" is returned.	Parsed Standardized	Email
EXTRA	Contains data that the transform does not parse using any of the active parsers, or that the transform does not recognize as applicable for one of the other output fields.	Parsed	None
FAMILY_NAME1	Contains the family name portion of a name. For example, Smith.	Parsed Standardized	Family_Name
FAMILY_NAME1_MATCH_STD1-6	Contains the match standard for family names. Use this field only with cleansing packages that include name data in more than one script. The match standards include the name as it is written in alternate script types. For example, for a family name included in the Japanese dictionary in Kanji script, the match standards include Kana renditions of the name. If the dictionary does not have an alias entry, the field is empty. Compare with *MATCH_FAMILY_NAME*.	Standardized	Family_Name1_ Match_Std
FAMILY_NAME2	Contains the second family name. Use to output the paternal and maternal family names to separate fields.	Parsed Standardized	Family_Name 2

Field name	Description	FIELD_CLASS	Content Type
FAMILY_NAME2_MATCH_STD1-6	Contains the match standard for second family names. Use this field only with cleansing packages that include name data in more than one script. The match standards include the name as it is written in alternate script types. For example, for a family name included in the Japanese dictionary in Kanji script, the match standards include Kana renditions of the name. If the dictionary does not have an alias entry, the field is empty. Compare with *MATCH_FAMILY_NAME*.	Standardized	Family_Name2_ Match_Std
FAMILY_NAME_FULL	Contains a combination of the *FAMILY_NAME1* and *FAMILY_NAME2* fields. **Example** *FAMILY_NAME1* = Smith *FAMILY_NAME2* = Jones *FAMILY_NAME_FULL* = Smith Jones	Parsed Standardized	Family_Name_Full
FIRM	Contains the firm, company, or organization name.	Parsed Standardized	Firm
FIRM_LOCATION	Contains a location within a company or organization, such as a department. For example, Mailstop.	Parsed Standardized	Firm_Location

Field name	Description	FIELD_CLASS	Content Type
FIRM_MATCH_STD1-6	Contains the match standard for the firm, company, or organization name. For example, HP is the match standard or alias for Hewlett Packard. If the cleansing package does not have an alias entry, the output field is empty. Compare with *MATCH_FIRM*.	Standardized	Firm_Match_Std
FIRM_LOCATION_MATCH_STD1-6	Contains the match standard for firm locations. For example, MS is the match standard or alias for mailstop. If the dictionary does not have an alias entry, the transform outputs an empty field.	Standardized	Firm_Location_ Match_Std

Field name	Description	FIELD_CLASS	Content Type
GENDER	Contains the gender description. The gender description can be one of the following values: • *AMBIGUOUS*: The name does not reliably indicate a gender. The name could be either male or female. For example, Pat. • *MALE_STRONG*: High confidence that the person is male. That is, the name belongs to someone who is almost certainly a male. For example, John. • *MALE_WEAK*: Some confidence that the person is male. That is, the name belongs to someone who is probably male. For example, Terry. • *FEMALE_STRONG*: High confidence that the person is female. That is, the name belongs to someone who is almost certainly a female. For example, Mary. • *FEMALE_WEAK*: Some confidence that the person is female. That is, the name belongs to someone who is probably a female. For example, Lynn. For dual names, the following output is also available: • *MULTI_NAMES_AMBIGUOUS*: At least one of the names does not reliably	Standardized	None

Field name	Description	FIELD_CLASS	Content Type
	indicate a gender. For example, Pat and John. • *MULTI_NAMES_FEMALE*: Some or high confidence that both of the names belong to people who are female. For example, Mary and Lynn. • *MULTI_NAMES_MALE*: Some or high confidence that both of the names belong to people who are male. For example, John and Terry. • *MULTI_NAMES_MIXED*: Some or high confidence that one of the names belongs to a person who is female, and the other name belongs to a person who is male. For example, Lynn and John.		
GENDER_ID	A numeric value that corresponds to the gender description: • *0*: Unassigned • *1*: Male_Strong • *2*: Male_Weak • *3*: Ambiguous • *4*: Female_Weak • *5*: Female_Strong • *6*: Multi_Names_Mixed • *7*: Multi_Names_Male • *8*: Multi_Names_Female • *9*: Multi_Names_Ambiguous	Standardized	None
GIVEN_NAME1	Contains the given name, for example, Robert.	Parsed Standardized	Given_Name

Field name	Description	FIELD_CLASS	Content Type
GIVEN_NAME1_MATCH_STD 1-6	Contains the match standard for given names. For example, the transform can tell you that Patrick and Patricia are potential matches for the given name Pat. Match standards help you overcome two types of matching problems: • Alternate spellings such as Catherine and Katherine • Nicknames such as Pat and Patty. Compare with *MATCH_GIVEN_NAME1*.	Standardized	Given_Name1_ Match_Std
GIVEN_NAME2	Contains the second given name.	Parsed Standardized	Given_Name2
GIVEN_NAME2_MATCH_STD 1-6	Contains the match standard for second given names. For example, the transform can tell you that Patrick and Patricia are potential matches for the given name Pat. Compare with *MATCH_GIVEN_NAME2*.	Standardized	Given_Name2_ Match_Std
GIVEN_NAME_FULL	Contains the combined values of *GIVEN_NAME1* and *GIVEN_NAME2*. Example *GIVEN_NAME1* = Jonathan *GIVEN_NAME2* = Peter *GIVEN_NAME_FULL* = Jonathan Peter	Parsed Standardized	Given_Name_Full

Field name	Description	FIELD_CLASS	Content Type
HONORARY_POSTNAME	Contains the honorary postname. Honorary postname indicates certification, academic degree, or affiliation. For example, CPA.	Parsed Standardized	Postname
HONORARY_POSTNAME_MATCH_STD1-6	Contains the match standard for 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 field is empty.	Standardized	Postname_Match_Std
INFO_CODE	Contains the code that identifies the rows that may require manual review because the data is suspect.	Assignment_Info	None
INTERNATIONAL_PHONE	Contains the entire international phone number, including extra items such as the country code.	Parsed Standardized	Phone
INTERNATIONAL_PHONE_COUNTRY_CODE	Contains the country code of an international phone number.	Parsed Standardized	Phone
INTERNATIONAL_PHONE_COUNTRY_NAME	Contains the name of the country of origin of an international phone number.	Parsed Standardized	Phone
INTERNATIONAL_PHONE_LINE	Contains the portion of the international phone number that is not the country code or the city code.	Parsed Standardized	Phone
INTERNATIONAL_LOCALITY_CODE	Contains the locality code of the international phone number.	Parsed Standardized	Phone

Field name	Description	FIELD_CLASS	Content Type
MATCH_FAMILY_NAME	Contains the combined Standardized form of *FAMILY_NAME1* and *FAMILY_NAME2* that the software used in the Match transform during the comparison process. The transform separates the two fields with a space. The transform outputs data in uppercase, removes apostrophes, and replaces other punctuation with a single space. Removes pre family name data. Compare with *FAMILY_NAME1_MATCH_STD1-6* and *FAMILY_NAME2_MATCH_STD1-6*.	Standardized	Family_Name1
MATCH_FIRM	Contains a form of Firm that may be used in the Match transform during the comparison process. The transform outputs data in uppercase, removes apostrophes, and replaces other punctuation with a single space, and removes data that is extraneous for matching purposes. This extraneous data includes business types such as Ltd. and GambH, and noise words such as The, And, and Of. **i Note** Some words are classified to be removed from all domains, while others are language-specific and are classified to be removed in specific cultural domains. Compare with *FIRM_MATCH_STD1-6*.	Standardized	Firm

Field name	Description	FIELD_CLASS	Content Type
MATCH_GIVEN_NAME1	Contains the Standardized form of *GIVEN_NAME1* used in the Match transform during the comparison process. The transform outputs data in uppercase, removes apostrophes, and replaces other punctuation with a single space. Removes pre given name data. Compare with *GIVEN_NAME1_MATCH_STD 1-6*.	Standardized	Given_Name1
MATCH_GIVEN_NAME2	Contains the Standardized form of *GIVEN_NAME2* used in the Match transform during the comparison process. The transform outputs data in uppercase, removes apostrophes, and replaces other punctuation with a single space. Removes pre given name data. Compare with *GIVEN_NAME1_MATCH_STD 1-6*.	Standardized	Given_Name2
MATCH_MATURITY_POSTNAME	Contains the Standardized form of *MATURITY_POSTNAME* used in the Match transform during the comparison process. The transform outputs data in uppercase, removes apostrophes, and replaces other punctuation with a single space. Compare with *MATURITY_POSTNAME_MATCH_STD1-6*.	Standardized	Postname

Field name	Description	FIELD_CLASS	Content Type
MATCH_PERSON	Contains a form of *PERSON* used in the Match transform during the comparison process. The transform outputs data in uppercase, removes apostrophes, and replaces other punctuation with a single space, and removes data that is extraneous for matching purposes. Extraneous data includes pre-given name, pre-family name, and pre-name as well as honorary and maturity post names and name designators.	Standardized	Person
MATCH_PHONE	Contains a form of *PHONE* used in the Match transform during the comparison process. Data is output as a string of digits. The transform removes spaces, punctuation, alphabetical characters, and leading zeros.	Standardized	Phone
MATCH_PRENAME	Contains the Standardized form of *PRENAME* used in the Match transform during the comparison process. The transform outputs data in uppercase, removes apostrophes, and replaces other punctuation with spaces. Compare with *PRENAME_MATCH_STD1-6*.	Standardized	Prename
MATURITY_POSTNAME	Contains the *MATURITY_POSTNAME* that indicates heritage, such as Jr., Sr., III.	Standardized	Postname
MATURITY_POSTNAME_MATCH_STD1-6	Contains the match standard for *MATURITY_POSTNAME*. For example, Sr. is a match standard or alias for Senior. If the cleansing package does not have an alias entry, the field is empty.	Standardized	Postname_Match_Std
NAME_CONNECTOR	Contains the connector component of a dual name. For example, “and” or “or.”	Parsed Standardized	None

Field name	Description	FIELD_CLASS	Content Type
NAME_DESIGNATOR	Contains the name designator such as “Attn” or “c/o”.	Parsed Standardized	None
NAME_SPECIAL	Contains the generic description of a person such as “occupant” or “current resident”.	Parsed Standardized	None
NORTH_AMERICAN_PHONE	Contains an entire North American Numbering Plan (NANP) phone number.	Parsed Standardized	Phone
NORTH_AMERICAN_PHONE_AREA_CODE	Contains the area code Parsed from the phone number.	Parsed Standardized	Phone
NORTH_AMERICAN_PHONE_EXTENSION	Contains the extension Parsed from the phone number.	Parsed Standardized	Phone
NORTH_AMERICAN_PHONE_LINE	Contains the last four numbers (excluding the extension) Parsed from a phone number. For example, in (123) 456-7890, the transform returns 7890.	Parsed Standardized	Phone
NORTH_AMERICAN_PHONE_PREFIX	Contains the middle three numbers Parsed from a phone number. For example, in (123) 456-7890, the transform returns 456.	Parsed Standardized	Phone
NORTH_AMERICAN_PHONE_TYPE	Contains the type of phone number Parsed from the input phone number. For example, “Home” or “Work.”	Parsed Standardized	Phone
PERSON	Contains combined components that define a single person. For example, Thomas Williams-Doyle Sr., M.D.	Parsed Standardized	None

Field name	Description	FIELD_CLASS	Content Type
PHONE	Contains the phone number that the transform identified as either North American or International. **i Note** Data Cleanse standardizes North American phone numbers but it does not standardize or enhance international phone numbers. Data Cleanse outputs international phone numbers as they are input.	Parsed Standardized	Phone
PRENAME	Contains the prename such as Mr. or Mrs.	Parsed Standardized	Prename
PRENAME_MATCH_STD1-6	Contains the match standard for a prename. For example, Mr. is the match standard or alias for Mister. If the cleansing package does not have an alias entry, the field is empty. Compare with *MATCH_PRENAME*.	Standardized	Prename_Match_Std

Field name	Description	FIELD_CLASS	Content Type
ROW_ID	Contains an internal ID generated by the transform that uniquely identifies a row processed by that transform. Use this field to join the input and output to the applicable non-summary data quality statistics table. If you select to generate any data quality statistics tables in the transform, and you don't include the *ROW_ID* output field, the software issues a warning. If you map this output field and don't select to generate any non-summary data cleanse statistics tables in the transform, the software also issues a warning. Applicable for the following transforms: • Data Cleanse • Global Address Cleanse • Geocoder	Standardized	Statistics
RULE_LABEL	Contains the rule that the transform uses to parse the indicated field.	Standardized	None
SCORE	Contains the confidence score for the Parsed field.	Standardized	None
SSN	Contains the entire Social Security number.	Parsed Standardized	SSN
SSN_AREA	Contains the first three numbers of the Social Security number.	Parsed Standardized	SSN
SSN_GROUP	Contains the fourth and fifth numbers of a Social Security number.	Parsed Standardized	SSN
SSN_SERIAL	Contains the last four numbers in a Social Security number.	Parsed Standardized	SSN

Field name	Description	FIELD_CLASS	Content Type
STATUS_CODE	Contains the code describing the following: • When the job is set for standardization, the code indicates how the transform Standardized the data. • When the job is not set for standardization, the code indicates how the transform could standardize the data.	Assignment_Info	None
TITLE	Contains the job or occupational title of a person, such as Manager or President.	Parsed Standardized	Title
TITLE_MATCH_STD1-6	Contains the match standard for occupational title. For example, CFO is the match standard or alias for Chief Financial Officer. If the dictionary does not have an alias entry, the field is empty.	Standardized	Title_Match_Std
UDPM	Contains the attribute field defined in the user-defined pattern rules in Cleansing Package Builder Reference Data.	Standardized	None
UDPM_SUBCOMPONENT1-5	Contains the subcomponents of the UDPM attribute field defined in the user-defined pattern rule.	Standardized	None

3.5.7 DSF2® Walk Sequencer transform

Use the DSF2 Walk Sequencer transform to add walk sequencing information to your output data.

DSF2 Walk Sequencer information

Characteristic	Description
	DSF2 Walk Sequencer icon
Use	Use the DSF2 Walk Sequencer transform to enhance your data with information that qualifies your mailings for United States Postal Service (USPS) walk sequence discounts. Potential discounts include: • Carrier Route • Walk Sequence • 90% Residential Saturation • 75% Total Active Saturation DSF2 walk sequencing is 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 in the particular database file only. The software uses DSF2 data to assign sequence numbers for all addresses that are delivery point validation (DPV) confirmed delivery points. The transform outputs addresses that are not DPV confirmed delivery points as blank or with 0000 in the Walk_Sequence_Number output field. **i Note** Before processing your data with the DSF2 Walk Sequencer transform, process it through CASS-certified software such as the USA Regulatory Address Cleanse transform.
Content objects	Use Data Quality blueprints and other content objects, including several blueprints to run DSF2 certifications, to help set up your jobs. These blueprints are located in `<LINK_DIR>\DataQuality\Certifications`.

DSF2 Walk Sequence reference fles [page 429]

The DSF2 reference file is the Delivery Statistics file, which contains the reference information for DSF2 walk sequencing.

DSF2 Walk Sequence Processing options [page 430]
Enter job processing site and certification test mode information.

DSF2 Walk Sequence USPS License Information [page 430]
The USPS requires that you supply your USPS license information, such as license ID and licensee name, for DSF2 walk sequencing.

DSF2 Walk Sequence Data Collection Config [page 431]
To optimize your data flow, control how SAP Data Services forms break keys.

DSF2 Walk Sequence input fields [page 432]
Map fields from your source data to SAP Data Services input fields.

DSF2 Walk Sequence output fields [page 433]
The transform outputs walk sequence number information to the DSF2 Walk Sequencer output fields.

Related Information

Data Quality transform common option [page 383]

3.5.7.1 DSF2 Walk Sequence reference fles

The DSF2 reference file is the Delivery Statistics file, which contains the reference information for DSF2 walk sequencing.

Option	Description
Delstats Directory	Type or browse to the path for the Delivery Statistics file, `dsf.dir`. Best practice is to use a substitution variable for the directory location. Data Services provides the substitution variable `<$$RefFilesAddressCleanse>` for your use. The Delivery Statistics directory file provides counts of business and residential addresses per ZIP Code and per carrier route. Data Services provides the Delivery Statistics file with the U.S. National Directory delivery on a periodic basis.

For the most up to date information about all of the directories, see the directory update letters for SAP Data Services address cleansing and geocoding at https://uacp2.hana.ondemand.com/viewer/p/ADDRESSING_DIRECTORIES.

3.5.7.2 DSF2 Walk Sequence Processing options

Enter job processing site and certification test mode information.

DSF2 Walk Sequencer Processing options

Option	Description
Site Location	The name of your company location at which this job is processed. If your company has multiple locations that provide DSF2 walk sequence processing, ensure that you indicate the location where you process the job that you are currently configuring. Best practice is to use a substitution variable for this value. Data Services provides the substitution variable `<$$DSF2SiteLocation>` for your use.
USPS Certification Testing Mode	The type of certification you are performing for DSF2 walk sequence if applicable. Options include: *None*: Not performing any DSF2 walk sequence certifications. This is the default setting. Choose this option if you are processing a job that will not be submitted to the USPS for DSF2 certification. *Invoice*: Performing certification for DSF2 walk sequence invoice certification. For invoice certification, you certify that the software assigns walk sequence numbers correctly and creates the Delivery Sequence Invoice report. *Sequence*: Performing certification for DSF2 sequence certification. For sequence certification, you certify that the software assigns walk sequence numbers correctly, creates the Delivery Sequence Invoice report, and creates the SEQ log file.

3.5.7.3 DSF2 Walk Sequence USPS License Information

The USPS requires that you supply your USPS license information, such as license ID and licensee name, for DSF2 walk sequencing.

The following table describes the options to complete in the USPS License Information group.

Option	Description
DSF2 License ID	Enter your DSF2 identification number as the USPS assigned it to you. Best practice is to use a substitution variable. Data Services provides the substitution variable `<$$DSF2LicenseeID>` for your use.
Licensee Name	Enter the name of the DSF2 licensed service provider. Best practice is to use a substitution variable. Data Services provides the substitution variable `<$$USPSLicenseeName>` for your use.
List ID	Enter the unique 6-digit identification code that you as the DSF2 licensee assigned to the owner of the mailing list being processed.

3.5.7.4 DSF2 Walk Sequence Data Collection Config

To optimize your data flow, control how SAP Data Services forms break keys.

The settings in the Data Collection Config group control the break key formation. Break keys optimize your data flow by sorting data to form collections of input records. The input records have the same `Postcode1` and `Sortcode_Route` field values.

Option	Description
Replace Null With Space	Specifies whether to convert NULL values to blank spaces in the break key fields. Eliminating NULL values helps to standardize data in the field. With standardized data, the transform can properly form break groups that are consistent in size. *Yes*: Convert NULL to blank spaces. *No*: Do not convert NULL to blank spaces.
Right Pad With Spaces	Specifies whether to right pad the break key field with spaces. Because the break key is used for sorting and aggregating, it is sensitive to the position in which data is placed. When you enable right-padding for the break key fields, the transform can properly form break groups that are consistent in size. *Yes*: Right-pad fields with blank spaces. *No*: Do not right-pad fields with blank spaces. → Tip If the *Replace NULL with space* and the *Right pad with spaces* options are set to *Yes*, then fields with NULL values are replaced with all spaces on the right, to the length of the field.
Presorted Data	Specifies whether the input data has been presorted or not. To make your input data more consistent, it is best to have the software sort data by the break key fields (Postcode1 and Sortcode_Route). *Yes*: The input data has already been presorted by Postcode1 and Sortcode_Route. *No*: The input data has not been sorted yet. → Tip Choosing *No* allows the software to sort your data by Postcode1 and Sortcode_Route. Best practice is to set this option to *No*.

For more information about break keys and how to create them, see the USPS DSF2 section of the *Designer Guide*, or search for "break key" in the *Designer Guide*.

3.5.7.5 DSF2 Walk Sequence input fields

Map fields from your source data to SAP Data Services input fields.

Input field descriptions for DSF2 Walk Sequencer transform

Input field name (DSF2 Walk Sequencer)	Description
Delivery_Point	The two-digit DPBC code.
DPV_Status	The DPV status component that is generated for this record. • *D*: The primary range is a confirmed delivery point, but the secondary range was not available on input. • *L*: The address triggered DPV locking. • *N*: The address is not a valid delivery point. The `<Walk_Sequence_Number>` output field is 0000. • *S*: Dropped secondary address information or dropped trailing alpha from primary range to match primary address to a confirmed delivery point. • *Y*: The address is a confirmed delivery point. The primary range and secondary range (if present) are valid. • Blank: A blank output value indicates that *Enable DPV* is set to *No*, DPV processing is locked, or the transform cannot assign the input address. Also, the `<Walk_Sequence_Number>` output field is blank.
DSF2_Business_Indicator (optional)	Residential or business indicator. Use this information to lower your parcel shipping costs. Some parcel delivery services charge more for delivery to residential addresses. • *Y*: Business address. • *N*: Not a business address. • Blank: Transform did not look up address.
LOT	The Line-of-Travel number.
LOT_Order	The Line-of-Travel sortation: • *A*: Ascending • *D*: Descending LOT codes are required for nonautomated, CART presorting in Standard Mail, Enhanced Carrier Route Subclass.
Postcode1	The five-digit primary ZIP Code. It does not include the four digit ZIP4 Code.
Postcode2	The four-digit ZIP4 code. On a mail piece, this code follows the primary postal code with a hyphen placed between, for example, 54601-1234.
Sortcode_Route	The four-digit carrier route number.

3.5.7.6 DSF2 Walk Sequence output fields

The transform outputs walk sequence number information to the DSF2 Walk Sequencer output fields.

The following table contains descriptions for the recognized DSF2 Walk Sequencer output fields.

Output field name (DSF2 Walk Sequencer)	Description
Active_Del_Discount	Indicates whether the Postcode1—Sortcode route combination qualifies for the 75% total active delivery discount. *Y:* The Postcode1—Sortcode route combination qualifies for the 75% total active delivery discount. *N:* The Postcode1—Sortcode route combination does not qualify for the 75% total active delivery discount. `<blank>`: The record was not sequenced. → Tip Active deliveries include residential, business, and PO Box addresses.
Residential_Sat_Discount	Indicates whether the Postcode1—Sortcode route combination qualifies for the 90% residential saturation discount. *Y:* The Postcode1—Sortcode route combination qualifies for the 90% residential saturation discount. *N:* The Postcode1—Sortcode route combination does not qualify for the 90% residential saturation discount. `<blank>`: The record was not sequenced.
Sortcode_Route_Discount	Indicates whether the Postcode1—Sortcode route combination qualifies for the Sortcode (Carrier Route) discount. *Y:* The Postcode1—Sortcode route combination qualifies for the Sortcode discount. *N:* The Postcode1—Sortcode route combination does not qualify for the Sortcode discount. `<blank>`: The record was not sequenced. → Tip Mailers must have 10 or more deliveries to the same Postcode1—Sortcode combination to qualify for the discount.

Output field name (DSF2 Walk Sequencer)	Description
Walk_Sequence_Discount	Indicates whether the Postcode1—Sortcode route combination qualifies for the walk sequence discount. *Y:* The Postcode1—Sortcode route combination qualifies for the walk sequence discount *N:* The Postcode1—Sortcode route combination does not qualify for the walk sequence discount. `<blank>`: The record was not sequenced. **→ Tip** Mailers must have 125 or more sequenced delivery points for each Postcode1—Sortcode route combination to qualify for the discount.
Walk_Sequence_Number	Indicates the sequence number from 0000 to 9999. **i Note** If the `<Postcode2>` field is blank, this field is blank. **i Note** If the `<Postcode2>` field is not blank and *DPV_Status* is *N*, then the field contains 0000.

3.5.8 Geocoder transform

Use the Geocoder transform to append latitude, longitude, and US census data to your data.

Geocoder information

Characteristic	Description
	Geocoder icon

Characteristic	Description
Use	The Geocoder transform uses geographic coordinates expressed as latitude and longitude, addresses, and point-of-interest (POI) data. Append addresses, latitude and longitude, US census data, and other information to your data. **i Note** Data Services uses the World Geodetic System (WGS84) for latitude and longitude results. WGS84 is defined and maintained by the US National Geospacial-Intelligence Agency (NGA). The transform has options to enable generation of data quality statistics tables in the repository for address geocoding mode. **i Note** Not applicable for reverse geocoding and point of interest (POI) textual search. These statistics provide insight into how the software cleansed and assigned your data by generating aggregated summary statistics and specific record-level statistics. 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 transform returns 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 transform outputs the results in the *Result_List XML* output field. Best practice is to use the Geocoder transform with the Global Address Cleanse or USA Regulatory Address Cleanse transform.

Characteristic	Description
Content objects	Download blueprints and other content objects for Geocoder from the SAP Community WIKI at https://wiki.scn.sap.com/wiki/display/EIM/SAP+Data+Services+Blueprints.

Geocoder Report and Analysis [page 437]
: Generate report data for the Geocoder transform by setting the option in the Report and Analysis group.

Geocoder transform options [page 437]
: To specify the assignment level options for the Geocoder transform, set options in the Geocoder options group.

Geocoder Reference files [page 445]
: To set the location for the Geocoder reference files, set the *Directory Path* option.

Geocoder directories [page 446]
: The Geocoder transform provides two directory formats based on the type of geocoding processing you perform.

Geocoder input fields [page 447]
: Map fields from your input data to Geocoder input fields so the transform knows the type of data to expect on input.

Geocoder output fields [page 451]
: Map Geocoder output fields to output standardized and corrected data, generated data, and additional data.

Geocoder transform: Result_List XML output fields [page 462]
: To include Geocoder result information in your generated output data, map output fields to the `Result_List` XML output field.

Geocoder information codes [page 466]
: The transform assigns information codes to a record when you include the *INFO_CODE* output field in your field mapping.

Geocoder status codes [page 468]
: The transform assigns status codes to a record when you include the *STATUS_CODE* output field in your field mapping.

Related Information

Data Quality transform common option [page 383]

Data Quality Statistics for Data Cleanse, Global Address Cleanse, and Geocoder transforms [page 383]

3.5.8.1 Geocoder Report and Analysis

Generate report data for the Geocoder transform by setting the option in the Report and Analysis group.

Report and analysis option description

Option	Description
Generate Report Data	Specifies whether to generate report data for this transform. • *YES*: Generates report data for this transform. • *NO*: Turns off report data generation To improve performance, set this option to *NO* when you do not need to generate statistics. For example, during testing. **i Note** This option is available only for the address geocoding mode. If you set this value to *YES* for the reverse geocoding and POI textual search modes, the software generates a warning and does not generate a report.

Parent topic: Geocoder transform [page 434]

Related Information

Geocoder transform options [page 437]
Geocoder Reference files [page 445]
Geocoder directories [page 446]
Geocoder input fields [page 447]
Geocoder output fields [page 451]
Geocoder transform: Result_List XML output fields [page 462]
Geocoder information codes [page 466]
Geocoder status codes [page 468]

3.5.8.2 Geocoder transform options

To specify the assignment level options for the Geocoder transform, set options in the Geocoder options group.

The following option group is required. The options support all countries.

Each option supports a directory format based on the mode. The mapped input fields determine the mode. The following table contains the directory format and related mode and returns.

Directory formats, modes, and returns

Format	Mode	Returns
Basic	**Mode:** Address geocoding mode.	**Returns:** • Centroid level latitude and longitude information • Address level latitude and longitude information
Advanced	**Mode:** • Address geocoding mode • Reverse geocoding mode • POI textual search mode	**Returns:** • Range based latitude and longitude • Centroid level latitude and longitude • Point of interest

The following table contains descriptions for the Geocoder transform options, and it includes the supported directory format.

Option	Description	Directory Formats
Best Assignment Level	Specifies the depth of assignment for the latitude and longitude output fields. • *PREFERRED*: Assigns to the finest depth, which is to the primary number level by default. • *PRIMARY_NUMBER*: Assigns to the primary number level. • *POSTCODE*: Assigns to the post-code level. • *LOCALITY*: Assigns to the locality, city, or suburb level. • *SMALLEST_AREA*: First attempts to assign to the primary number. If no primary number, assigns based on postcode or locality information, depending on the level that is the smaller area. **Example** For example, the French post-code 75014 is a smaller area than the locality of Paris. **Note** If you set the *Locality Assignment Threshold* or *Postcode Assignment Threshold* options to *NONE* for the *Preferred*, *Primary Number*, and *Smallest Area* assignment level values, the software uses locality or postcode centroid assignment.	Basic and Advanced

Option	Description	Directory Formats
Distance Unit	Specifies the unit of distance used for the radius. • *KILOMETERS* • *MILES* The value of the *Option_Distance_Unit* input field takes precedence over the value of the *Distance Unit* option. The transform uses the value of the *Distance Unit* option only under the following circumstances: • You do not map the *Option_Distance_Unit* input field • The *Option_Distance_Unit* input field is invalid • The *Option_Distance_Unit* input field is blank • The *Option_Distance_Unit* input field is NULL	Advanced

Option	Description	Directory Formats
Locality Assignment Threshold	Applicable for address geocoding mode and reverse geocoding with address mode. Limits the level of locality centroid assignment. Specifies the locality level to use for the assignment threshold. • *NONE*: Skips the specific assignment level. Set to *NONE* when you do not want to return an assignment threshold on locality. • *PREFERRED*: Assigns to the finest depth at the locality level. **Example** The address in the following table does not have a Locality4. If you choose *PREFERRED*, the transform returns Locality3, which is the finest depth when there is no Locality4. **Address** / **Locality Level** Church Cottage / Locality3 Pemborough / Locality2 Bristol / Locality1 (most general) Setting to *Locality1* excludes Locality2–4 during assignment, even though the software may return values at those levels. However, if you set to *Locality4* and there is no *Locality4* data, the transform returns the finest available data even if that data is at Locality2 level.	Basic and Advanced

Option	Description	Directory Formats
Max Records	Specifies the maximum number of records that can be returned. Enter a number up to 100. The value of the *Option_Max_Records* input field takes precedence over the value of the *Max Records* option. The software uses the value of the *Max Records* option only when you do not map the *Option_Max_Records* input field or it is blank.	Advanced
Offset Coordinates	Specifies whether the software returns the offset values of latitude and longitude when the side of the street is known. Use this option for address geocoding mode, and also for reverse geocoding with address mode. • *YES*: Returns the offset values. • *NO*: Returns the center value regardless of whether the side of the street is known.	Advanced

Option	Description	Directory Formats
Postcode Assignment Threshold	Limits the level of postcode centroid assignment. • *POSTCODE_FULL*: Assigns to the entire extended postcode. **Example** In the USA, *POSTCODE_FULL* assigns to the 5-digit postcode and all four digits of the ZIP+4. • *POSTCODE1*: Assigns to the city or postcode area. **Example** In the USA, *POSTCODE1* assigns to the 5-digit ZIP Code. • *POSTCODE2_PARTIAL*: Assigns to the first few characters of the extended postcode. **Example** In the USA, *POSTCODE2_PARTIAL* assigns the 5-digit postcode and the first two digits of the ZIP+4. • *PREFERRED*: Assigns to the finest depth at the postcode level. • *NONE*: Skips the specific assignment level. Use this setting when you do not want to return an assignment threshold on postcode.	Basic and Advanced

Option	Description	Directory Formats
Radius	The distance from a specified reference point. The reference point identifies an area in which matching records are located. The default value is 1 Kilometer. In reverse geocoding mode, the maximum radius distance is 111 kilometers or 68.97 miles. The value of the *Option_Radius* input field takes precedence over the value of the *Radius* option. The software uses the value of the *Radius* option only when the *Option_Radius* input field is not mapped or is blank.	Advanced

Parent topic: Geocoder transform [page 434]

Related Information

Geocoder Report and Analysis [page 437]
Geocoder Reference files [page 445]
Geocoder directories [page 446]
Geocoder input fields [page 447]
Geocoder output fields [page 451]
Geocoder transform: Result_List XML output fields [page 462]
Geocoder information codes [page 466]
Geocoder status codes [page 468]

3.5.8.3 Geocoder Reference files

To set the location for the Geocoder reference files, set the *Directory Path* option.

Geocoder reference file option

Option	Description
Directory Path	Specifies the path to the location of your Geocoder directory files. For easier set up, use the substitution variable $$RefFiles-Geocoder as a value for this option. The $$RefFilesGeocoder substitution variable is set for the default reference file location, which is the path relative to where Data Services is installed. **Example** `<INSTALL_DIR>\Data Services \DataQuality\reference_data` If you installed Data Services to a different location, change the substitution variable while designing the work flow. For more information about variables, see the Variables and Parameters section in the *Data Services Designer Guide.* To set more than one location, duplicate the Directory Path option by right-clicking and selecting *Duplicate option*. Keep in mind that you cannot have the same directory file located in both locations. 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 letters at https://uacp2.hana.ondemand.com/viewer/p/ADDRESSING_DIRECTORIES.

Parent topic: Geocoder transform [page 434]

Related Information

Geocoder Report and Analysis [page 437]
Geocoder transform options [page 437]
Geocoder directories [page 446]
Geocoder input fields [page 447]
Geocoder output fields [page 451]
Geocoder transform: Result_List XML output fields [page 462]
Geocoder information codes [page 466]
Geocoder status codes [page 468]

3.5.8.4 Geocoder directories

The Geocoder transform provides two directory formats based on the type of geocoding processing you perform.

The Geocoder directories are designed specifically to use with the Geocoder transform. 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 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, supported countries, directory format, geocoding mode, centroid level, and census information. It also lists the available POI types supported by the vendor.

The Geocoder transform is flexible enough to accept new country directory data immediately after the directory data is released. There is no need to wait for the next SAP Data Services release to begin using new country directory data.

For information about downloading directories, see the latest Directories Update letters at https://uacp2.hana.ondemand.com/viewer/p/ADDRESSING_DIRECTORIES.

Parent topic: Geocoder transform [page 434]

Related Information

Geocoder Report and Analysis [page 437]
Geocoder transform options [page 437]
Geocoder Reference files [page 445]
Geocoder input fields [page 447]
Geocoder output fields [page 451]
Geocoder transform: Result_List XML output fields [page 462]
Geocoder information codes [page 466]
Geocoder status codes [page 468]

3.5.8.5 Geocoder input fields

Map fields from your input data to Geocoder input fields so the transform knows the type of data to expect on input.

> **i Note**
>
> Reference to "country" or "countries" in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

The following table contains an alphabetical list for all of the recognized input fields for the Geocoder transform. It includes the applicable category for each field.

The table also shows the input field availability based on currently supported directory formats. Supported directory formats are:

- *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.

The input fields support all countries.

Name	Category	Description	Directory format
COUNTRY	Address	Contains the two-character ISO country code.	Basic, Advanced
DATA_SOURCE_ID	N/A	Specifies the name of the data source that the record comes from. **→ Tip** If you choose to generate data quality statistics tables, use this field as part of your mapping strategy, along with primary key fields and the output field *ROW_ID*. This mapping identifies the source of individual records in any of the non-summary data quality statistics tables, thus enabling you to trace problem data to the data source. This input field is applicable for the following transforms: • Data Cleanse • Global Address Cleanse • Geocoder	N/A
LATITUDE	Latitude/ Longitude	Contains the relative distance north or south of the equator, measured in 0-90 degrees.	Advanced

Name	Category	Description	Directory format
LOCALITY1–4	Address	Contains the city, town, or suburb and any additional related information. **i Note** For China geocoding, locality description is required. Use LOCALITY1_FULL, LOCALITY2_FULL from the GAC output fields for mapping since they contain a locality description.	Basic, Advanced
LONGITUDE	Latitude/ Longitude	Contains the relative distance east or west of the Greenwich meridian, measured in 0-180 degrees.	Advanced
OPTION_DISTANCE_UNIT	Search Filter	Contains the distance unit of measure used for the radius. • *KILOMETERS* • *MILES* The value of the *OPTION_DISTANCE_UNIT* input field takes precedence over the value of the *Distance Unit* Geocoder option. The transform uses the value of the *Distance Unit* option only when the *OPTION_DISTANCE_UNIT* input field isn't mapped, is invalid, blank, or NULL. The *OPTION_DISTANCE_UNIT* field is a dynamic input field. If you change the setting, you don't have to terminate and reinitialize the transform for the transform to recognize the new configuration.	Advanced
OPTION_MAX_RECORDS	Max Records	Contains the maximum number of records that the transform returns. Enter a number from 0 through 100. A value greater than 0 outputs multiple results as XML to the *RESULT_LIST* output field rather than to individual output fields. The value of the *OPTION_MAX_RECORDS* input field takes precedence over the value of the *Max_Records* Geocoder option. The transform uses the value of the *Max_Records* option only when the *OPTION_MAX_RECORDS* input field isn't mapped, is blank, or NULL. The *OPTION_MAX_RECORDS* field is a dynamic input field. If you change this setting, you don't have to terminate and reinitialize the transform for the transform to recognize the new configuration.	Advanced

Name	Category	Description	Directory format
OPTION_RADIUS	Search Filter	Contains the distance from a specified reference point to identify an area in which matching records are located. In reverse geocoding mode, the maximum radius distance is 111 kilometers or 68.97 miles. The value of the *OPTION_RADIUS* input field takes precedence over the value of the *Radius* Geocoder option. The transform uses the value of the *Radius* option only when the *OPTION_RADIUS* input field isn't mapped, is blank, or NULL. This field is a dynamic input field. If you change this setting, you don't have to terminate and reinitialize the transform for the transform to recognize the new configuration.	Advanced
POI_NAME	Address POI	Contains the name of a point of interest, such as the Washington Monument.	Advanced
POI_TYPE	Address POI	Contains the point-of-interest type expressed as a number. For example, for one vendor 5999 is a historical monument. To return multiple point-of-interest types in POI textual search mode, concatenate POI type codes using a colon as a delimiter. **Example** To return all schools (type 8211) and libraries (type 8231) within a defined area, you enter: `8211:8231` The POI types and their corresponding codes differ depending on the data vendor that you use. For a detailed list of available POI types, see the vendor-specific directory update.	Advanced
POSTCODE1–2	Address	Contains the postal code and a secondary postal code, when available.	Basic, Advanced
PRIMARY_NAME1–4	Address	Contains the street name.	Basic, Advanced
PRIMARY_NUMBER	Address	Contains the premise number.	Basic, Advanced
PRIMARY_POSTFIX1	Address	Contains the abbreviated directional, such as N, S, NW, SE, that follows a street name.	Basic, Advanced
PRIMARY_PREFIX1	Address	Contains the abbreviated directional, such as N, S, NW, SE, that precedes a street name.	Basic, Advanced
PRIMARY_TYPE1–4	Address	Contains the abbreviated type of primary name, such as St., Ave., or Pl.	Basic, Advanced

Name	Category	Description	Directory format
REGION1–2	Address	Contains the region symbol of the state, province, or territory.	Basic, Advanced
SEARCH_FILTER_NAME	Search Filter	Contains the search criteria for a point-of-interest name.	Advanced
SEARCH_FILTER_TYPE	Search Filter	Contains the search criteria for a point-of-interest type, expressed as a four-digit number. For example, for one vendor, 5999 is a historical monument. To return only an address, enter **`ADDR`**. To return multiple point-of-interest types, concatenate POI codes using a colon as a delimiter. **Example** To return all schools (type 8211) and libraries (type 8231) within a defined area, you enter: **`8211:8231`** To return a point-of-interest type and its address, enter: **`5999:ADDR`** The POI types and their corresponding codes differ depending on the data vendor that you use. For a detailed list of available POI types, see the vendor-specific directory update.	Advanced

Parent topic: Geocoder transform [page 434]

Related Information

Geocoder Report and Analysis [page 437]
Geocoder transform options [page 437]
Geocoder Reference files [page 445]
Geocoder directories [page 446]
Geocoder output fields [page 451]
Geocoder transform: Result_List XML output fields [page 462]
Geocoder information codes [page 466]
Geocoder status codes [page 468]
Dynamic transform settings [page 244]

3.5.8.6 Geocoder output fields

Map Geocoder output fields to output standardized and corrected data, generated data, and additional data.

The following table contains an alphabetical list of recognized output fields to use in the output mapping for the Geocoder transform. The table also shows the output field availability based on supported countries and directory formats.

Supported countries include the following:

- **All**
- **USA**

Supported directory formats include the following:

- **Basic:** Supports address geocoding mode. Returns centroid-level and address-level latitude and longitude information.
- **Advanced:** Supports address geocoding, reverse geocoding, and POI textual search modes. Returns range-based and centroid-level latitude and longitude information and point-of-interest information.

Name	Category	Description	Supported countries	Directory Format
ADDRESS_LINE	Address	Contains the primary address. The primary address can contain components such as the primary range, primary name, post-directional, pre-directional, and suffix.	All	Basic, Advanced

Name	Category	Description	Supported countries	Directory Format
ASSIGNMENT_LEVEL	Assignment Level	Contains the level to which the transform matches the address to the data in the reference files (directories). • *PRE*: Primary Range Exact. Assigns to the exact location of the address, for example, 123 Main St. *PRE* is the most precise level of assignment. To obtain the PRE, map either the *POI_TYPE* input field or the *PRIMARY_NAME* and *PRIMARY_NUMBER* input fields. • *PRI*: Primary Range Interpolated. Assigns to the level of the address range, for example, 100–500 Main St. • *L1–4*: Assigns to the level of city, town, or suburb. • *P1*: Postcode1. Assigns to the level of Postcode1. • *P2P*: Postcode2 Partial. Assigns the full Postcode1 and the first few characters of Postcode2. • *PF*: Postcode Full. Assigns to the level of Postcode1	All	Basic, Advanced

Name	Category	Description	Supported countries	Directory Format
		and Postcode2, when available.		
ASSIGNMENT_LEVEL_LOCALITY	Assignment Level	Contains the level to which the transform assigns the locality. *L1–4*: Returns up to four locality levels. L1 is the most general and L4 is the most specific.	All	Basic, Advanced
ASSIGNMENT_LEVEL_POSTCODE	Assignment Level	Contains the level to which the transform assigns the postcode. • *P1*: Postcode1. Assigns to the level of Postcode1. • *P2P*: Postcode2 Partial. Assigns the full Postcode1 and the first few characters of Postcode2. • *PF*: Postcode Full. Assigns to the level of Postcode1 and Postcode2, when available.	All	Basic, Advanced
CENSUS_TRACT_BLOCK	Census	Contains 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	Contains the census tract code in the previous version of census data.	N/A	N/A

Name	Category	Description	Supported countries	Directory Format
CENSUS_TRACT_BLOCK_GROUP	Census	Contains the census tract and block group code as defined by the government for reporting census information. The government uses census tract and block group codes for matching with demographic-coding databases. In the USA, the code represents the following elements: • The first six digits contain the tract number, for example, 002689. • The first of the last four digits contain the BG number within the tract. The BG is a cluster of census blocks having 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	Contains the census tract block group code in the previous version of census data.	N/A	N/A
COUNTRY_CODE	Address	Contains the two-character ISO country code.	All	Basic, Advanced

Name	Category	Description	Supported countries	Directory Format
DISTANCE	Distance	Contains 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	Contains a unique county code as defined by the government for reporting census information. For example, in the USA, the code is a Federal Information Processing Standard (FIPS) three-digit county code.	USA	Basic, Advanced
GOV_LOCALITY1_CODE	Census	Contains 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	Contains a unique region code as defined by the government for reporting census information. For example, in the USA, the code is a Federal Information Processing Standard (FIPS) two-digit state code.	USA	Basic, Advanced

Name	Category	Description	Supported countries	Directory Format
INFO_CODE	Info Code	Contains a three-character code that provides information about the geocoding results: • Third character indicates the status for address and point-of-interest geocoding assignment. • Second and third characters indicate the status for reverse geocoding assignment. If assigned to the best level, the *INFO_CODE* field is blank. The first character is reserved for future use.	All	Basic, Advanced
LATITUDE	Latitude/ Longitude	Contains the latitude at the best assigned level, which is 0–90 degrees north or south of the equator. The transform standardizes the latitude to six decimals in the format 45.801357.	All	Basic, Advanced
LATITUDE_LOCALITY	Latitude/ Longitude	Contains the latitude at the locality level centroid of the city, town, locality, or suburb. The transform standardizes the latitude to six decimals in the format 45.801357.	All	Basic, Advanced
LATITUDE_POSTCODE	Latitude/ Longitude	Contains the latitude at the postcode level centroid of the postcode. The transform standardizes the latitude to six decimals in the format 45.801357.	All	Basic, Advanced

Name	Category	Description	Supported countries	Directory Format
LATITUDE_PRIMARY_NUMBER	Latitude/ Longitude	Contains the latitude at the primary number level. The transform standardizes the latitude to six decimals in the format 45.801357.	All	Basic, Advanced
LOCALITY1–4	Address	Contains the city, town, or suburb and any additional related information.	All	Basic, Advanced
LONGITUDE	Latitude/ Longitude	Contains the longitude at the best assigned level, which is 0–180 degrees east or west of Greenwich meridian. The transform standardizes the longitude to six decimals in the format 123.458331.	All	Basic, Advanced
LONGITUDE_LOCALITY	Latitude/ Longitude	Contains the longitude at the locality level centroid of the city, town, locality, or suburb. The transform standardizes the longitude to six decimals in the format 123.458331.	All	Basic, Advanced
LONGITUDE_POSTCODE	Latitude/ Longitude	Contains the longitude at the postcode level centroid of the postcode. The transform standardizes the longitude to six decimals in the format 123.45833.	All	Basic, Advanced
LONGITUDE_PRIMARY_NUMBER	Latitude/ Longitude	Contains the longitude at the primary number level. The transform standardizes the longitude to six decimals in the format 123.458331.	All	Basic, Advanced

Name	Category	Description	Supported countries	Directory Format
METRO_STAT_AREA_CODE	Census	Contains the metropolitan statistical area. A metropolitan statistical area has a large population with a high degree of social and economic integration with the core of the area. The government defines the area for reporting census information. In the USA, the 0000 code indicates that the address does not lie in a metropolitan statistical area; usually a rural area.	USA	Basic
METRO_STAT_AREA_CODE_PREV	Census	Contains the metropolitan statistical area in the previous version of census data.	N/A	N/A
MINOR_DIV_CODE	Census	Contains 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 is defined in a state or province that does not have a well-defined minor civil division. The government defines the area for reporting census information.	USA	Basic, Advanced

Name	Category	Description	Supported countries	Directory Format
MINOR_DIV_CODE_PREV	Census	Contains the minor civil division or census county division code in the previous version of census data.	N/A	N/A
POI_NAME	Address POI	Contains the point of interest name such as the Washington Monument.	All	Advanced
POI_TYPE	Address POI	Contains the point of interest type expressed as a four-digit number, such as 5999, which is a historical monument.	All	Advanced
POPULATION_CLASS_LOCALITY1	Population	Contains a code that indicates the population falls within a certain size. • *0*: Undefined. The population may be too large or small to provide accurate data. • *1*: Over 1 million. • *2*: 500,000–999,9999. • *3*: 100,000–499,999. • *4*: 50,000–99,999. • *5*: 10,000–49,999. • *6*: Less than 10,000.	All	Advanced
POSTCODE	Address	Contains the post code.	All	Basic, Advanced
POSTCODE1–2	Address	Contains the post code and a secondary post code, when available.	All	Basic, Advanced
PRIMARY_NAME1–4	Address	Contains the street name.	All	Basic, Advanced
PRIMARY_NUMBER	Address	Contains the premise number.	All	Basic, Advanced

Name	Category	Description	Supported countries	Directory Format
PRIMARY_POSTFIX1	Address	Contains the abbreviated directional, such as N, S, NW, SE, that follows a street name.	All	Basic, Advanced
PRIMARY_PREFIX1	Address	Contains the abbreviated directional, such as N, S, NW, SE, that precedes a street name.	All	Basic, Advanced
PRIMARY_RANGE_LOW	Address	Contains the high value of a primary number range.	All	Basic, Advanced
PRIMARY_RANGE_HIGH	Address	Contains the low value of a primary number range.	All	Basic, Advanced
PRIMARY_TYPE1–4	Address	Contains the abbreviated type of primary name, such as St., Ave., or Pl.	All	Basic, Advanced
REGION1–2	Address	Contains the region symbol of the state, province, or territory.	All	Basic, Advanced
RESULT_LIST	Results	Contains the XML output when multiple records are returned for a search.	All	Advanced
RESULT_LIST_COUNT	Results	Contains the number of results in the *RESULT_LIST* output field.	All	Advanced

Name	Category	Description	Supported countries	Directory Format
ROW_ID	Statistics	Contains an internal ID generated by the transform that uniquely identifies a row processed by that transform. Use this field to join the input and output to the applicable non-summary data quality statistics table. If you select to generate any data quality statistics tables in the transform, and you don't include the *ROW_ID* output field, the software issues a warning. Applicable transforms are Data Cleanse, Global Address Cleanse, and Geocoder.	N/A	N/A
SIDE_OF_PRIMARY_ADDRESS	Side of Street	Indicates that the primary address 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	Contains 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 government defines the area for reporting census information.	USA	Basic
STAT_AREA_CODE_PREV	Census	Contains the statistical area code in the previous version of census data.	N/A	N/A

Name	Category	Description	Supported countries	Directory Format
STATUS_CODE	Status Code	Contains a four-character status code that always starts with an S. This code represents the fuzzy matches made to the address/point of interest (POI) component during address geocoding processing.	All	Advanced, Basic

Parent topic: Geocoder transform [page 434]

Related Information

Geocoder Report and Analysis [page 437]
Geocoder transform options [page 437]
Geocoder Reference files [page 445]
Geocoder directories [page 446]
Geocoder input fields [page 447]
Geocoder transform: Result_List XML output fields [page 462]
Geocoder information codes [page 466]
Geocoder status codes [page 468]

3.5.8.7 Geocoder transform: Result_List XML output fields

To include Geocoder result information in your generated output data, map output fields to the `Result_List` XML output field.

The following table contains output fields that you can include in the `Result_List` XML output field. The table lists the fields 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.

Output field name	Category	Description
Assignment_Level	Assignment Level	The level to which this transform matched the address to the data in the reference fields (directories). *PRE:* Primary Range Exact assigns to the exact location of the address; for example, 123 Main St. Primary Range Exact is the most precise level of assignment. To obtain the PRE, map either the `<POI_Type>` input field or the `<Primary_Name>` and `<Primary_Number>` input fields. *PRI:* Primary Range Interpolated assigns to the level of the address range; for example, 100-500 Main St. *L1–4:* Locality1-4 assigns to the level of city, town, or suburb. *P1:* Postcode1 assigns to the level of Postcode1. *P2P:* Postcode2 Partial assigns the full Postcode1 and the first few characters of Postcode2. *PF:* Postcode Full assigns to the level of Postcode1 and Postcode2, when available.
Country_Code	Address	The two-character ISO country code.
Distance	Distance	The distance from the input address, geographical coordinates, or point of interest to the closest address or point of interest.
Group_Number	Group	An identifier for multiple records that represent the same address. For example, one record might have an alias primary name of another. This output field has a value only when the same address is returned in multiple entries. When entries have the same value in Group_Number, it means that the entries have the same address.

Output field name	Category	Description
Group_Rank	Group	Indicates whether a record in a group is a master or a subordinate. For each group of entries that have the same Group_Number value, one of them has Group_Rank of "M" (master) and all other entries in the group have "S" (subordinate). You may obtain only one of each address returned by filtering where the Group_Rank is either "M" or a blank string. *M:* Master *S:* Subordinate
Latitude	Latitude/Longitude	The latitude at the best assigned level (0–90 degrees north or south of the equator) in the format 45.32861.
Locality1–4	Address	The city, town, or suburb and any additional related information.
Longitude	Latitude/Longitude	The longitude at the best assigned level (0–180 degrees east or west of Greenwich meridian) in the format 123.45833.
POI_Name	Address POI	The point of interest name, such as the Washington Monument.
POI_Type	Address POI	The point of interest type expressed as a four-digit number; for example, 5999 (historical monument).
Postcode	Address	The postal code.
Postcode1–2	Address	The postal code and a secondary postal code, if available.
Primary_Name1–4	Address	The street name.
Primary_Number	Address	The premise number.
Primary_Postfix1	Address	Abbreviated directional (N, S, NW, SE) that follows a street name.
Primary_Prefix1	Address	Abbreviated directional (N, S, NW, SE) that precedes a street name.
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.).

Output field name	Category	Description
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.

To learn how to incorporate the `Result_List` output field in your output data, read about XML extraction in the *Designer Guide*. Search for "Result_List".

Example

The following is the Result_List XML output for 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>
```

Parent topic: Geocoder transform [page 434]

Related Information

Geocoder Report and Analysis [page 437]
Geocoder transform options [page 437]
Geocoder Reference files [page 445]
Geocoder directories [page 446]
Geocoder input fields [page 447]
Geocoder output fields [page 451]
Geocoder information codes [page 466]
Geocoder status codes [page 468]

3.5.8.8 Geocoder information codes

The transform assigns information codes to a record when you include the *INFO_CODE* output field in your field mapping.

> **i Note**
>
> Reference to "country" or "countries" in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

The *INFO_CODE* output field is a three-character code that provides information about geocoding results. Each character of the code holds specific types of information:

- First character: Not currently used.
- Second and third character: Status for the reverse geocoding assignment.
- Third character: Status for the address geocoding and POI textual search modes.

If you set the assignment to the best level, the *INFO_CODE* field is blank.

Use the following table to understand the code assigned to the *INFO_CODE* output field.

Information code	Description
001	Reference data isn't available for the input country. Verify that the directory is installed and the reference path to the directory is valid.
002	Address-level reference data isn't 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.
003	Centroid-level reference data isn't 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.
004	Assignment is limited. The input data is insufficient or incorrect to match the reference data. When the Best Assignment Level fails, this code is output. The assignment may be made to a lower assignment level than the one specified. For example, if you set Best Assignment Level to Primary Number and the Primary Number field is blank, the assignment may be at the postcode or locality level, if the data is available. Verify your input data and input field mapping and make sure that the fields required for best assignment exist and are correctly mapped.
005	The input data doesn't match anything in the reference data. When the input record doesn't match the directory data for the Best Assignment Level or a lower assignment level, this code is output. Verify your input data and input field mapping and make sure that the fields required for best assignment exist and are correctly mapped.

Information code	Description
006	The input data assigns ambiguously in the reference data. There's a tie for the Best Assignment Level. The input record matches several records in the directory data and the software can't decide which one is the best. **Example** Your reference data has two records: Record 1: 100 Main St La Crosse WI 54650 Record 2: 100 Main St Bt Micts WI 54650 Record 2 doesn't have a valid locality name. Therefore, the software outputs 006 for ambiguous assignment. Verify your input data and input field mapping and make sure that the fields required for best assignment exist and are correctly mapped.
007	The input data is blank or invalid. For example, if the US Postcode1 is a five-digit string and your input data is a six-digit string, the 007 information code is output. Verify your input data and input field mapping and make sure that the fields required for best assignment exist and are correctly mapped.
008	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.
009	The POI type provided on input is invalid. The point-of-interest type isn't correct. Verify your input POI type with the POI types described in the appropriate directory update letter.
050	Reverse geocoding. The input data doesn't match anything in the reference data.
060	Reverse geocoding. The input data assigns ambiguously in the reference data.
070	Reverse geocoding. The input data is blank or invalid.
090	Reverse geocoding. The POI type provided on input is invalid.
00A	The POI input data wasn't used. The POI name or type doesn't match the directory data. A PRI or PRE level assignment was made; however, the input POI name and POI type weren't used for the assignment.
0B0	The input data wasn't 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 can't be performed based on the address.
0C0	Not all results returned for the input data, because the number of results exceeds the specified Max_Record. This code only occurs during reverse geocoding assignment. Increase the Max_Record value.
0D0	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.

Information code	Description
00E	The closest latitude and longitude to the input address is returned because the input house number doesn't exist in the geocoder directories. For example: Input address: 100 Main St La Crosse WI 54650 Directory data: 1–88 Main St La Crosse WI 54650 The software returns the latitude and longitude values for 88 Main St with the 00E information code to indicate that the house number doesn't match the directory data.
0F0	The feature enabled by the current configuration isn't supported by the installed basic directory. For example, a configuration requests reverse geocoding but it finds only the basic directory installed. Verify that the advanced directory format is installed and the reference path to the directory is valid. For more information about directories, see "Directory data" in the *SAP Data Services Installation Guide*.

Parent topic: Geocoder transform [page 434]

Related Information

Geocoder Report and Analysis [page 437]
Geocoder transform options [page 437]
Geocoder Reference files [page 445]
Geocoder directories [page 446]
Geocoder input fields [page 447]
Geocoder output fields [page 451]
Geocoder transform: Result_List XML output fields [page 462]
Geocoder status codes [page 468]

3.5.8.9 Geocoder status codes

The transform assigns status codes to a record when you include the *STATUS_CODE* output field in your field mapping.

Status codes are four characters that represent the fuzzy matches made to the address—point of interest (POI) component during address geocoding processing. Each character represents an aspect of the status:

- The first character is always S (for Status).
- The second character is associated with any last-line fuzzy matches to the postal code, region, or locality.
- The third character is associated with any address-line fuzzy matches to the primary name prefix, primary name postfix, primary type, primary name, and primary range.

- The fourth character is associated with any POI data fuzzy matches.

> **i Note**
>
> The transform uses the *STATUS_CODE* output field only for the address geocoding process.

First character

The value of the first character is always S for Status.

Second character

The value of the second character depends on fuzzy matches to the postal code, region, or locality.

Second character status code descriptions

Value	Description
0	No significant difference between the input data and the reference data.
1	Fuzzy match on postal code.
2	Fuzzy match on region.
3	Fuzzy match on postal code and region.
4	Fuzzy match on locality.
5	Fuzzy match on postal code and locality.
6	Fuzzy match on region and locality.
7	Fuzzy match on postal code, region, and locality.

Third character

Third character status code descriptions

Value	Description
0	No significant difference between the input data and the reference data.
1	Fuzzy match on primary name prefix/primary name postfix.
2	Fuzzy match on primary type.
3	Fuzzy match on primary name prefix/primary name postfix and primary type.
4	Fuzzy match on primary name.
5	Fuzzy match on primary name prefix/primary name postfix and primary name.
6	Fuzzy match on primary type and primary name.

Value	Description
7	Fuzzy match on primary name prefix/primary name postfix, primary type, and primary name.
8	Fuzzy match on primary range.
9	Fuzzy match on primary name prefix/primary name postfix and primary range.
A	Fuzzy match on primary type and primary range.
B	Fuzzy match on primary name prefix/primary name postfix, primary type, and primary range.
C	Fuzzy match on primary name and primary range.
D	Fuzzy match on primary name prefix/primary name postfix, primary name, and primary range.
E	Fuzzy match on primary type, primary name, and primary range.
F	Fuzzy match on primary name prefix/primary name postfix, primary type, primary name, and primary range.

Fourth character

Fourth character status code descriptions

Value	Description
0	No significant difference between the input data and the reference data.
1	Fuzzy match on POI name.

Parent topic: Geocoder transform [page 434]

Related Information

Geocoder Report and Analysis [page 437]
Geocoder transform options [page 437]
Geocoder Reference files [page 445]
Geocoder directories [page 446]
Geocoder input fields [page 447]
Geocoder output fields [page 451]
Geocoder transform: Result_List XML output fields [page 462]
Geocoder information codes [page 466]

3.5.9 Global Address Cleanse transform

Use the Global Address Cleanse transform to identify, parse, validate, and correct your global address data.

Global Address Cleanse information

Characteristic	Description
	Global Address Cleanse icon
Use	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. The transform has options to enable generation of data quality statistics tables. These statistics provide insight into how the software cleansed and assigned your data by generating aggregated summary statistics and specific record-level statistics. **i 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, use the USA Regulatory Address Cleanse transform, which supports CASS. If you perform both address cleansing and data cleansing, the Global Address Cleanse transform typically comes before the Data Cleanse transform in the data flow.
Content objects	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.

Global Address Cleanse Report and Analysis [page 473]
To generate report data for the Global Address Cleanse transform, use the option in the Report and Analysis.

Data Quality Statistics for Data Cleanse, Global Address Cleanse, and Geocoder transforms [page 383]
To enable the generation of data quality statistics, make settings in the Data Quality Statistics Settings group.

Global Address Cleanse reference files [page 477]
Set the location for the Global Address Cleanse reference files, which includes Global, USA, or Canada reference files as applicable.

Global Address Cleanse Country ID Options [page 478]

Specify how the Global Address Cleanse transform uses country ID processes in the Country ID Options group.

Use the NW input fields for working with data from SAP applications.

Global Address Cleanse NW_PO_Box output fields [page 601]
The Global Address Cleanse transform populates the *NW_PO_BOX_* output fields only when you map them to NW input fields.

Related Information

Download Data Quality blueprints and other content objects [page 340]
Data Quality transforms [page 341]
Address Cleanse reference [page 757]
Data Quality Statistics for Data Cleanse, Global Address Cleanse, and Geocoder transforms [page 383]

3.5.9.1 Global Address Cleanse Report and Analysis

To generate report data for the Global Address Cleanse transform, use the option in the Report and Analysis.

Option	Description
Generate Report Data	Specifies whether to generate report data for this transform. • *YES*: Generates report data for this transform. • *NO*: Does not generate report data for this transform. *NO* is the default setting. → Tip If you do not need to generate reports, such as during testing, set this option to *NO* to improve performance.

3.5.9.2 Data Quality Statistics for Data Cleanse, Global Address Cleanse, and Geocoder transforms

To enable the generation of data quality statistics, make settings in the Data Quality Statistics Settings group.

The Data Quality Statistics Settings group appears in the Data Cleanse, Global Address Cleanse, and the Geocoder transforms.

The following table contains the options for the *Data Quality Statistics Settings* group. Because the group is applicable for more than one transform, the table lists the applicable transform for each option.

The software generates the following types of statistics for each transform:

- The Data Cleanse and the Global Address Cleanse transforms generate additional statistics and info codes about address cleansing results on the data.
- The Geocoder transform generates statistics and info codes that are unique to the Geocoding processes.

Option	Transform	Description
Generate Cleanse Address Record Info Table	Global Address Cleanse	Specifies whether the software generates the Cleanse Address Record Information Table. • *YES*: Generates the Cleanse Address Record Information Table. • *NO*: Does not generate the Cleanse Address Record Information Table. *NO* is the default setting. The Cleanse Address Record Information Table contains non-summary information about the results of the address cleanse process on the data. The information includes assignment information and assignment type for each address. The software generates this information only when you include the Global Address Cleanse transform for address data.
Generate Cleanse Change Info Table	Data Cleanse Global Address Cleanse	Specifies whether the software generates the Cleanse Change Information Table. • *YES*: Generates the Cleanse Change Information Table. • *NO*: Does not generate the Cleanse Change Information Table. *NO* is the default setting. The Cleanse Change Information Table contains non-summary information. It identifies areas of concern from your source data by the number of significant changes.

Option	Transform	Description
Generate Cleanse Component Info Table	Data Cleanse Global Address Cleanse	Specifies whether the software generates the Cleanse Component Information Table. • *YES*: Generates the Cleanse Component Information Table. • *NO*: Does not generate the Cleanse Component Information Table. *NO* is the default setting. **i Note** This option is available in SAP Data Services version 4.2.6 and later. The Cleanse Component Information Table contains non-summary information. It contains position information for each specified data element when one or both of the following occurs: • The transform parses the data element from an input field. • The transform writes the data element to an output field.
Generate Cleanse Info Codes Table	Data Cleanse Global Address Cleanse	Specifies whether the software generates Cleanse Information Codes Table. • *YES*: Generates the Cleanse Information Codes Table. • *NO*: Does not generate the Cleanse Information Codes Table. *NO* is the default setting. The Cleanse Information Codes table contains non-summary information. The information includes the distribution of information codes or a count of missing, suspect, or blank data.

Option	Transform	Description
Generate Cleanse Statistics Table	Data Cleanse Global Address Cleanse	Specifies whether the software generates the Cleanse Statistics Table. • *YES*: Generates the Cleanse Statistics Table. • *NO*: Does not generate the Cleanse Statistics Table. *NO* is the default setting. The Cleanse Statistics Table contains summarized information for each unique Entity ID. It provides a high-level insight into what the software changed during the cleansing process.
Generate Geocode Info Codes Table	Geocoder	Specifies whether to generate the Geocode Information Codes Table. • *YES*: Generates the Geocode Information Codes Table. • *NO*: Does not generate the Geocode Information Codes Table. *NO* is the default setting. The Geocode Information Codes Table contains non-summary information about why records were not assigned the highest level of latitude and longitude. The statistics are based on the information code data. **i Note** Not all existing information codes apply to the data quality statistics.

Option	Transform	Description
Generate Geocode Statistics Table	Geocoder	Specifies whether the software generates the Geocode Statistics Table. • *YES*: Generates the Geocode Statistics Table. • *NO*: Does not generate the Geocode Statistics Table. *NO* is the default setting. The Geocode Statistics Table contains totals for the following: • The number of rows in the data • The number of rows that were assigned a latitude and longitude.

Related Information

Data quality statistics common requirements [page 351]
Data quality statistics transform set up requirements [page 353]
Content information for data quality statistics tables [page 1404]

3.5.9.3 Global Address Cleanse reference files

Set the location for the Global Address Cleanse reference files, which includes Global, USA, or Canada reference files as applicable.

Reference files are directories required by the Global Address Cleanse transform to process your data.

Option	Description
Directory Path	Specifies the path to the location of your Global Address directory files. For easier setup, use the substitution variable `$$RefFilesAddressCleanse` as a value for this option. SAP Data Services has set the `$$RefFilesAddressCleanse` substitution variable for the default reference file location. The default reference file location is the path relative to where Data Services is installed. If you installed to a location other than the default location, edit the substitution parameter accordingly.

For information about downloading directories, see the latest Directories Update letters at https://uacp2.hana.ondemand.com/viewer/p/ADDRESSING_DIRECTORIES.

Related Information

Overview of substitution parameters

3.5.9.4 Global Address Cleanse Country ID Options

Specify how the Global Address Cleanse transform uses country ID processes in the Country ID Options group.

> **i Note**
>
> Reference to "country" or "countries" in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

When the transform can't determine a best country from the parsed and ranked countries, it uses the settings in the Country ID group to determine the country.

This option group is required.

Country ID group option descriptions

Option	Description
Country ID Mode	Specifies whether the transform uses the specified country name from the *Country Name* option or runs Country ID processing using the Global Address engine. • *CONSTANT*: Assumes all of your input data is for the specified country in *Country Name* and doesn't run Country ID processing. Choose this option only when all of your input data is from one country. Selecting this option can save processing time. • *ASSIGNED*: Runs Country ID processing. Choose this option when the input data contains addresses from more than one country.
Country Name	Specifies the country of destination. Select the applicable country or select *None*. Select *NONE* when you set the *Country ID Mode* to *ASSIGNED* and you don't want the transform to use a default country for records with unidentified countries. Instead, the transform uses the Global Address engine. Special considerations: • When you set the *Country ID Mode* to *CONSTANT*, ensure that you specify a country name. • If you set the *Country ID Mode* to *ASSIGNED*, and you specify a country for *Country Name*, the transform uses the specified country name instead of the Global Address engine when it can't identify a country.
Script Code	Specifies the ISO four-character script code for your data. • *CJKK*: Chinese, Japanese, and Korean • *CYRL*: Cyrillic • *GREK*: Greek • *LATN*: Latin • *NONE*: Attempts to identify the overall script of the input data as *CJKK*, *GREK*, *LATN*, or other.

Option	Description
Identify Localities	Specifies the locality resource the transform uses to identify candidate countries for addresses with no country information. • *MAJOR*: Uses the major localities list in addition to the country dictionary, region dictionary, and parsing dictionary. *MAJOR* is the default setting. • *EXTENDED*: Uses the extended list of localities in addition to the country dictionary, region dictionary, and parsing dictionary. • *ALL*: Uses the Universal Postal Union (UPU) all-world directory in addition to the country dictionary, region dictionary, and parsing dictionary. **i Note** The major and extended localities lists are included with the installation at no extra cost. The UPU all-world directory isn't included with the installation and is available for purchase. If you select *ALL*, but you don't own the UPU all-world directory, or the directory isn't located in the directory path you specified in the *Reference files* group, the transform uses the *EXTENDED* option and issues a warning. The transform requires that you set the *Country ID Mode* option to *ASSIGN* to use the *Identify Localities* option.

Option	Description

Enable Suggestion Lists	Specifies whether the transform outputs a ranked list of candidate countries in the form of a suggestion list.
	• *YES*: Generates a ranked list of candidate countries in a suggestion list. After you select a country from the list, the transform continues to process the record using the selected country.
	• *NO*: Doesn't generate a ranked list of candidate countries in a suggestion list, but uses the setting in *Country Name* as follows:
	  • When you select a country for the *Country Name* option, the transform assigns the record using that country.
	  • When you set *Country Name* to *NONE*, the transform outputs the record as unassigned and generates the applicable information code that indicates the country is unknown.
	NO is the default setting.
	i Note
	Software performance can be affected when you enable this option and your data contains a high number of records that don't contain country information.
	The transform generates suggestion list information for the *Enable Suggestion Lists* option in the following suggestion list components:
	• *ISO Country Code 2-Char*: Contains the two-character ISO country code.
	• *Sugg_Confidence_Score*: Contains the confidence score for the two-character ISO country code.
	The transform requires you to set the *Country ID Mode* option to *ASSIGN* to use this option.

Related Information

Global Address Cleanse information codes [page 809]
Global Address Cleanse Suggestion List Components [page 536]

3.5.9.5 Global Address Cleanse Engines group

Choose one or more engines to use in the Global Address Cleanse transform and set other country-specific options.

> **i Note**
>
> Reference to "country" or "countries" in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

Before you set options in this group, consider the countries represented in your data. For example, if your data doesn't contain addresses from Canada or USA, select the Global Address engine. The transform requires at least one engine.

This option group is required.

Engine group option descriptions

Option	Description
Canada	Specifies whether the transform uses the Canada engine. • *YES*: Uses the Canada engine for Global Address Cleanse processes. • *NO*: Doesn't use the Canada engine. *NO* is the default setting.
Dynamic Engine Init	Specifies whether engines that the transform uses are limited to the engines that it can initialize. • *YES*: Uses only the engines that it can successfully initialize. If an engine fails to initialize, the software issues a warning and the job continues. • *NO*: Uses all enabled engines. If an engine fails to initialize, the job fails. *NO* is the default setting.

Option	Description

Assign Country ID Results	Specifies how the transform determines the country to assign when there are multiple suggested candidate countries.
	• *SINGLE*: Assigns the address using the country with the highest confidence score from the list of candidate countries. When multiple countries have the same high confidence score, the transform takes one of the following actions:
	  • Generates a country suggestion list when country suggestion list is enabled.
	  • Outputs the record as unassigned and generates an info code that indicates the country is unknown.
	  SINGLE is the default setting.
	• *MULTIPLE*: Assigns an address for each candidate country using the applicable directory for each country. If the address assigns for only one of the candidate countries with a high confidence score, the transform assigns the address using that country. If the address assigns for multiple countries with equal high confidence scores, the transform takes one of the following actions:
	  • Generates a country suggestion list when country suggestion list is enabled.
	  • Outputs the record as unassigned and generates an info code that indicates the country is unknown.
	Requirements:
	• Candidate countries must have PAF-level address data to make complete assignments to the delivery point.
	• For countries that don't have PAF-level address data, Data Services uses the all world directory to make locality level assignments only.
	• You must place the applicable directories in the directory path set in the *Reference files* group.
	The transform requires that you set the *Country ID Mode* option in the *Country ID Options* group to *ASSIGN* to use this option.

Global Address	Specifies whether the transform uses the Global Address engine. • *YES*: Uses the Global Address engine for Global Address transform processes. • *NO*: Doesn't use the Global Address engine.

Option	Description
USA	Specifies whether the transform uses the USA engine. • *YES*: Uses the USA engine for Global Address transform processes. • *NO*: Doesn't use the USA engine.

3.5.9.6 Global Address Cleanse Country

Set a country name before you set options in the Standardization group.

> **i Note**
>
> Reference to "country" or "countries" in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

Option	Description
Country Name	Specifies the country to which the standardization options apply. The default is *GLOBAL*.

3.5.9.7 Global Address Cleanse Standardization Options

To standardize addresses for the selected country, set options in the Standardization Options group.

> **i Note**
>
> Reference to "country" or "countries" in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

> **! Restriction**
>
> BEST_PRACTICE values are not supported as dynamic options in DQM SDK and Data Services.

The Standardization Options settings apply to the country that you specify for the *Country Name* option in *Standardization Options* > *Country*.

Set these options once to apply to all source records by selecting *GLOBAL* for *Country Name*. Or, duplicate the Country group and specify a specific country for each group in the *Country Name* option.

The Standardization Options group is required and can be duplicated.

Standardization Options group

Option	Description
Address Line Alias	Specifies how to standardize the address line when the address contains alias address elements. Engines supported: Canada, Global Address, and USA. • *CONVERT*: Converts address lines based on official address line components instead of delivery address line components. • *PRESERVE*: Retains non-preferred data in address lines unless the data is incorrect. **i Note** Since December 31, 2013, Korea does not officially support the old Korean address system. The Korea reference data contains the old addresses as aliases to the new addressing system. SAP recommends that you convert South Korean land-lot addresses to their new road name addresses. Set the *Address Line Alias* and *Assign Locality* options to *CONVERT*.
Assign Locality	Specifies how to standardize the locality name. • *CONVERT*: Converts the locality name to the locality name preferred by the postal authority for the country. • *PRESERVE*: Preserves the input locality name unless it is incorrect. • *VALID*: Retains the input locality name unless it is not valid for mailing. If it is not valid for mailing, replaces it with the preferred locality name. **i Note** The *Translate Major Locality* option setting in the *Global Address* > *Country* section takes precedence over the setting in this option. **i Note** Since December 31, 2013, Korea does not officially support the old Korean address system. The Korea reference data contains the old addresses as aliases to the new addressing system. SAP recommends that you convert South Korean land-lot addresses to their new road name addresses. Set the *Address Line Alias* and *Assign Locality* options to *CONVERT*.

Option	Description
Capitalization	Specifies the casing of your address data. • *MIXED*: Converts data to initial capitals. For example, "MAIN STREET SOUTH" becomes "Main Street South." • *UPPER*: Converts data to full capitals. For example, "Main Street South" becomes "MAIN STREET SOUTH." • *BEST_PRACTICE*: Use initial or full capitals based on what the software sees as the most common for a country. See "BEST_PRACTICE" for more information. . **i Note** If you want consistent casing for your data, make sure that this option and the *Capitalization* setting in the Data Cleanse transform are the same.
Character Width Style	Specifies whether to standardize half-width and full-width characters. This option applies only to Chinese, Japanese, and Korean data. • *NORMAL_WIDTH*: Converts full-width Latin characters to half width and converts half-width Chinese, Japanese, and Korean characters to full width. • *HALF_WIDTH*: Converts all characters to half width. • *FULL_WIDTH*: Converts all characters to full width.
Correct Assigned Data	Specifies whether to use the parsed or corrected data for the assigned output fields of type Best. • *YES*: Populates the Best components with corrected data. • *NO*: Populates the Best components with parsed data. **i Note** If you choose *NO* for this option, the *Capitalization* option is the only available Standardization option for your assigned data.
Correct Unassigned Data	Specifies whether the Global Address Cleanse transform standardizes your unassigned data. • *YES*: Populates the Best components with corrected data. • *NO*: Populates the Best components with parsed data. **i Note** If you choose *NO* for this option, the *Capitalization* option is the only available Standardization option for your unassigned data.

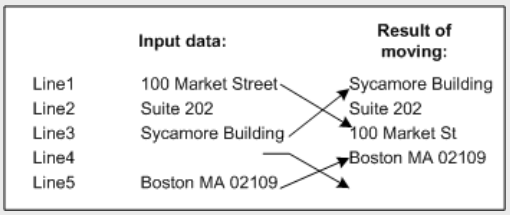
Option	Description
Country Style	Specifies how the software standardizes country data. • *ISO_2CHAR*: Standardizes country data to the two-character ISO code, such as AU, CA, or US. • *ISO_3CHAR*: Standardizes country data to the three-character ISO code, such as AUS, CAN, or USA • *ISO_3DIGIT*: Standardizes country data to the three-digit ISO code, such as 038, 124, or 840. • *NAME*: Standardizes country data to the full country name, such as Australia, Canada, or United States. • *PRESERVE*: Attempts to retain the country data from the input record, otherwise uses the corrected country value on output.
Directional Punctuation	Specifies whether to use punctuation in the abbreviated directional data. • *YES*: Outputs directionals with punctuation. For example, N. or S.W.. • *NO*: Outputs directionals without punctuation. For example, N, SW. • *PRESERVE*: If parsed input component contains punctuation, retains it on output with corrections applied when the transform does not find the input in the dictionary. For example, “N/W” on input is updated to “N.W.”. • *BEST_PRACTICE*: Punctuates the abbreviated directional data based on what the software sees as the most common for a country. See "BEST_PRACTICE" for more information.
Directional Style	Specifies whether to abbreviate directional data. • *LONG*: Uses fully spelled directionals such as "North," "South," "East," "West." • *PRESERVE*: Preserves the style used in the input record. • *SHORT*: Uses abbreviated directionals such as "N," "S," "E," "W." • *BEST_PRACTICE*: Abbreviates directional data based on what the software sees as the most common for a country. See "BEST_PRACTICE" for more information.

Option	Description
European Postcode Prefix	For mail generated and distributed inside Europe, adds the one- to three-character European Postcode prefix, followed by a dash. • *YES*: Adds the European Postcode prefix. • *NO*: Does not add the European Postcode prefix. • *PRESERVE*: If the transform finds a European Postcode prefix on input, retains the European Postcode prefix. **Example** In the following address, the D- is the European Postcode extension: Hallesches Ufer 32–38 D-10963 Berlin Germany **Note** The *European Postcode Prefix* is for mail distributed from one European country to another European country.
Extra Lines	Specifies what to do with extra lines of non address data. • *PRESERVE*: Attempts to retain the extra line of non-address data in the general location in which it was input. • *REMOVE*: Does not include any extra line of non-address data in the standardized lines or multiline fields. • *PREFERRED*: Places all populated Extra fields above or below the multiline fields and standardized input lines based on the country data being processed. For example, Extra fields for Japan are located below the standardized lines.
Format Assigned Data	Specifies whether to format your assigned data based on the preferred address format of the country. For example, the format for Germany is: {Primary_Name1} {Primary_Number} {Postcode1} {Locality} {Country} • *YES*: Formats the assigned data. • *NO*: Does not format the assigned data and leaves it in the location in which it was input. If data is added to the record, the transform bases the placement on the format string.

Option	Description
Format Unassigned Data	Specifies whether to format unassigned data based on the preferred address format for the country. For example, the format for Germany is: {Primary_Name1} {Primary_Number} {Postcode1} {Locality} {Country} • *YES*: Formats the unassigned data. • *NO*: Does not format the unassigned data and leaves it in the line in which it was input.
Convert Latin Output To US ASCII	If a character conversion is available for Latin script records, converts any extended ASCII characters in the Best component to the US ASCII character set. • *YES*: Converts extended ASCII characters. • *NO*: Does not convert extended ASCII characters. *NO* is the default setting. • *BEST_PRACTICE*: Convert or not based on what the software sees as the most common for a country. See "BEST_PRACTICE" for more information. The transform leaves any extended ASCII characters for which there is no conversion as is. For example, the transform leaves the degree symbol or inverted exclamation and question marks as is. **Example** With the input street name “Østerbrogade”, preserve the local character or convert it to the international data format “Osterbrogade” in the cleansed output.
Include Country	Specifies whether to include country names in standardized lines or multiline fields. • *YES*: Includes country name. • *NO*: Does not include country name. • *PRESERVE*: If the transform finds the country name on input, retains the country name.

Option	Description
Include Locality Addition (Germany only)	Specifies whether the *LOCALITY1_FULL* output field contains both the *LOCALITY1_NAME* and the *LOCALITY1_ADDITION* information in the standardized output. • *YES*: Includes both the locality name and the locality addition information only when it is valid. • *NO*: Does not include the locality addition. • *PRESERVE*: If the transform finds a locality addition on input, includes the locality name and the locality addition in the standardized output only when it is valid. When there is no locality addition on input but the software finds a locality addition in the postal directory, the software includes the locality name and the locality addition in the standardized output only when it is valid. *Preserve* is the default setting. **i Note** When the software recognizes the locality addition, for example, the software finds it in the postal directory, but it is flagged as invalid by Deutsche Post, the software does not include the *LOCALITY1_ADDITION* in the standardized output regardless of a *YES* or *PRESERVE* setting. **i Note** If you set the *Locality Name Style* option to *SHORT*, the transform does not output the *LOCALITY1_ADDITION* to the standardized output, even if it is valid, regardless of a *YES* or *PRESERVE* setting. **i Note** If you set the *Transaslate Major Locality* option in the Global Address Country Options group to English, the transform does not output the *LOCALITY1_ADDITION* field to the standardized output, even if it is valid, regardless of a *YES* or *PRESERVE* setting.

Option	Description
Include Unused Address Line Data	Specifies whether to output the unused address line data for standardized lines and multiline fields. • *YES*: Outputs the unused address line data in the remainder fields *ADDRESS_LINE_REMAINDER1* to 4. For example, when input is 100 Main St Red House, outputs "Red House" to the address line remainder output field. • *NO*: Does not output the unused address line data. This option affects unused address data classified as remainder. This option does not affect unused address data classified as extra.
Include Unused Lastline Data	Specifies whether to output the unused last line data for standardized lines and multiline fields: • *YES*: Outputs the unused last line data. • *NO*: Does not output the unused last line data.
Locality Name Style	Specifies the format for locality data in the *LOCALITY1_NAME* output field for addresses. This option applies to German addresses. • *PRESERVE*: Preserves the locality data format as it was input. *PRESERVE* is the default setting. • *SHORT*: If the transform finds the locality data in the reference data, it outputs the locality data in the abbreviated version. **i Note** To use the short locality name style, set the *Address Line Alias* option to *CONVERT*. **i Note** This option overrides the setting in the *Include Locality Addition* option. **i Note** If the *Translate Major Locality* option is set to translate the *LOCALITY1* output field, the transform does not output short locality regardless of the setting in the *Locality Name Style* option.

Option	Description
Move Multiline Data	Determines the position of blank lines in output addresses. • *BOTTOM*: Moves blank lines to the top and shifts the data to the bottom of the address block. • *NO*: Does not rearrange any lines, blank or otherwise. • *TOP*: If there are any blank lines, the transform moves them to the bottom of the address block and shifts the data to the top of the block. **Example** Top: 

Option	Description
Output Country Language	Specify which language and script to use on output for the country name, not the entire record. *PRESERVE*: Preserves the input country name on output. If you don't select *PRESERVE*, select one of the following option language in which to output the country name: • *CATALAN - LATIN* • *CHINESE - HANI* • *DANISH - LATIN* • *DUTCH- LATIN* • *ENGLISH- LATIN* • *FINNISH - LATIN* • *FRENCH - LATIN* • *GREEK - GREEK* • *GERMAN - LATIN* • *HUNGARIAN - LATIN* • *ITALIAN - LATIN* • *JAPANESE - HANI* • *JAPANESE - KANA* • *KOREAN - HANG* • *NORWEGIAN - LATIN* • *POLISH - LATIN* • *PORTUGUESE - LATIN* • *RUSSIAN - CYRILLIC* • *SPANISH - LATIN* • *SWEDISH - LATIN*
Postal Phrase Punctuation	If you choose *SHORT* for the *Postal Phrase Style* option, this option specifies whether to use punctuation in the postal abbreviation. • *YES*: Includes punctuation in postal abbreviations. For example, “P.O. Box”. • *NO*: Does not insert any punctuation for postal abbreviations. For example, “PO Box”. • *PRESERVE*: If there is punctuation in the input parsed component, returns the corrected punctuation when the input is not found in the dictionary. • *BEST_PRACTICE*: Punctuates based on what the software sees as the most common for a country. See "BEST_PRACTICE" for more information.

Option	Description
Postal Phrase Style	Specifies whether to abbreviate postal phrases. • *LONG*: Outputs the fully spelled out postal phrase. For example, Post Office Box. • *PRESERVE*: If the transform finds a postal phrase on input, it retains the style of the postal phrase on output. • *SHORT*: Outputs the abbreviated postal phrase. For example, PO Box. The transform uses the setting in the *Postal Phrase Punctuation* option to determine the punctuation for this option. • *BEST_PRACTICE*: Uses the style the software sees as the most common for a country. See "BEST_PRACTICE" for more information.
Primary Type Punctuation	If you choose *SHORT* for the *Primary Type Style* option, this option specifies whether to use punctuation in the primary type abbreviations. • *YES*: Includes punctuation at the end of primary type abbreviations. For example, “St.”. • *NO*: Does not include punctuation at the end of primary type abbreviations, regardless of whether the input contains punctuation. For example, “St”. • *PRESERVE*: If there is punctuation in the input parsed component, returns the corrected punctuation when the transform does not find the input in the dictionary. • *BEST_PRACTICE*: Punctuates based on what the software sees as the most common for a country. See "BEST_PRACTICE" for more information.
Primary Type Style	Specifies the style for primary type address elements. • *LONG*: Uses fully spelled primary types such as Street, Avenue, Road, or Strasse. • *PRESERVE*: Retains the style used in the input record. • *SHORT*: Uses abbreviated primary type such as St, Ave, Rd, or Str. The transform determines the punctuation for this option by the setting in the *Primary Type Punctuation* option. • *BEST_PRACTICE*: Uses the style the software sees as the most common for a country. See "BEST_PRACTICE" for more information.

Option	Description
Region Style	Specifies whether to abbreviate the region name such as state or province. • *LONG*: Uses the fully spelled region name. For example, California or Ontario. • *PRESERVE*: Retains the style used in the input record. • *SHORT*: Abbreviates the region name. For example, CA or ON. • *BEST_PRACTICE*: Abbreviate the region name based on what the software sees as the most common for a country. See "BEST_PRACTICE" for more information.
Remove Address Apostrophes	Specifies whether to include apostrophes in certain street names such as DE L' or D'. • *YES*: Retains apostrophes in street names if it was present on input, for example, Rue D'Abbeville. • *NO*: Removes apostrophes in street names, for example, Rue D Abbeville.
Secondary Description Punctuation	If you choose *SHORT* for the *Secondary Description Style* option, the *Secondary Description Punctuation* option specifies whether to use punctuation in the abbreviation. • *YES*: Uses punctuation in the abbreviation, for example, “Apt.”. • *NO*: Does not use punctuation, for example, “Apt”. • *PRESERVE*: If there is punctuation in the input parsed component, returns the corrected punctuation when the transform does not find the input in the dictionary. • *BEST_PRACTICE*: Punctuates abbreviations based on what the software sees as the most common for a country. See "BEST_PRACTICE" for more information.
Secondary Description Style	Specifies whether to abbreviate the secondary description, such as a unit or an apartment. • *LONG*: Uses the fully spelled secondary description. For example, “Apartment”. • *PRESERVE*: Retains the style used in the input record. • *SHORT*: Abbreviates the secondary description. For example, “Apt”. The transform uses the setting in the *Secondary Description Punctuation* option to determine whether to use punctuation for this option. • *BEST_PRACTICE*: Uses the style the software sees as the most common for a country. See "BEST_PRACTICE" for more information.

Option	Description
Secondary Number Style	Applicable for Canada and New Zealand addresses. Specifies the format of the secondary number such as a suite or apartment number. • *ATTACHED*: Converts all secondary ranges to the attached format in which the secondary number is prepended to the primary number and separated with a delimiter. For example, for Canada addresses, places a dash between the secondary and primary range: 5–100 Main St. • *PRESERVE*: Preserves the style of the address as it was input. • *UNATTACHED*: Converts all secondary ranges to the unattached format. For example, for Canada, it places the unit designator at the end of the primary address: 100 Main St Suite 5.

Option	Description
Street Name Style	For the Netherlands only. Specifies the format for street data for addresses in the Netherlands. • *PRESERVE*: Preserves the street data format as it was input. **Example** For example: Annelien Kappeyne Van de Coppellostr 2 Herten Limburg 6049 HD • *SHORT*: Outputs street data in the format preferred by the postal authority. For the Netherlands, the Short option returns a street address with a maximum of 24 characters in mixed case. **Example** For example: A K vd Coppellostr 2 Herten Limburg 6049 HD **Note** To use the short street name style, set the *Address Line Alias* option to *Convert*. **Note** The *Capitalization* option in the Global Address Cleanse transform overrides the setting in the *Street Name* option.
Use Local Primary Type Style	Applies to Austria, Germany, and Switzerland. Specifies whether to use the type style for primary address components that is present in the address data. • *YES*: Uses the Primary Type Style that is present in the address data. • *NO*: Uses the Primary Type Style specified in the *Primary Type Style* option. When you set this option to *YES*, the transform ignores the setting in the *Primary Type Style* option.

Option	Description
Use Postal Country Name	Specifies which country data is output for countries that receive their postal service from another country. • *YES*: Uses the postal country name. • *NO*: Uses the territory country name. **Example** When you use the USA engine for addresses from U.S. territories, the software populates the *COUNTRY* field with the name of the postal country, United States. The software bases the style of the *COUNTRY* field on the *Country Style* option setting. If the country does not have a postal country, this setting does not change the output.

3.5.9.7.1 BEST_PRACTICE

The BEST_PRACTICE value (available for some Global Address Cleanse Standardization options) tells the software to use what it sees as the most common setting for a country.

The table shows what setting the software uses for each country when Global Address Cleanse Standardization options use the BEST_PRACTICE value.

! Restriction

BEST_PRACTICE values are not supported as dynamic options in DQM SDK and Data Services.

BEST_PRACTICE settings

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
AD	Andorra	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
AE	United Arab Emirates	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
AF	Afghanistan	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
AG	Antigua and Barbuda	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
AI	Anguilla	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
AL	Albania	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
AM	Armenia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
AO	Angola	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
AQ	Antarctica	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
AR	Argentina	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
AS	American Samoa	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT
AT	Austria	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
AU	Australia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT
AW	Aruba	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
AX	Åland Islands	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
AZ	Azerbaijan	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
BA	Bosnia and Herzegovina	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
BB	Barbados	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
BD	Bangladesh	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
BE	Belgium	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	LONG
BF	Burkina Faso	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
BG	Bulgaria	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
BH	Bahrain	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
BI	Burundi	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
BJ	Benin	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
BL	Saint Barthélemy	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
BM	Bermuda	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
BN	Brunei Darussalam	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
BO	Plurinational State of Bolivia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
BQ	Bonaire, Sint Eustatius and Saba	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
BR	Brazil	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
BS	Bahamas	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
BT	Bhutan	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
BV	Bouvet Island	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
BW	Botswana	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
BY	Belarus	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
BZ	Belize	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
CA	Canada	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
CC	Cocos (Keeling) Islands	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT
CD	Democratic Republic of the Congo	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
CF	Central African Republic	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
CG	Congo	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
CH	Switzerland	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
CI	Côte D'ivoire	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
CK	Cook Islands	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
CL	Chile	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
CM	Cameroon	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
CN	China	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
CO	Colombia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
CR	Costa Rica	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
CU	Cuba	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
CV	Cabo Verde	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
CW	Curaçao	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
CX	Christmas Island	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT
CY	Cyprus	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
CZ	Czechia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
DE	Germany	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
DJ	Djibouti	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
DK	Denmark	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
DM	Dominica	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
DO	Dominican Republic	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
DZ	Algeria	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
EC	Ecuador	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
EE	Estonia	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
EG	Egypt	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
EH	Western Sahara	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
ER	Eritrea	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
ES	Spain	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
ET	Ethiopia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
FI	Finland	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
FJ	Fiji	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
FK	Falkland Islands (Malvinas)	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
FM	Federated States of Micronesia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT
FO	Faroe Islands	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
FR	France	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
GA	Gabon	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
GB	United Kingdom	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
GD	Grenada	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
GE	Georgia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
GF	French Guiana	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
GG	Guernsey	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
GH	Ghana	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
GI	Gibraltar	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
GL	Greenland	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
GM	Gambia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
GN	Guinea	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
GP	Guadeloupe	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
GQ	Equatorial Guinea	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
GR	Greece	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
GS	South Georgia and the South Sandwich Islands	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
GT	Guatemala	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
GU	Guam	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT
GW	Guinea-Bissau	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
GY	Guyana	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
HK	Hong Kong	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
HM	Heard Island and McDonald Islands	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT
HN	Honduras	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
HR	Croatia	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
HT	Haiti	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
HU	Hungary	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
ID	Indonesia	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	LONG
IE	Ireland	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
IL	Israel	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
IM	Isle of Man	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
IN	India	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	LONG
IO	British Indian Ocean Territory	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
IQ	Iraq	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
IR	Islamic Republic of Iran	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
IS	Iceland	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
IT	Italy	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
JE	Jersey	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
JM	Jamaica	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
JO	Jordan	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
JP	Japan	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
KE	Kenya	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
KG	Kyrgyzstan	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
KH	Cambodia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
KI	Kiribati	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
KM	Comoros	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
KN	Saint Kitts and Nevis	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
KP	Democratic People's Republic of Korea	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
KR	Republic of Korea	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
KW	Kuwait	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
KY	Cayman Islands	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
KZ	Kazakhstan	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
LA	Lao People's Democratic Republic	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
LB	Lebanon	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
LC	Saint Lucia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
LI	Liechtenstein	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
LK	Sri Lanka	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	LONG
LR	Liberia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
LS	Lesotho	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
LT	Lithuania	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
LU	Luxembourg	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
LV	Latvia	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
LY	Libya	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
MA	Morocco	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
MC	Monaco	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
MD	Republic of Moldova	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
ME	Montenegro	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
MF	Saint Martin (French Part)	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
MG	Madagascar	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
MH	Marshall Islands	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT
MK	North Macedonia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
ML	Mali	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
MM	Myanmar	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
MN	Mongolia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
MO	Macao	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
MP	Northern Mariana Islands	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT
MQ	Martinique	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
MR	Mauritania	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
MS	Montserrat	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
MT	Malta	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
MU	Mauritius	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
MV	Maldives	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
MW	Malawi	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
MX	Mexico	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT
MY	Malaysia	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	LONG
MZ	Mozambique	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
NA	Namibia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
NC	New Caledonia	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
NE	Niger	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
NF	Norfolk Island	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT
NG	Nigeria	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
NI	Nicaragua	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
NL	Netherlands	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
NO	Norway	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
NP	Nepal	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
NR	Nauru	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
NU	Niue	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
NZ	New Zealand	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
OM	Oman	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
PA	Panama	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
PE	Peru	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
PF	French Polynesia	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
PG	Papua New Guinea	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
PH	Philippines	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
PK	Pakistan	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	LONG
PL	Poland	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
PM	Saint Pierre and Miquelon	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
PN	Pitcairn	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
PR	Puerto Rico	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	SHORT	NO	SHORT
PS	State of Palestine	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
PT	Portugal	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
PW	Palau	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT
PY	Paraguay	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
QA	Qatar	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
RE	Réunion	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
RO	Romania	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
RS	Serbia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
RU	Russian Federation	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
RW	Rwanda	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
SA	Saudi Arabia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
SB	Solomon Islands	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
SC	Seychelles	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
SD	Sudan	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
SE	Sweden	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
SG	Singapore	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
SH	Saint Helena, Ascension and Tristan da Cunha	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
SI	Slovenia	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
SJ	Svalbard and Jan Mayen	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
SK	Slovakia	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
SL	Sierra Leone	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
SM	San Marino	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
SN	Senegal	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
SO	Somalia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
SR	Suriname	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
SS	South Sudan	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
ST	Sao Tome and Principe	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
SV	El Salvador	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
SX	Sint Maarten (Dutch Part)	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
SY	Syrian Arab Republic	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
SZ	Eswatini	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
TC	Turks and Caicos Islands	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
TD	Chad	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
TF	French Southern Territories	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
TG	Togo	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
TH	Thailand	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	LONG
TJ	Tajikistan	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
TK	Tokelau	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
TL	Timor-Leste	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
TM	Turkmenistan	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
TN	Tunisia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
TO	Tonga	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
TR	Turkey	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
TT	Trinidad and Tobago	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
TV	Tuvalu	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
TW	Province of China Taiwan	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
TZ	United Republic of Tanzania	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
UA	Ukraine	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
UG	Uganda	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
UM	United States Minor Outlying Islands	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT
US	United States	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT
UY	Uruguay	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
UZ	Uzbekistan	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
VA	Holy See	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
VC	Saint Vincent and the Grenadines	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
VE	Bolivarian Republic of Venezuela	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
VG	British Virgin Islands	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
VI	U.S. Virgin Islands	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	SHORT	NO	SHORT
VN	Viet nam	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
VU	Vanuatu	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
WF	Wallis and Futuna	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
WS	Samoa	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
YE	Yemen	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT

Country Code (ISO)	Country Name	Capitalization	Convert Latin Output To US ASCII	Directional Punctuation	Directional Style	Postal Phrase Punctuation	Postal Phrase Style	Primary Type Punctuation	Primary Type Style	Region Style	Secondary Description Punctuation	Secondary Description Style
YT	Mayotte	UPPER	NO	NO	LONG	NO	SHORT	NO	LONG	LONG	NO	SHORT
ZA	South Africa	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
ZM	Zambia	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT
ZW	Zimbabwe	UPPER	NO	NO	SHORT	NO	SHORT	NO	SHORT	LONG	NO	SHORT

3.5.9.8 Global Address Cleanse Canada group

Set Canada-specific processing options in the Canada group.

Option	Description
Disable Certification	Specifies whether to disable Canada address certification. • *YES*: Processes Canada addresses without including certification, and also processes with non-poc (Point of Call) directories for non-mailing purposes. *YES* is the default setting. > **i Note** > Does not print the SERP report. Any list that you create with certification disabled cannot be used for mailing. • *NO*: Processes Canada addresses using processes required for SERP certification. Includes using POC directories and generating the SERP report. Any list that you create with certification enabled can be used for mailing. > **i Note** > If you set this option to *YES*, also set the following certification options to *YES*: > • *Enable LVR Rule* > • *Enable Rural Rule* > • *Postcode Only Search* > • *Postcode No Match Search*
Dual Address	Specifies transform behavior when there is an input dual address. • *POSITION*: Selects an address based on the position of the address in the address block. The Canada engine tries to validate the address that is closest to the lower left corner of the address block. The address could be the postal or the street address based on how the data is input. *POSITION* is required for SERP certification. • *POSTAL*: Tries to validate based on the postal address. If that fails, tries again based on the street address. • *STREET*: Tries to validate based on the street address. If that fails, tries again based on the postal address, such as rural route or PO box.

Option	Description
Enable LVR Rule	Specifies whether to enable the Canada Post LVR (Large Volume Receiver) postal code rule. The rule states that the LVR postal code is valid and cannot be changed to match other address components. Canada Post permits correction of LVR postal codes only when a unique address can be determined without changing the postal code. • *YES*: Regards any LVR address as assigned, even when the address line is so flawed that a match to the postal directory is impossible. Select *YES* to qualify for SERP certification. • *NO*: Disables the LVR postal code rule. Reports LVR addresses as unassigned when the address line is flawed.
Enable Rural Rule	Specifies whether to enable the Canada Post Rural rule. The rule states that any address with a valid rural postal code is considered valid. Rural postal codes have a zero in the second position. • *YES*: Regards any rural address as valid, even if the address line is so flawed that a match to the postal directory is impossible. Select *YES* to qualify for SERP certification. • *NO*: Reports a rural address as invalid when the address line is bad. Canada Post states that you should not correct rural postal codes. However, the transform always attempts to correct all other address components except for the rural postal code. The rule applies under the following circumstances: • The address line is empty or contains bad data. • The address does not have a postal code or has an incorrect postal code. • The locality (city) has just one postal code associated with it that is a rural postal code.
Output Address Language	Specifies the language for the output records. • *CONVERT*: Uses French for records in Quebec, and English for records in other regions or provinces. • *ENGLISH*: Converts output records to English. • *FRENCH*: Converts output records to French. • *PRESERVE*: Detects the input language and preserves that language upon output, ignoring the region or province. *PRESERVE* is the default setting.

Option	Description
Parse Only	Specifies whether the transform should parse input records into discrete components, but not perform a lookup in the postal directories. • *YES*: Parses records into discrete components, but does not perform a lookup in the postal directories. When you select *YES*, processing is fast, but parsing results are unverified. • *NO*: Parses records into discrete components and performs a lookup in the postal directories. When you select *NO*, processing may be slower, but parsing results are verified when the appropriate reference data is available.

Postcode No Match Search	Specifies whether the transform should search the postal directories when the address line can be assigned, but it doesn't match the incoming postal code.
	• *YES*: Searches the postal directories when the address line can be assigned, but it doesn't match the incoming postal code. Select *YES* to qualify for SERP certification. When you select *YES*, the transform searches the postal directories to determine the following:
	  • If the incoming address line is a PO Box address, the postal code cannot be a valid postal code for an LVR (Large Volume Receiver), firm, or a civic (street) address, such as 100 Main St.
	  • If the incoming address line is a civic (street) address, the postal code cannot be a valid postal code for an LVR PO Box address.
	  SERP rules state that, if either one of these circumstances exists, Data Services cannot assign the address.
	• *NO*: Does not search the postal directories when the address line can be assigned, but it doesn't match the incoming postal code. *NO* is the default setting.

Option	Description
Postcode Only Search	Specifies whether the transform should search for the address based on postal code only, and attempt to find a street record that contains the range. • *YES*: Performs a postcode-only search, and attempts to find a street record that contains the range. Select *YES* to qualify for SERP certification. • *NO* Does not perform a post-code only search. In some cases, the result is a better address line. In other cases, the transform more reliably detects that it cannot assign an address line. This option affects assignment when the input address line is badly incomplete. For example, when the address includes a range but no street name. When you set this option to *YES* and the transform finds only one street record that contains the range, the transform assigns the address line from the postal code. A postal-code-only search is time consuming. Therefore, disabling this search could improve your processing time.
Postcode Priority Over Street	Specifies whether the transform weighs the postcode and street equally or whether street has priority over postcode. • *YES*: Assigns by placing more weight on the address line when breaking a tie. This assignment makes sense when you consider how common data-entry errors are for postal codes. • *NO*: Assigns by placing equal weight on the address line and the postal code when breaking a tie. Select *NO* to qualify for SERP certification. This option setting is important when the transform is trying to break a tie between two possible assignments: • A near match on address line. • An exact match on postal code.
Unit Description	Specifies the unit description in English: • *APARTMENT*: Uses “Apartment” as the default unit designator. • *DEFAULT*: Uses the default unit designator. • *UNIT*: Uses “Unit” as the default unit designator.

Option	Description
Use Firm To Assign	Specifies whether the transform uses the firm information to make an assignment and whether to display the firm in suggestion lists. • *YES*: Uses and displays the firm. *YES* is the default setting. • *NO*: Does not use or display the firm.

3.5.9.8.1 Global Address Cleanse Canada Report Options

To output the SERP required Statement of Address Accuracy report information, complete the options in the Canada engine report group.

Complete the options in this group when you prepare a mailing list for Canada Post certification.

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 CPC number that is located on the Canada Post Contract (up to 15 characters).

3.5.9.8.2 Global Address Cleanse Canada Suggestion List options

Enable suggestion lists and set how the transform presents suggestions for your Canada address data.

Option	Description
Address Lines Match Minimum	Specifies the similarity score required for address-line suggestions. Type a value from 0 through 80. The similarity score determines what suggestions the software returns in the list. A higher value requires the suggestion to be more similar to the input address than a lower value.
Address Range	Specifies a span around the input primary address range for which to return suggestions. Maximum value is 5000. Enter 0 to set no limits to the ranges that the transform returns in suggestion lists. By setting an address range, you limit the suggestions returned to be within the range. Example *Address Range* = 500 **Input address:** 1000 Pine St. **Suggestion list range:** 750–1250 Pine Street

Option	Description
Combine Overlapping Ranges	Specifies whether individual suggestions with overlapping ranges are combined. • *YES*: Ignores gaps and overlaps in ranges and combines individual suggestions. • *NO*: Does not combine individual suggestions when they have gaps or overlapping ranges. Set this option to *YES* if you want to limit the number of total suggestions presented to your user. However, you might not see gaps of invalid ranges that would be apparent if you set the option to *NO*. **Example** The software presents the following suggestions when you set this option to *NO*: 1000–1099 Maple Ave 1100–1199 Maple Ave The software presents the following suggestions when you set this option to *YES*: 1000–1199 Maple Ave
Enable Suggestion Lists	Specifies whether the software generates suggestion lists. • *YES*: Generates suggestion lists when assignment candidates are present. • *NO*: Does not generate suggestion lists.
Lastlines Match Minimum	Specifies the minimum similarity score required for lastline suggestions. The score then determines which suggestions are returned in the list. Type a value from 0 through 80. A higher number indicates that the suggestion must be more similar to the input to be returned as a possible suggestion.
Max Number Address Lines	Specifies the maximum number of address line suggestions that the software generates. Type a value from 2 through 50. Set this option, for example, to limit the size of SOAP documents that the Web service sends or to limit the maximum number of suggestions that users choose from.

Option	Description
Max Number Lastlines	Specifies the maximum number of lastline suggestions that the software generates. Type a value from 2 through 15. Set this option, for example, to limit the size of SOAP documents that the Web service sends or to limit the maximum number of suggestions that users choose from.

3.5.9.9 Global Address Cleanse Country Options

To set options for specified countries, set country options in the Global Address group.

> **i Note**
>
> Reference to "country" or "countries" in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

Specify a country and then set address assignment options for the specified country.

Option	Description
Country Name	Specifies the country to apply the standardization options. • *GLOBAL (APPLY TO ALL COUNTRIES)*: Applies country standardization options to all countries in your data. • *Specific Country*: To apply Country Options to a specific country, choose a specific country from the dropdown list.

Retain Postcode if Valid Format	Specifies to process blank, valid, and invalid input postcodes.
	
	• *YES*: Retains input postcode unless:
	  • It is in an invalid format for the country
	  • There's a single postcode match
	  • There's a possible intelligent match
	• *NO*: Updates output postcode if:
	  • There's a single postcode match
	  • There's a possible intelligent match
	  Otherwise, retains input postcode. *NO* is the default setting
	
	i Note
	The transform populates the postcode when the input postcode is blank, and it finds a single postcode match.

Option	Description
Dual Address	Specifies the action to take when the Global Address engine encounters a dual address. • *POSITION*: Selects an address based on the arrangement of the input data. The transform attempts to assign the first address found in the input data. If it can't assign the first address, it attempts to assign the next address it finds. • *POSTAL*: Attempts to validate based on the postal address. If that fails, attempts again based on the street address. • *STREET*: Attempts to validate based on the street address. If that fails, attempts again based on the postal address.

Option	Description

Disable Certification	Specifies whether to perform noncertified or certified processing of addresses for Australia, France, or New Zealand.
	
	If the country is Australia:
	
	• *YES*: Enables noncertified features and extends the directory expiration for nonmailing purposes. More information:
	  • Extends the directory expiration period for up to 14 months from the date the directories were created.
	  • Processing with expired directory data is allowed when you aren't planning to use the records for AMAS mailing purposes.
	  • Processing with expired directory data is ideal for data warehousing industries, for example. However, when you select *YES*, you can't print the AMAS report.
	  • You can't use any lists created with expired directories for postage discounts.
	• *NO*: Disables noncertified features and uses the most current directory information. Enables printing of the AMAS report.
	
	If the country is France:
	
	• *YES*: Enables noncertified processing of France addresses for nonmailing purposes.
	• *NO*: Disables noncertified processing of France addresses for mailing purposes. More information:
	  • Removes all punctuation except for firm data.
	  • Removes all accent characters (allows only A-Z and 0–9).
	  • Returns address in 6 lines.
	
	ℹ Note
	To return the address in all upper case, set the Capitalization option under *Standardization Options* > *Country* > *Options* > to *Upper*.
	
	If the country is New Zealand:
	
	• *YES*: Enables noncertified processing of New Zealand addresses and allows processing with expired directories for nonmailing purposes. Disables printing the SOA Report.

Option	Description
	> **i Note** > Don't use any list for mailing that you process with certification disabled. - *NO*: Enables certified processing for New Zealand and enables printing of the SOA report. > **i Note** > If the reference data is missing or older than the certification requirements for any of the following countries, the software issues a warning: > - Australia > - France > - New Zealand > If the software issues the message for data that isn't required for the country you're processing, ignore the message. To avoid receiving this message, replicate the country options group for each country. Then set the *Disable Certification* option to *NO* only for the relevant country. > When you set a Country Options group for each country, the software uses all other options in the country-specific group in place of the options you set in the Global group.
Output Address Script	Specifies how a native script is output. Supports the following countries: - Chinese when the input character script is CJKK - Russian when the input character script is CYRL Options include the following: - *LATIN*: Transliterates the native script and outputs the data as Latin. - *PRESERVE*: Outputs the data in the native script. > **i Note** > The setting in the *Translate Major Locality* option takes precedence over the setting in the *Output Address Script* option for the locality.

Option	Description
Translate Major Locality	Specifies whether to translate the *LOCALITY1* output field for major localities. • *ENGLISH*: Translates the output to English, if available. • *PRESERVE*: Outputs the locality data as it is input. *PRESERVE* is the default setting. **i Note** The setting in the *Translate Major Locality* option takes precedence over the setting in the *Assign Locality* option. The setting in the *Translate Major Locality* option also takes precedence over the setting in the *Output Address Script* option for the locality.
Use Firm to Assign	Specifies whether to use the firm name to make an assignment and to display the firm name in a suggestion list. *YES*: Uses and displays the firm. *YES* is the default setting. *NO*: Doesn't use or display the firm.
Processing Timeout	Specifies the timeout duration in milliseconds. After the duration is exceeded, the software stops processing and issues the information code 7000. You can use a substitution variable for this option. Enter *0* to disable the timeout. *0* is the default setting. Enter a number greater than zero to set the amount of time to wait before the software stops processing. **i Note** SAP recommends that you set the *Processing Timeout* at or above 100 milliseconds.
Suppress Suggestion List With no Range	Specifies whether the software generates a suggestion list when it finds an "S/N" or no range on input, and the only records in the reference data are range records. • *YES*: Doesn't generate a suggestion list. Returns information code 3030 and makes a primary name level match assignment. • *NO*: Generates a suggestion list with the range values in the reference data record. *NO* is the default setting

3.5.9.10 Global Address Cleanse Suggestion List Options

Enable suggestion lists and set how the transform presents suggestions for your global addresses.

> **i Note**
>
> Reference to “country” or “countries” in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

The settings that you make in the Suggestion List Options group are for global addresses only. The settings don’t affect how the software processes addresses from the United States or Canada.

Option	Description
Enable Suggestion Lists	Specifies whether the transform generates suggestion lists for global addresses. • *YES*: Generates suggestion lists when assignment candidates are present. • *NO*: Doesn’t generate suggestion lists.

Option	Description
Enable Lastline Drilldown	Specifies whether you can drill down to the next level of the lastline component in suggestion lists after selecting a parent component. • *YES*: Enables drilling down to the next level of lastline components in suggestion lists after selecting a parent component. • *NO*: Doesn't enable drilling down to the next level of lastline components in suggestion lists after selecting a parent component. **Example** Given an input country, a suggestion list provides a list of available regions. After you select one of the regions, a suggestion list provides a list of available locality1 options. After you select one of the localities, a suggestion list provides a list of available locality2 options. When you enable this option, ensure that you set the *Enable Suggestion Lists* option to *YES*. **Example** The Canada and USA engines don't support the *Enable Lastline Drilldown* option. A suggestion list using the *Enable Lastline Drilldown* option is the same as a regular lastline suggestion list, except that the suggestion list doesn't include postcode or address type components. Also, the expected input for lastline drilldown is country. All other input fields are blank. For example, input files such as locality, region, postcode, firm, and multiline are blank. **Note** Ensure that your input field combination complies with one of the accepted formats: Multiline, hybrid, and discrete. For a few countries, such as India, the suggestion list doesn't always return some localities. The data directory doesn't always link localities with regions for these countries. Performance can be slower when the software generates suggestion lists for countries and regions with a large number of localities.

Option	Description
	For a list of supported countries and country codes, see Countries supported by lastline drilldown [page 802].
Combine Overlapping Ranges	Specifies whether the software combines individual suggestions with overlapping ranges. • *YES*: Ignores gaps and overlaps in ranges. • *NO*: Doesn't combine overlapping ranges. Set this option to *YES* to limit the number of total suggestions presented. However, it is possible that you won't see gaps of invalid ranges that would be apparent if this option was set to *NO*. **Example** The software presents the following suggestions when you set this option to *NO*: 1000–1099 Maple Ave 1100–1199 Maple Ave The software presents the following suggestions when you set this option to *YES*: 1000–1199 Maple Ave **Tip** If you set this option to *NO*, increase the size of the *SUGGESTION_LIST* output field so that it can contain all of the suggestions.
Address Range	Specifies a span around the input primary address range for which to return suggestions. Maximum value is 5000. Enter 0 to set no limits to the ranges that the transform returns in suggestion lists. By setting an address range, you limit the suggestions returned to be within the range. **Example** *Address Range* = 500 **Input address:** 1000 Pine St. **Suggestion list range:** 750–1250 Pine Street

Option	Description
Max Number Address Lines	Specifies the maximum number of address line suggestions that the software generates. Minimum setting: 2 Maximum setting: 10,000 Default setting: 100 Set this option, for example, to limit the size of the SOAP documents sent by the web service, or to limit the maximum number of suggestions to choose from.
Max Number Lastlines	Specifies the maximum number of lastline suggestions that the software generates. • Minimum setting: 2 • Maximum setting: 10,000 • Default setting: 60 Set this option, for example, to limit the size of SOAP documents that the Web service sends or to limit the maximum number of suggestions that users choose from. **i Note** You can't set this option when you set the *Enable Lastline Drilldown* is set to *YES*.

3.5.9.11 Global Address Cleanse Report Options

Use the Report Options group to add required certification information to reports for Australia and New Zealand.

Country	Description
Australia	Set report options to complete the required Australia AMAS (Address Matching Processing Summary) Report required by Australia postal authorities.
New Zealand	Set report options to complete the required Statement of Accuracy (SOA) report required by New Zealand postal authorities.

3.5.9.11.1 Global Address Cleanse New Zealand report options

Set options to include the required New Zealand Statement of Accuracy (SOA) information to reports.

Completing this group of options is required only when you are preparing a mailing that needs a New Zealand SOA report for postal certification.

Option	Description
Customer Number	Specifies the New Zealand Post number assigned to the customer for whom you are preparing the list. New Zealand post requires the customer number in the SOA report to qualify for postage discounts.
Customer Company Name	Specifies the customer company name. The customer company is the entity for which you are preparing the list. Maximum is 40 characters.
Mailer Address1 *Mailer Address2* *Mailer Address3* *Mailer Address4* *Mailer Address5* *Mailer Address6*	Specifies the name and address of the person or organization for which you are preparing the list. Maximum is 29 characters per line. Example *Mailer Address1* = ABCD Mailing, Inc. *Mailer Address2* = 123 Main St. *Mailer Address3* = Johnsonville *Mailer Address4* = Wellington 6004 *Mailer Address5* = NEW ZEALAND
SOA Issuer Name	Specifies the company name that prepared the list. Maximum of 40 characters.
File Name	Specifies the input file name. For example, `new_zealand.dbf`.

3.5.9.11.2 Global Address Cleanse Australia report options

Set options to include the required Australia AMAS (Address Matching Processing Summary) information to reports.

Completing this group of options is required only when you are preparing a mailing that needs an Australia AMAS report for postal certification.

Option	Description
Customer Company Name	Specify the customer company name. Maximum is 40 characters. The customer company is the entity for which you are preparing the list.
Mailer Address1 *Mailer Address2* *Mailer Address3* *Mailer Address4*	Specify the name and address of the person or organization for which you are preparing the list. Maximum is 29 characters per line. **Example** *Mailer Address1* = ABCD Mailing, Inc. *Mailer Address2* = 123 Main St. *Mailer Address3* = West Beach SA 1234 *Mailer Address4* = Australia
List Name	Specify the name of your database or mailing list. Maximum is 40 characters. For example, enter the file name, your title, or a formal name for the list.
File Name	Specify the input file name. Maximum is 40 characters. For example, `australia.dbf`.

3.5.9.12 Global Address Cleanse USA engine

To cleanse addresses from the United States and its territories, set USA-specific options in the Global Address Cleanse transform.

Set options for address parsing and suggestion lists for addresses from the United States and territories.

3.5.9.12.1 Global Address Cleanse USA Options

To determine how the transform standardizes addresses from the United States and territories in your data, set USA options.

Option	Description
Parse Only	Specifies whether the software parses addresses and performs a lookup in the postal directory, or only parses addresses. • *YES*: Parses records into discrete components, but does not perform a lookup in the postal directories. *Parse Only* is fast, but parsing results are unverified. • *NO*: Parses records into discrete components and performs a lookup in the postal directories. Setting this option to *NO* may slow down processing, but parsing results are verified.
Unit Description	Specifies how to standardize unit descriptions including apartment, suite, room, or floor. • *CONVERT*: Standardizes unit description based on what it finds in the postal directory (apartment, suite, room, or floor). • *PRESERVE*: Preserves the unit description from the input record, but corrects any spelling errors.
Dual Address	Specifies what address line to validate when the input address is a dual address. *POSITION*: Validates the address that is closest to the lower left corner of the address block. The address may be the postal or street address based on how the data was entered on input. *POSTAL*: Validates the postal address. If the postal address (rural route or PO Box) validation fails, validates the street address. *STREET*: Validates the street address. If the street address validation fails, validates the postal address (rural route or PO Box).
Use Firm To Assign	Specifies whether the transform uses the firm to make an assignment and to display firm in a suggestion list. • *YES*: Assigns address using the firm, and displays the firm in the suggestion list. *YES* is the default setting. • *NO*: Does not assign address using the firm, and does not display the firm name in suggestion lists.

Related Information

Global Address Cleanse Standardization Options [page 484]

3.5.9.12.2 Global Address Cleanse USA Suggestion List Options

Set Suggestion List Options to enable suggestions for addresses from the United States and territories.

Option	Description
Enable Suggestion Lists	Specifies whether the transform generates suggestion lists. *YES*: Generates suggestion lists when assignment candidates are present. *NO*: Does not generate suggestion lists.
Max Number Lastlines	Specifies the maximum number of lastline suggestions that the transform can generate. Enter a value from 2 through 100. The default setting is 15. **Example** Set the *Max Number Lastlines* option to limit the size of SOAP documents sent by the web service. Or, set this option to limit the maximum number of suggestions that users have to choose from.
Max Number Address Lines	Specifies the maximum number of lastline suggestions that the transform can generate. Enter a value from 2 through 100. The default setting is 50. **Example** Set the *Max Number Address Lines* option to limit the size of SOAP documents sent by the web service. Or, set this option to limit the maximum number of suggestions that users have to choose from.
Lastlines Match Minimum	Specifies the similarity score required for last line suggestions. This score then determines which suggestions the transform returns in the list. Type a value from 0 through 80. A higher number indicates that the suggestion must be more similar to the input to be returned as a possible suggestion.

Option	Description
Address Lines Match Minimum	Specifies the similarity score required for address line suggestions. This value then determines which suggestions the transform returns in the list. Type a value from 0 through 80. A higher number indicates that the suggestion must be more similar to the input to be returned as a possible suggestion.
Combine Overlapping Ranges	Specifies whether the transform combines individual suggestions with overlapping ranges. • *YES*: Ignores gaps and overlaps in ranges. • *NO*: Does not combine overlapping ranges. You might set this option to *YES* 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*. **Example** A suggestion list shows the following suggestions when you set this option to *NO*: 1000-1099 Maple Ave 1100-1199 Maple Ave But the suggestion list shows the following suggestion if you set this option to *YES*: 1000-1199 Maple Ave

Option	Description
Address Range	Specifies a span around the input primary address range for which to return suggestions. By using this option, you can limit the suggestions returned to be within a few blocks of your input. If you don't want to limit the ranges returned in suggestions, enter 0 for a value. Example Example You enter 500 for *Address Range*. Then, you submit the following street address: 1000 Pine St. The software returns only suggestions in a range from 750 to 1250 Pine Street.

3.5.9.13 Global Address Cleanse Suggestion List Format options

Select the format options for the suggestion list string that the transform outputs to the *Suggestion_List* output field.

The Global Address Cleanse transform uses settings from each suggestion list group and the suggestion list components group for the content of the *Suggestion_List* output field.

Option	Description
Output Style	Specifies the format for the output suggestion list string. • *DELIMITED*: Outputs the suggestion list string in a delimited text format. Uses the delimiters that you specify in the *Delimiter* and *Field Delimiter* options. • *XML*: Outputs the suggestion list data as hierarchical XML. Use *XML* when you integrate suggestion lists via the web service. Use your own XML tools to parse the suggestion list data. *XML* is the default setting.

Option	Description
Delimiter	Specifies a character to use to separate each suggestion in a suggestion list string. The value can be any character or string. Common delimiters include a pipe symbol (|), or a string of multiple asterisks (***). Select a value that is different from the *Field Delimiter* value. The transform uses this value only when you set the *Output Style* option to *Delimited*.
Field Delimiter	Specifies a character to use to separate each field in a single suggestion. Each suggestion can be made up of one or more fields. If you choose to retrieve multiple fields per suggestion, the software separates the fields with the character specified here. The value can be any character or string. The default value is a pipe symbol (|). Select a delimiter that is different from the *Delimiter* value. The transform considers this value only when you set the *Output Style* option to *Delimited*.

Related Information

Global Address Cleanse Suggestion List option

3.5.9.14 Global Address Cleanse Suggestion List Components

Select the suggestion list components to include in the *SUGGESTION_LIST* output field.

The components you select here apply to suggestion lists for Global, Canada, and USA engines. For each component, select *YES* or *NO*:

- Select *YES* to include the component in suggestion lists.
- Select *NO* to exclude the component from suggestion lists.

Component	Description
Selection	Returns a unique index number that distinguishes this suggestion from the other suggestions in the returned list. The suggestion selection number ranges from 1 to the number of suggestion selections in the suggestion list.
Locality1 *Locality2* *Locality3* *Locality4*	Returns the city, town, or suburb. Additional locality information goes in *Locality4*.
Locality1_Official *Locality2_Official* *Locality3_Official* *Locality4_Official*	Returns the locality name that the applicable postal authority prefers.
Postcode	Returns the postal code. For USA: Returns the 5-digit ZIP Code and ZIP+4.
Postcode1	Returns information specific to the country: • **Australia:** Returns 4-digit postcode • **Canada:** Returns the first three characters (FSA) of the postal code. • **Global:** Returns the postal code. • **USA:** Returns the 5-digit primary postal code (ZIP Code). Does not include the 4-digit secondary postal code (ZIP4).
Postcode2	Returns the secondary postal code. • **Canada:** Returns the last three characters (LDU) of the postal code. • **USA:** Returns the 4-digit ZIP Code (ZIP4). The ZIP4 follows the primary postal code with a hyphen placed between, such as 54601-1234.
Region1 *Region2*	Returns the state, province, or region.
ISO Country Code 2-Char	Returns the two-character ISO country code.

Component	Description
Primary Number Low *Primary Number High*	Returns the low and high primary number. **Example** If the house number is a range, such as 100-102: • *Primary Number Low*: 100 • *Primary Number High*: 102 If the house number is not a range, both fields contain the house number: • *Primary Number Low*: 100 • *Primary Number High*: 100
Primary Number Description	Returns a description that precedes the primary number. For example, LOT (Australia) 99, Building 10, or blk. 101.
Primary Number Extra	Returns data that is near the parsed primary number in the address that the transform cannot identify or that does not belong in a standardized address.
Primary Number Full	Returns the primary number, primary number description, and primary number extra, when they exist.
Primary Prefix1-2	Returns the abbreviated or full directional that precedes a street name. For example, N, South, NW, SE. The transform uses your setting for *Directional Style* to determine whether to return abbreviated or non-abbreviated directional.
Primary Name1	Returns the street name description, such as a street name or box description.
Primary Name2	Returns the second street name and description, such as a street name or box description.
Primary Name3 *Primary Name4*	Returns the street name, delivery mode, and so on.
Primary Type 1-4	Returns the primary name type. For example, rue, strasse, street, Ave, or Pl.
Primary Postfix1-2	Returns an abbreviated or non-abbreviated directional that follows a street name. For example, N, South, NW, SE. Abbreviated or non-abbreviated is based on the standardization setting for *Directional Style*.

Component	Description
Primary Name Full1 *Primary Name Full2*	Returns the primary name, primary type, primary prefix, and primary postfix.
Primary Name Full3 *Primary Name Full4*	Returns the primary name and primary type.
Delivery Installation Name	Returns the delivery installation city name, which is the same as the city name in most cases. If it is the same as the city name, the transform omits it from the address line.
Delivery Installation Qualifier	Returns the delivery installation qualifier. **Example** **Input:** RR2 Vancouver Stn Main **Output:** Main
Delivery Installation Type	Returns the delivery installation type. **USA:** • **PO**: Post Office • **RPO**: Retail Post Outlet • **STN**: Station • **LCD**: Letter Carrier Depot • **CMC**: Community Mail Center • **CDO**: Commercial Dealership Outlet **France:** • **BDP**: Bureau de Poste • **CSP**: Comptoir Service Postal • **SUCC**: Succursale • **PDF**: Poste de Facteurs • **CPC**: Centre Postal Communautaire • **CC**: Concession Commerciale
Primary Side Indicator	Returns the valid primary side indicator. The value applies to Street and PO box. **E**: The record is even-numbered. **O**: The record is odd-numbered. **B**: The record covers both the even- and odd-numbered sides of the street and PO Boxes.
Firm	Returns the firm, company, or organization name.

Component	Description
Building Name	Returns the building name for the address, which in some countries is used in place of the primary number. **Example** Input address from the U.K.: White House, High Street With *Building Name* set to *YES*, the output is "White House." When set to *NO*, the transform outputs the primary number, "100 High Street."
Unit Description	Returns the unit description within a building. For example, Room, Unit, Apt, and Suite.
Unit Number Low *Unit Number High*	If the unit number is a range, returns the low and high unit number. **Example** With a unit number range of 1-20: • *Unit Number Low* = 1 • *Unit Number High* = 20
Stairwell Description	Returns the entrance or stairwell identifier for a building. For example, Stiege.
Stairwell Name	Returns the name or number of an entrance or stairwell for a building. For example, Stiege "1."
Floor Number Low *Floor Number High*	If the floor number is a range, returns the low and high number. **Example** with the range 20–22: • *Floor Number Low* = 20 • *Floor Number High* = 22 If the floor number is not a range, both fields contain the floor number. • *Floor Number Low* = 20 • *Floor Number High* = 20
Floor Description	Returns the level description, such as Floor.

Component	Description
Secondary Side Indicator	Returns the secondary record side indicator. This value applies to floors and units: • **E**: The secondary record is even-numbered. • **O**: The secondary record is odd-numbered. • **B**: The secondary record covers both the even and the odd-numbered values.
Sugg Range Type	Indicates the range type for each suggestion: • **PRIM** = Suggestion represents a range of house numbers. • **UNIT** = Suggestion represents a range of unit numbers. • **FLOOR** = Suggestion represents a range of floor numbers. • *blank* = Suggestion does not include a range.
Sugg Full Addressline	Returns the complete address line, including firm name, secondary address, and dual address (street and postal) as appropriate for the country. If the input address line uses a street name alias, includes the input street name alias in parentheses. **Example** Given the input 250 Bliss Rd: *Sugg Full Addressline* = 250 MAIN ST (BLISS RD) If the input address contains a firm name, includes the firm name in parentheses after the address line. **Example** Given the input: ABC, Inc Main St *Sugg Full Addressline* = 100 MAIN ST (ABC,INC)
Sugg Full Lastline	Returns the locality, region, and postal code together in one component as appropriate for the country. Additionally, if you select *YES* for *Enable Suggestion Lists* in the *Country ID Options* group, outputs the two-character ISO country code that you select from the list of candidate countries.

Component	Description
Sugg Single Address	Returns the combined result of the full address line and full last line in the order appropriate for the country. Additionally, if you select *YES* for *Enable Suggestion Lists* in the *Country ID Options* group, outputs the two-character ISO country code that you select from the list of candidate countries.
Sugg Confidence Score	Returns the confidence score for the suggestion entry. Values are from 0 through 100.

3.5.9.15 Output field category columns for Global Address Cleanse and USA Regulator Address Cleanse transforms

Choose the fields that SAP Data Services includes in the output by selecting a field category in the *Output* tab.

The *Output* tab lists output fields that hold the data that the transform cleanses or creates. 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 in the *Schema Out* pane.

All : Lists all the output fields that are available for this transform.

> **i Note**
>
> For details about mapping input and output fields, see the *Designer Guide*.

The following table describes the output field attributes that appear as column headings in the *Output* tab for each output field. The field category displays “None” when it does not apply to the field.

Field category column	Description
Field Name	Specifies the output field name.

Field category column	Description
Field Class (USA Regulatory Address Cleanse)	Specifies the type of information the transform includes in the output. Map the *Field Name* that corresponds to the desired *Field Class*. • *Best:* Outputs data based on factors such as whether an address was assigned and the settings that you define in the *Standardization Options* group in the *Options* tab. > **i Note** > When you enable NCOALink and a valid move is available, Best fields contain the move-updated address data if it exists and if it has a match in the national directories. If the move does not exist, or it does not have a match in the national directories, contains the original address data. • *Correct*: Outputs the complete and correct value from the directories, and standardizes the value according to settings that you define in the *Standardization Options* group in the *Options* tab. • *Parsed*: Outputs the parsed value. • *Pre_LACSLink*: Outputs retained address components that the transform replaced with LACSLink address information. • *Move_Updated*: Outputs the address components that the transform updated with move-updated address data. > **i Note** > The transform looks for the move-update address information in the national directories. When it does not find the move-update address information in the national directories, the transform populates the Move Updated fields with information from the Move Update directories only. The transform does not update the Move Updated fields that were populated as a result of standardizing against the national directories.
Field Class (Global Address Cleanse)	Specifies the type of information the transform includes in the output. Map the *Field Name* that corresponds to the desired *Field Class*. • *Best:* Outputs data based on factors such as whether an address was assigned and the settings that you define in the *Standardization Options* group in the *Options* tab. • *Parsed*: Outputs the parsed value.

Field category column	Description
Field Addrclass	Specifies the address class for the generated field. • *Delivery*: Generates fields that reflect the address that the software uses when it attempts to assign an address. Use with the applicable *Field Name*. • *Dual*: Generates fields that reflect the address that the software does not use when it attempts to assign an address for input records that may contain both a street and postal address on input. Use with the applicable *Field Name*. *Official:* Generates fields in the form of the data preferred by the Postal Authority. Use with the applicable *Field Name*. **Example** In Winona Minnesota USA, Broadway and 6th Street are alternate names for the same street. The USPS delivers a letter using Broadway in the address, but it prefers to use 6th street.
Field Category	• *Component*: Individual address components and postal codes that are related to the processed record. • *Standardized*: Standardized input lines based on the settings in the *Standardization Options* group in the transform. • *Suggestion*: Suggestion list output data based on the settings in the *Suggestion List Options* group.
Type	Specifies the type and default length of data the output field contains. For example, specifies varchar, date, and time.
Content Type	Identifies the type of data in the field. Setting the content type helps you map fields when you set downstream transforms.

Related Information

Data Quality field content types [page 348]
How address cleanse works

3.5.9.16 Global Address Cleanse input fields

Map fields from your input data to Global Address Cleanse input fields so the transform knows the type of data to expect on input.

> **Note**
>
> Reference to “country” or “countries” in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

The following table contains an alphabetical list of the Global Address Cleanse input fields. The table also shows the engines applicable for each field. Engines include the following:

- Canada (C)
- Global Address (G)
- USA (U)

To view properties for each field, open the Input tab in the transform editor.

Input field	Description	Engine
ADDRESS_LINE	Contains the delivery address line, for example, "123 Main Street, Unit 4." **China:** *ADDRESS_LINE* contains the following address components: • Street and street number • Building, floor, and unit • Residential community Example 晨晖路 123 号中华大厦 12 楼 1201 室 宝山新村 100 号 201 室 **Japan:** *ADDRESS_LINE* contains the following address components: • Block (chome, kumi, Hokkaido go), sub-block (banchi, gaiku, tochi kukaku), and house number (go). • The building name, building floor, and building room. • The P.O. Box portion of the address, if applicable. *South Korea*: *ADDRESS_LINE* contains one of the following address components: • Land-oriented lot number address, neighborhood or village, lot number, and additional address information. For example, 효자동 200 • Road name address, street, house number, and additional address information. For example, 월미로 377-1	All engines

Input field	Description	Engine
COUNTRY	Contains the identified country name of the address.	All engines
DATA_SOURCE_ID	Specifies the name of the data source that the record comes from. → Tip If you choose to generate data quality statistics tables, use this field as part of your mapping strategy, along with primary key fields and the output field *ROW_ID*. This mapping identifies the source of individual records in any of the non summary data quality statistics tables, thus enabling you to trace problem data to the data source. This input field is applicable for the following transforms: • Data Cleanse • Global Address Cleanse • Geocoder	All engines
FIRM	Contains 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, the Global Address Cleanse transform can assign more specific postal codes. **China:** China doesn't support Firm assignment. There's no Firm data for China. However, if firm data is available on input, the transform places it in the *FIRM* field. **Japan:** Map all Firm data for addresses in Japan to the *FIRM* field.	All engines
LASTLINE	Contains the locality, region (when it's included in the data), and postal code on one line.	All engines

Input field	Description	Engine
LOCALITY1	Contains the city, town, or suburb. **China:** Contains the following Prefecture-level localities when present on input: • Prefectures (地区 diqu) • Autonomous prefectures (自治州 zizhizhou) • Prefecture-level cities (地级市 dijishi) • Leagues (盟 meng) • Provincial countries (省直辖县 shengzhixiaxian) **Japan:** Contains the following information when present on input: • City (shi) • Island (shima) • Ward (ku) • County (gun) • District (machi) • Village (mura)	All engines

Input field	Description	Engine
LOCALITY2	Contains any additional city, town, or suburb information. **China:** Contains county level localities when present on input: • 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) • Special districts (特区 tequ) **Japan:** Contains any additional ward, district, village, or sub-district (aza, bu, chiwari, sen) when present on input. ! Restriction **South Korea:** For land-oriented lot number address input, don't include neighborhood or village in *LOCALITY2*. Instead, map neighborhood, village, and lot number, to *ADDRESS_LINE* or *MULTILINE1-12*. **USA:** Contains the Puerto Rico urbanization when present on input.	G, U

Input field	Description	Engine
LOCALITY3	Contains any additional city, town, or suburb information. **China:** Contains the following information when present on input: • 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) • Ethnic sumu (民族苏木 minzusumu). **Japan:** Contains any additional district, village, sub-district (aza, bu, chiwari, sen, donchi, and tori), or super block (joh). ! Restriction **South Korea:** For land-oriented lot number address input, don't include neighborhood or village in *LOCALITY2*. Instead, map neighborhood, village, and lot number, to *ADDRESS_LINE* or *MULTILINE1-12*.	G

Input field	Description	Engine
LOCALITY4	Contains any additional city, town, or suburb information. **China:** Contains village and neighborhood level localities when present on input: • Administrative villages (行政村) • Neighborhood committees (社区居民委员会) • Neighborhoods or communities (社区) • Village committees (村民委员会). **Japan:** Contains any additional district, village, sub-district (aza, bu, chiwari, sen, and tori), or super block (joh).	G
MULTILINE1-12	Contains any data. The type of data in these fields can vary from record to record. **Japan:** Contains any data with the following restrictions: • If the input is address data, the address in total has to be in the traditional order of a Japanese address. • If the input is block data, place the block (chome, kumi, Hokkaido go), subblock (banchi, gaiku, tochi kukaku), and house number (go) in one line on input.	All engines
NW_FIELDS	Contains fields for SAP application data. Use with other NW input fields for cleansing data. Use NW fields as a group only, not individually or with non-NW input fields. For descriptions of NW input fields, see Global Address Cleanse NW input fields [page 596]	
POSTCODE	Contains the postal code. **USA:** Contains the five-digit ZIP Code and the ZIP+4.	All engines

Input field	Description	Engine
REGION1	Contains the state, province, or region. **China:** Contains the province-level regions when present on input: • Provinces (省 sheng) • Autonomous regions (自治区 zizhiqu) • Municipalities (直辖市 zhixiashi) • Special administrative regions (特别行政区 tebie xingzhengqu) **Japan:** Contains the prefecture (to, do, fu, ken). A prefecture is similar to a state in the U.S.	All engines
REGION2	Contains the state, province, or region.	All engines

Input field	Description	Engine
SUGGESTION_REPLY1-6	Contains the index number that corresponds to a specific last line suggestion, an address line suggestion, or secondary list suggestion. Can also contain a street primary range or a street secondary range. **i Note** To use these fields, enable suggestion lists in the transform configuration. To use one suggestion reply input field to hold all of the suggestion replies rather than using all six reply fields, use the *SUGGESTION_REPLY1* field and separate the replies with a pipe (|). When using the *SUGGESTION_REPLY1-6* fields for SAP applications for street and PO Box addresses, insert the following symbols to indicate whether the user has accepted changes made to the street address and when they're done with the street address: • asterisk plus (*+): User accepts the changes made to the street address up to the specified point and is done with the street address. • asterisk minus (*-): User doesn't accept the changes made to the street address up to the specified point and is done with the street address.	All engines

3.5.9.17 Global Address Cleanse suggestion list input fields

The suggestion list feature in the Global Address Cleanse transform supports all Global Address Cleanse input fields in addition to the suggestion reply fields.

Field name	Description
Suggestion_Reply1-6	Contains the index number that corresponds to a specific last line suggestion, an address line suggestion, or secondary list suggestion. May also contain a street primary range or a street secondary range. **i Note** To use these fields, enable suggestion lists in the transform configuration. To use one suggestion reply input field to hold all of the suggestion replies rather than using all six reply fields, use the *Suggestion_Reply1* field and separate the replies with a pipe (|). When using the *Suggestion_Reply1-6* fields for SAP applications for street and PO Box addresses, insert the following symbols to indicate whether the user has accepted changes made to the street address and when they are done with the street address: • *asterisk plus (*+)*: User accepts the changes made to the street address up to the specified point and is done with the street address. • *asterisk minus (*-)*: User does not accept the changes made to the street address up to the specified point and is done with the street address. **For country suggestion lists** The *Suggestion_Reply1-6* input field contains the two-character ISO country code prefixed with "CID:" under the following circumstances: • You enable the *Enable Suggestion Lists* option in the *Country ID Options* group. • You enable the *Country* suggestion list component. The transform sends "CID:XX", where "XX" is the two-character ISO country code, to the *Suggestion_Reply1-6* input field. It then reassigns the address using the country code in this field.

3.5.9.18 Global Address Cleanse output fields

The Global Address Cleanse transform provides output fields for your processed data and output fields for additional generated data.

> **i Note**
>
> Reference to “country” or “countries” in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

The following table contains an alphabetical list of the Global Address Cleanse output fields.

The table also indicates the applicable engines for each field. The engines include the following:

- Canada (C)
- Global Address (G)
- USA (U)

To see the *Field_AddrClass* and *Field_Class* attributes for each output field, open the *Output* tab of the Global Address Cleanse transform editor.

Output field name	Description	Engine

ADDITIONAL_INFO1	Contains additional information specific to the country.
	• **Austria:** The PAC code of the current valid address when:
	  • You preserve the alias address on output
	  • The PAC codes for the master record and the alias record are different.
	• **Belgium:** The NIS code.
	• **Canada:** The official 13-character abbreviation of the city name, or the full spelling when the city name is less than 13 characters including spaces.
	• **France:** The INSEE code.
	• **Germany:** A portion of the German freightcode (Frachtleitcode).
	• **Liechtenstein:** The postal service district (Botenbezirke) when it is present in the data.
	• **Poland:**The community name (gmina).
	• **Russia:** A tax code (IFNSUL) and territory code (TERRIFNSUL) for the level that the address validates. The two codes are separated with a pipe symbol (|) as in IFNSUL|TERRIFNSUL.
	• **South Korea:** The 25-digit administration number.
	• **Spain:** The INE 91 section code.
	• **Switzerland:** The postal service district (Botenbezirke) when it is available in the data.
		C, G

Output field name	Description	Engine
ADDITIONAL_INFO2	Contains additional information specific to the country: • **Austria:** The City ID (OKZ). • **Canada:** The official 18-character abbreviation of the city name, or the full spelling if the city name is less than 18 characters including spaces. • **Germany:** The District Code. • **Liechtenstein:** Additional post-code. • **Russia:** An administrative territorial division code (OKATO) and municipality code (OKTMO) for the level that the address validates. The two codes are separated with a pipe symbol (|) as in OKATO|OKTMO. • **Spain:** The INE Street code. • **Switzerland:** Additional postcode.	C, G
ADDITIONAL_INFO3	Contains additional information specific to the country: • **Austria:** The Pusher-Leitcode (parcel). • **Germany:** The German City ID (ALORT). • **Spain:** The INE Town code.	G
ADDITIONAL_INFO4	Contains additional information specific to the country: • **Austria:** The Pusher-Leitcode (letter). • **Germany:** The German street name ID (StrSchl).	G
ADDITIONAL_INFO5	Contains additional information specific to the country: • **Austria:** The 7-digit SKZ Street Code. • **Germany:** The discount code for the freightcode.	G

Output field name	Description	Engine
ADDITIONAL_INFO6	Contains additional information specific to the country: • **Austria:** The corner-house identification (1-digit). The value for a corner house is 1. • **Russia:** An identifier (OBJECTID) and global unique identifier (OBJECTGUID) for the level that the address validates. The two codes are separated with a pipe symbol (|) as in OBJECTID|OBJECTGUID.	G
ADDITIONAL_INFO7	**Russia:** A postcode for the level that the address validates.	All engines
ADDITIONAL_INFO8	**Russia:** A code of hierarchy. • 0=Both • 1=Administrative-Territorial • 2=Municipal	All engines
ADDRESS_LINE_REMAINDER1-4	Contains extraneous data in the address line that the transform cannot identify or that does not belong in a standardized address. **USA,** *ADDRESS_LINE_REMAINDER1-2***:** Complete secondary non-postal address such as Apt. 10, Ste 500, Box 34, Rm 7, 5th Flr.	All engines
ADDRESS_TYPE	Contains a one-character code that represents the type of address identified: • *P*: Postal • *S*: Street • *X*: Unknown	All engines
AREA_NAME1	Contains an industrial area such as RIICO INDUSTRIAL AREA.	G
ASSIGNMENT_INFO	Contains a code that indicates whether a record is valid, invalid, or corrected, based on the status and information codes. • *C*: Corrected • *I*: Invalid • *V*: Valid • *B*: Blank	All engines

Output field name	Description	Engine
ASSIGNMENT_LEVEL	Contains the level to which this transform matched the address to the data in the reference files (directories): • *C*: Country • *L1*: Locality1 • *L2*: Locality2 • *L3*: Locality3 • *L4*: Locality4 • *PN*: Primary name • *PR*: Primary range • *R*: Region • *S*: Secondary • *X*: Unknown, or the address was unassigned	All engines
ASSIGNMENT_TYPE	Contains a one- or two-character code that represents the type of address. • *BN*: Building name (Canada, Global Address) • *F*: Firm (Canada, Global Address, USA) • *G*: General delivery (Canada, Global Address, USA) • *H*: High-rise building (Canada, USA) • *HB*: House Boat (Global Address) • *L*: LOT (Global Address) • *M*: Military (Canada, USA) • *R*: Rural (Canada, USA) • *P*: Postal (Canada, Global Address, USA) • *PI*: Point of reference (Global Address) • *PS*: Packstation or Paketbox (Global Address) • *RP*: Postal Served by Route (Global Address) • *S*: Street (Canada, Global Address, USA) • *SR*: Street served by route (Canada, Global Address) • *U*: Uninhabited (Global Address) • *W*: Caravan (Global Address) • *X*: Unknown type or unassigned address (Canada, Global Address, USA)	Various engines
BLOCK_DESCRIPTION	Contains the block description.	G

Output field name	Description	Engine
BLOCK_FULL	Contains combined data from the following output fields: • *BLOCK_DESCRIPTION* • *BLOCK_NUMBER*	G
BLOCK_NUMBER	Contains the block number.	G
BUILDING1	Contains the base name for a building. The transform separates the base name from the description for countries that separate the two in reference data. **Example** A building name in the Russian Federation is “корпус Б.” The transform outputs building data as follows: • *BUILDING1* contains the base name “Б” • *BUILDING_DESCRIPTION1* contains the descriptor “корпус” For countries that consider the descriptor as part of the building name, the transform outputs both elements to the *BUILDING1* field. **Example** A building name in the United Kingdom is “Dover House.” The transform outputs the building data as follows: • *BUILDING1* contains “Dover House” • *BUILDING_DESCRIPTION1* is blank	G

Output field name	Description	Engine
BUILDING2	Contains the base name for the second building of an address that contains two buildings. The transform separates the base name from the description for countries that separate the two in reference data. For countries that consider the descriptor as part of the building name, the transform outputs both elements to the *BUILDING2* field. For examples, see the description for *BUILDING1*.	G
BUILDING_DESCRIPTION1	Contains the building descriptor. For countries that separate the base building name from the descriptor in reference data, the transform outputs the building description to this field. **Example** The transform outputs the building name “корпус Б” from the Russian Federation as follows: • *BUILDING_DESCRIPTION1* contains the descriptor “корпус” • *BUILDING1* contains the base name “Б” For countries that consider the descriptor as part of the building name, the transform outputs both elements to the *BUILDING1* field and leaves the *BUILDING_DESCRIPTION1* field blank. **Example** The transform outputs the building name “Dover House” from the United Kingdom as follows: • *BUILDING_DESCRIPTION1* is blank • *BUILDING1* contains “Dover House”	

Output field name	Description	Engine
BUILDING_DESCRIPTION2	Contains the building descriptor for the second building in an address that contains two buildings. For countries that separate the base building name from the descriptor in reference data, the transform outputs the building description to this field. For countries that consider the descriptor as part of the building name, the transform outputs both elements to the *BUILDING2* field and leaves the *BUILDING_DESCRIPTION2* field blank. For examples, see the description for *BUILDING_DESCRIPTION1*.	
BUILDING_NAME1	Contains the full building name, which includes the building name and the building descriptor. **Example** "White House" or "Empire Tower"	G
BUILDING_NAME2	Contains the full building name for the second building when an address contains two building names. The transform outputs the second building name to *BUILDING_NAME2*. **Example** "White House" or "Empire Tower"	G
BUILDING_NAME1_2	Contains combined data from the following output fields: • *BUILDING_NAME1* • *BUILDING_NAME2*	G

Output field name	Description	Engine
BUILDING_PRIMARY_ADDR_DELIVERY_DUAL	Contains combined data from the following output fields: • *BUILDING_NAME1* • *BUILDING_NAME2* • *BUILDING_NAME1_2* • *PRIMARY_ADDRESS_DELIVERY* • *PRIMARY_ADDRESS_DELIVERY_DUAL* **i Note** *PRIMARY_ADDRESS_DELIVERY_DUAL* consists of the following combined output fields: • *PRIMARY_ADDRESS1-4 (delivery)* • *PRIMARY_ADDRESS1-4 (dual)*	All engines
BUILDING_PRIMARY_SECONDARY_ADDR_DELIVERY_DUAL	Contains combined data from the following output fields: • *BUILDING_NAME1* • *BUILDING_NAME2* • *PRIMARY_SECONDARY_ADDRESS (delivery)* • *PRIMARY_SECONDARY_ADDRESS (dual)*	All engines
CERT_VALID	Contains a valid certification indicator.	All engines
COUNTRY	Contains 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	Contains the fully spelled country name in the languages specified in the *OUTPUT_COUNTRY_LANGUAGE* option.	All engines
COUNTY_NAME	Contains the fully spelled county name. *USA*: Addresses do not include county information.	U
DELIVERY_INSTALLATION_FULL	Contains combined data from the following output fields: • *DELIVERY_INSTALLATION_NAME* • *DELIVERY_INSTALLATION_QUALIFIER* • *DELIVERY_INSTALLATION_TYPE*	C, G

Output field name	Description	Engine
DELIVERY_INSTALLATION_NAME	Contains the delivery installation city name, which is sometimes the same as the city name. If the delivery installation city name is the same as the city name, the transform omits the delivery installation city name from the address line. **Canada:** If the delivery installation name is different than the locality name, the transform outputs the delivery installation name to the secondary address fields. **Japan:** Returns the post office name.	C, G
DELIVERY_INSTALLATION_QUALIFIER	Contains the delivery Installation qualifier. For example, "Main" in "RR 2 Vancouver Stn Main".	C
DELIVERY_INSTALLATION_TYPE	Contains the delivery installation type. **USA:** • *PO*: Post Office • *RPO*: Retail Post Outlet • *STN*: Station • *LCD*: Letter Carrier Depot • *CMC*: Community Mail Center • *CDO*: Commercial Dealership Outlet **France:** • *BDP*: Bureau de Poste • *CSP*: Comptoir Service Postal • *SUCC*: Succursale. • *PDF*: Poste de Facteurs • *CPC*: Centre Postal Communautaire • *CC*: Concession Commerciale	C

Output field name	Description	Engine
DELIVERY_POINT	Contains the delivery indicator specific for the country of the address. • **Australia:** Eight-digit delivery point identifier, which is the primary component for generating a bar-code. The delivery point identifier is not printed on the mail pieces. • **Austria:** The PAC code, which is a unique identifier assigned by the Austrian postal authority. • **New Zealand:** A seven-character code that represents the delivery-point identifier. • **United Kingdom:** A two-character code that represents the delivery-point suffix.	G
EXTRA1-12	Contains any non-address data in the address block. Available only when the input data is presented through multi-line fields.	All engines
FIRM	Contains the firm name for the address. When you use a multiline format, the transform identifies the firm name based on the level of firm data available in the postal directories for each engine. Therefore results may be inconsistent. To avoid inconsistent identification of firm data, use the discrete *FIRM* field when you process multiline data. **Canada and USA:** The transform re-trieves the firm name from the postal directory when the directory has the firm name; otherwise, the transform re-trieves the firm name from the input re-cord. Be aware that the postal directory might contain some unusual or short-ened spellings that you may not find suitable for printing on mail pieces. If you prefer to retain your own firm data, retrieve the parsed component. **Global Address:** If the firm name is available on input or from reference data, the Global Address engine returns the firm name.	All engines

Output field name	Description	Engine
FLOOR_DESCRIPTION	Contains a building level description such as "Floor." **Japan:** The level description, such as kai.	All engines
FLOOR_FULL	Contains combined data from the following output fields: • *FLOOR_DESCRIPTION* • *FLOOR_NUMBER* • *FLOOR_QUALIFIER*	All engines
FLOOR_NUMBER	Contains the building level number or floor information.	All engines
FLOOR_QUALIFIER	Contains any additional word that precedes or follows the floor information.	All engines
ERROR	Contains a value that represents the error status generated when the transform looks up a record and performs suggestion processing. Valid values are 0 through 6: • *0*: No suggestion selection error. • *1*: Blank suggestion selection/entry. • *2*: Invalid suggestion selection. • *3*: Invalid primary range. • *4*: Invalid floor range. • *5*: Invalid unit range. • *6*: Too many possible matches to generate a suggestion list. Provide more information, such as a postal code, region, or locality.	All engines
FULL_ADDRESS	Contains the complete address line, including secondary address and dual address (street and postal).	All engines
INFO_CODE	Contains a four-character code for addresses that do not fully assign that describes why the transform could not assign the address. If the address is fully assigned, the field is blank. For more information, see "Global Address Cleanse information codes".	All engines

Output field name	Description	Engine
ISO_COUNTRY_CODE_2CHAR	Contains the two-character ISO code that identifies a country, for example, DE is for Germany.	All engines
ISO_COUNTRY_CODE_3CHAR	Contains the three-character ISO-3166 code that identifies a country. For example, DEU is for Germany.	All engines
ISO_COUNTRY_CODE_3DIGIT	Contains the three-digit ISO code that identifies a country. For example, 276 is for Germany.	All engines
ISO_SCRIPT_CODE	Contains the four-character script code for an identified country. For example, LATN or KANA.	All engines
LANGUAGE	Contains the two-character ISO language code that represents the language of the address.	All engines
LASTLINE	Contains combined data from the following output fields: • *LOCALITY* (1–4 if available) • *REGION* • *POSTCODE* The transform includes the region only when the country requires it.	All engines
LASTLINE_REMAINDER1-4	Contains unused lastline remainder data.	G
LOCALITY_CODE	Contains a locality code, which some countries use to distinguish sections of a large locality. For example, in France, locality codes are called “arrondissements”.	G
LOCALITY1_2_FULL	Contains combined data from the following output fields: • *LOCALITY1_1_FULL* • *LOCALITY1_2_FULL*	All engines
LOCALITY1_2_NAME	Contains combined data from the following output fields: • *LOCALITY1_NAME* • *LOCALITY2_NAME*	All engines

Output field name	Description	Engine
LOCALITY1_4_FULL	Contains combined data from the following output fields: • *LOCALITY1_FULL* • *LOCALITY2_FULL* • *LOCALITY3_FULL* • *LOCALITY4_FULL*	All engines
LOCALITY1_4_NAME	Contains combined data from the following output fields: • *LOCALITY1_NAME* • *LOCALITY2_NAME* • *LOCALITY3_NAME* • *LOCALITY4_NAME*	All engines
LOCALITY1_ADDITION	Contains additional locality information.	G
LOCALITY1_ALTERNATE	Contains the preserved input locality when the postal authority recognizes it as a legitimate locality. The transform corrects any spelling errors.	C, U
LOCALITY1_DESCRIPTION	Contains the *LOCALITY1* descriptor. • **Japan:** For example, shi, shima. • **China:** For example, 市(Shi).	G
LOCALITY1_FULL	Contains combined data from the following output fields: • *LOCALITY1_NAME* • *LOCALITY_CODE* • *LOCALITY1_DESCRIPTION* • *LOCALITY1_QUALIFIER* The field may include *LOCALITY1_ADDITION*, depending on the standardization option settings of *LOCALITY_NAME_STYLE* and *INCLUDE_LOCALITY_ADDITION*.	All engines

Output field name	Description	Engine
LOCALITY1_NAME	Contains the city, town, locality, or suburb that is one of the following output fields based on the standardization option setting for *ASSIGN_LOCALITY*: • *LOCALITY1_ALTERNATE* • *LOCALITY1_OFFICIAL* **Japan:** Contains the city (shi), island (shima), ward (ku), county (gun), district (machi), or village (mura).	All engines
LOCALITY1_OFFICIAL	Contains the locality name preferred by the postal authority.	All engines
LOCALITY1_QUALIFIER	Contains the locality qualifier. Applicable only for France for Cedex.	G
LOCALITY2_4_FULL	Contains combined data from the following output fields: • *LOCALITY2_FULL* • *LOCALITY3_FULL* • *LOCALITY4_FULL*	All engines
LOCALITY2_4_NAME	Contains combined data from the following output fields: • *LOCALITY2_NAME* • *LOCALITY3_NAME* • *LOCALITY4_NAME*	All engines
LOCALITY2_CODE	Contains the LOCALITY2 code when an address has LOCALITY1 and LOCALITY2 information. **i Note** Currently applicable to Czechia addresses only.	
LOCALITY2_DESCRIPTION	Contains the description of the subdivision of *LOCALITY1* when it exists.	G
LOCALITY2_FULL	Contains the combined data from the following output fields: • *LOCALITY2_NAME* • *LOCALITY2_DESCRIPTION*	G, U

Output field name	Description	Engine
LOCALITY2_NAME	Contains additional locality information. **USA:** Contains urbanization (Puerto Rico addresses only).	G, U
LOCALITY2_OFFICIAL	Contains the locality name preferred by the postal authority.	G
LOCALITY2_QUALIFIER	Contains the locality qualifier.	G
LOCALITY3_4_FULL	Contains combined data from the following output fields: • *LOCALITY3_FULL* • *LOCALITY4_FULL*	All engines
LOCALITY3_4_NAME	Contains combined data from the following output fields: • *LOCALITY3_NAME* • *LOCALITY4_NAME*	All engines
LOCALITY3_DESCRIPTION	Contains the description of a subdivision of *LOCALITY2* when it exists.	G
LOCALITY3_FULL	Contains combined data from the following output fields: • *LOCALITY3_NAME* • *LOCALITY3_DESCRIPTION*	G
LOCALITY3_NAME	Contains additional locality information.	G
LOCALITY3_OFFICIAL	Contains the locality name preferred by the postal authority.	G
LOCALITY3_QUALIFIER	Contains the locality qualifier.	G
LOCALITY4_DESCRIPTION	Contains the description of a subdivision of Locality3 when it exists.	G
LOCALITY4_FULL	Contains combined data from the following output fields: • *LOCALITY4_NAME* • *LOCALITY4_DESCRIPTION*	G
LOCALITY4_NAME	Contains additional locality information.	G
LOCALITY4_OFFICIAL	Contains the locality name preferred by the postal authority.	G
LOCALITY4_QUALIFIER	Contains the locality qualifier.	G

Output field name	Description	Engine
MATCH_BLOCK_NUMBER	Contains a form of *BLOCK_NUMBER* that the Match transform can use during the comparison process. The transform does not output block descriptions. The transform outputs data in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the transform uses normal width standardization for consistency. This field is only available as a Best component.	All engines
MATCH_BUILDING_NAME	Contains a form of *BUILDING_NAME1* that the Match transform may use during the comparison process. The transform outputs building descriptions in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the transform uses normal width standardization for consistency. ℹ Note For China, the transform removes building indicators. The field is available only as a Best component.	All engines

Output field name	Description	Engine
MATCH_COUNTRY	Contains a form of *ISO_COUNTRY_CODE_2CHAR* that the Match transform may use during the comparison process. The transform outputs data in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the field uses normal width standardization for consistency. The field is available only as a Best component.	All engines
MATCH_FLOOR_NUMBER	Contains a form of *FLOOR_NUMBER* that the Match transform may use during the comparison process. The transform does not output floor descriptions and qualifiers. The transform outputs data in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the field uses normal width standardization for consistency. The field is available only as a Best component.	All engines

Output field name	Description	Engine
MATCH_LOCALITY	Contains a form of *LOCALITY* that the Match transform may use during the comparison process. Standardization settings do not affect the output. The transform outputs *LOCALITY1_OFFICIAL* when a locality or better level assignment is made; otherwise, the transform outputs the *LOCALITY1_NAME* data. The transform does not output locality codes, qualifiers, or descriptions. The transform outputs data in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the field uses normal width standardization for consistency. The field is available only as a Best component.	All engines

Output field name	Description	Engine
MATCH_LOCALITY2	Contains a form of *LOCALITY* that the Match transform may use during the comparison process. Standardization settings to not affect the output. The transform outputs *LOCALITY2_OFFICIAL* when a locality or better level assignment is made; otherwise, the transform outputs the *LOCALITY2_NAME*. The transform does not output locality codes, qualifiers, or descriptions. The transform outputs data in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the field uses normal width standardization for consistency. **i Note** For China, Japan, South Korea, and Taiwan, the transform outputs *LOCALITY2–4_OFFICIAL* or *LOCALITY2–4_NAME*, if present. For all other countries or regions, the transform outputs only *LOCALITY2_OFFICIAL*I or *LOCALITY2_NAME*. The field is available only as a Best component.	All engines

Output field name	Description	Engine
MATCH_POSTCODE1	Contains a form of *POSTCODE1* that the Match transform may use during the comparison process. The transform outputs data in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the field uses normal width standardization for consistency. The field is available only as a Best component.	All engines
MATCH_PRIMARY_DIRECTIONAL	Contains a form of *PRIMARY_PREFIX1* and *PRIMARY_POSTFIX1* that the Match transform may use during the comparison process. Standardization settings do not affect the output. The transform outputs the abbreviated form, if available. If a prefix and postfix are both present, the transform outputs them separated by a space. The transform outputs data in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the field uses normal width standardization for consistency. The field is available only as a Best component.	All engines

Output field name	Description	Engine
MATCH_PRIMARY_NAME	Contains a form of *PRIMARY_NAME1* that the Match transform may use during the comparison process. Standardization settings do not affect the output. The transform outputs data in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. The transform removes prefix, postfix, suffix, and type data. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the field uses normal width standardization for consistency. The field is available only as a Best component.	All engines

Output field name	Description	Engine
MATCH_PRIMARY_NAME2	Contains a form of *PRIMARY_NAME2* that the Match transform may use during the comparison process. Standardization settings do not affect the output. The transform outputs data in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. The transform removes prefix, postfix, suffix, and type data. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the field uses normal width standardization for consistency. **i Note** For Brazil, China, and Japan, *PRIMARY_NAME2-4* are output, if present. For all other countries, only *PRIMARY_NAME2* is output. The field is available only as a Best component.	All engines

Output field name	Description	Engine
MATCH_PRIMARY_NUMBER	Contains a form of *PRIMARY_NUMBER* that the Match transform may use during the comparison process. The transform outputs data from only the *PRIMARY_NUMBER* and *PRIMARY_NUMBER_EXTRA* fields, and not the *PRIMARY_NUMBER_DESCRIPTION* field. The transform outputs data in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the field uses normal width standardization for consistency. The field is available only as a Best component.	All engines

Output field name	Description	Engine
MATCH_PRIMARY_TYPE	Contains a form of *PRIMARY_TYPE* that the Match transform may use during the comparison process. Standardization settings do not affect the output. The transform outputs the abbreviated primary type, if available. The transform outputs data in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the field uses normal width standardization for consistency. **i Note** For Brazil, China, and Japan, *PRIMARY_NAME2-4* are output, if present. For all other countries, only *PRIMARY_NAME2* is output. The field is available only as a Best component.	All engines
MATCH_REGION	Contains a form of *REGION1_NAME* that the Match transform may use during the comparison process. The transform outputs data in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the field uses normal width standardization for consistency. The field is available only as a Best component.	All engines

Output field name	Description	Engine
MATCH_STAIRWELL_NAME	Contains a form of *STAIRWELL_NAME* that the Match transform may use during the comparison process. The transform does not output stairwell descriptions. The transform outputs data in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the field uses normal width standardization for consistency. The field is available only as a Best component.	All engines
MATCH_UNIT_NUMBER	Contains a form of *UNIT_NUMBER* that the Match transform may use during the comparison process. The transform does not output unit descriptions and qualifiers. The transform outputs data in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the field uses normal width standardization for consistency. The field is available only as a Best component.	All engines

Output field name	Description	Engine
MATCH_WING_NAME	Contains a form of *WING_NAME* that the Match transform may use during the comparison process. The transform does not output wing descriptions. The transform outputs data in uppercase and removes diacritical characters and apostrophes. The transform replaces other punctuation and multiple spaces with a single space. **For China, Russia, South Korea, and Taiwan:** The transform transliterates non-Latin scripts to Latin to support inter-script matching. For other CJK-script countries or regions, the field uses normal width standardization for consistency. The field is available only as a Best component.	All engines
MULTILINE1-12	Contains a line of any data. The type of data in this line may vary from record to record.	All engines
NW_FORMATTED_POSTCODE	Contains the postcode in the format that SAP applications require.	All engines
NW_POSTCODE_IN_SUPPORTED_FORMAT	Contains an indicator that specifies whether the *NW_FORMATTED_POSTCODE* output field is populated.	All engines
PMB_FULL	Contains private mailbox information.	All engines
PNAME_SECONDARY_ADDR	Contains the full primary name, with no associated primary number, and the full secondary address.	All engines
POINT_OF_REFERENCE1_2	Contains combined data from the following output fields: • *POINT_OF_REFERENCE1* • *POINT_OF_REFERENCE2*	All engines
POINT_OF_REFERENCE1-2	Contains a well known place or easily visible location to help locate an address. For example, “Opposite of Citibank ATM”.	G

Output field name	Description	Engine
POST_OFFICE_NAME	Contains the name or numeric representation for a post office. For example, "01" BP 1012.	G
POSTCODE_DESCRIPTION	Contains a word that indicates a postal code. The transform populates this field only when the information is available on input. **Example** • **Brazil:** CEP. CEP stands for Código de Endereçamento Postal, and is output as "CEP 52041-970". • **China:** 邮编 • **Japan:** 〒	G
POSTCODE_FULL	Contains a complete postcode based on country. • **Australia:** Complete four-digit postal code. • **Canada:** Complete six-character postal code, such as FSA + LDU. • **Global Address:** Complete postal code. • **USA:** The full ZIP Code with a hyphen (10 characters). • **Japan:** The seven-digit postcode.al	All engines
POSTCODE_IN_VALID_FORMAT	Contains an indicator that the postcode is in the correct format as defined by the postal authority for that country.	All engines

Output field name	Description	Engine
POSTCODE_PREFIX	Contains the postcode prefix that some European countries use to differentiate one countries postcode from another. A prefix helps identify the country of the postcode. The prefix is useful for postal codes from countries that use the same format. For example, many countries use the four- or five-digit format. You can avoid confusion when sending mail to or from the European country when the postal code contains the country code as a prefix. The codes used are generally based on License plate codes (D for Germany or F for France) rather than ISO codes.	G
POSTCODE1	Contains the postcode, which varies by country. • **Australia:** Four-digit postcode. • **Canada:** First three characters (FSA) of the postal code. • **Global Address:** Postal code. • **Japan:** The first three digits of the postal code. • **USA:** Five-digit primary postal code (ZIP Code). Does not include the four-digit secondary postal code (ZIP4). • **China:** A software-generated default locality level postcode might be assigned when the transform cannot obtain an actual postcode . **i Note** When the transform assigns a software-generated default locality level postcode, it generates a status code for the record, with a sixth character of P or Q.	All engines

Output field name	Description	Engine
POSTCODE2	Contains the secondary postal code. • **Canada:** The last three characters (LDU) of the postal code. • **Japan:** The last four digits of the postal code. • **USA:** The four-digit ZIP Code, which follows the primary postal code on a mail piece with a hyphen placed between, for example, 54601-1234.	All engines
PRIMARY_ADDRESS	Contains the primary address line, such as the street address or post office box. Does not include secondary address information such as apartment. **Japan:** The full block data.	All engines
PRIMARY_ADDRESS_DELIVERY_DUAL	Contains combined data from the following output fields: • *PRIMARY_ADDRESS1-4 (DELIVERY)* • *PRIMARY_ADDRESS1-4 (DUAL)*	All engines
PRIMARY_DELIVERY_MODE	Contains the delivery mode for a street served by a route-type address, such as Rural Route.	C
PRIMARY_DELIVERY_NUMBER	Contains the delivery number for a street served by a route-type address, such as Rural Route.	C
PRIMARY_NAME_FULL1	Contains the primary name, primary type, primary prefix, and primary postfix.	All engines
PRIMARY_NAME_FULL1_2	Contains combined data from the following output fields: • *PRIMARY_NAME_FULL1* • *PRIMARY_NAME_FULL2*	All engines
PRIMARY_NAME_FULL1_4	Contains combined data from the following output fields: • *PRIMARY_NAME_FULL1* • *PRIMARY_NAME_FULL2* • *PRIMARY_NAME_FULL3* • *PRIMARY_NAME_FULL4*	All engines

Output field name	Description	Engine
PRIMARY_NAME_FULL2	Contains combined data from the following output fields: • *PRIMARY_NAME2* • *PRIMARY_TYPE2* • *PRIMARY_PREFIX2* • *PRIMARY_POSTFIX2*	G
PRIMARY_NAME_FULL3_4	Contains combined data from the following output fields: • *PRIMARY_NAME_FULL3* • *PRIMARY_NAME_FULL4*	All engines
PRIMARY_NAME_FULL3-4	Contains the primary name and primary type.	G
PRIMARY_NAME1	Contains the street name description, which is typically a street name or box description. **Japan:** Block (chome, kumi, Hokkaido go), subblock (banchi, gaiku, tochi kukaku). The Post office name description (yuubinnkyoku or siten).	All engines
PRIMARY_NAME2	Contains the second street and name description, typically a street name, or box description. **Japan:** Additional block and subblock information.	G
PRIMARY_NAME3	Contains the street name, delivery mode, and so on. **Japan:** Additional block and subblock information.	G
PRIMARY_NAME4	Contains the street name, delivery mode, and so on. **Japan:** Additional block and subblock information.	G
PRIMARY_NUMBER	Contains the premise number, rural route number, or PO Box number. In some cases, it may include a range.	All engines

Output field name	Description	Engine
PRIMARY_NUMBER_DESCRIPTION	Contains a description that precedes the primary number. For example, KM for Kilometer or Blk. for block. • **Japan:** The postal number identifier 号 (go) or house number description 号 (go). • **China:** The description after the street number. For example, 号 (hao).	G
PRIMARY_NUMBER_EXTRA	Contains data found near the parsed primary number, which in most cases cannot be identified or does not belong in a standardized address. **Japan:** The postal box identifier.	G
PRIMARY_NUMBER_FULL	Contains the primary number, primary number description, and primary number extra.	All engines
PRIMARY_POSTFIX1	Contains the abbreviated or non abbreviated directional, such as N, South, NW, that follows a street name. **Japan:** Directional that follows block or subblock.	All engines
PRIMARY_POSTFIX2	Contains the abbreviated or non abbreviated directional, such as N, South, NW, SE, that follows a street name. The transform uses the standardization setting for directional style to determine whether to output abbreviated or non abbreviated format. **Japan:** Directional that follows block or subblock.	G
PRIMARY_PREFIX1	Contains the abbreviated or non abbreviated directional, such as N, South, NW, SE, that precedes a street name. The transform uses the standardization setting for directional style to determine whether to output abbreviated or non abbreviated format. **Japan:** Directional that precedes block or subblock.	G, U

Output field name	Description	Engine
PRIMARY_PREFIX2	Contains the abbreviated or non abbreviated directional, such as N, South, NW, SE, that precedes a street name. The transform uses the standardization setting for directional style to determine whether to output abbreviated or non abbreviated format. **Japan:** Directional that precedes block or subblock.	G
PRIMARY_SECONDARY_ADDR_DELIVERY_DUAL	Contains combined data from the following output fields: • *PRIMARY_SECONDARY_ADDRRESS (DELIVERY)* • *PRIMARY_SECONDARY_ADDRRESS (DUAL)*	All engines
PRIMARY_SECONDARY_ADDRESS	Contains the combined data from the following output fields: • *PRIMARY_ADDRESS* • *SECONDARY_ADDRESS*	All engines
PRIMARY_TYPE1	Contains the type of primary name. For example, rue, strasse, street, Ave, or Pl.	All engines
PRIMARY_TYPE2-4	Contains the type of primary name. For example, rue, strasse, street, Ave, or Pl.	G
QUALITY_CODE	Contains 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 when an information code is not generated. For more information about status codes, see “Global Address Cleanse quality codes”.	All engines
REGION1	Contains either the *REGION1_NAME* or *REGION1_SYMBOL* output based on the standardization setting for region style.	All engines

Output field name	Description	Engine
REGION1_2_FULL	Contains combined data from the following output fields: • *REGION1_FULL* • *REGION2_FULL* **USA:** Does not include *REGION2_FULL*.	All engines
REGION1_2_NAME	Contains combined data from the following output fields: • *REGION1_NAME* • *REGION2_NAME* **USA:** Does not include *REGION2_NAME*.	All engines
REGION1_CODE	Contains the region code, which may be the ISO region code.	All engines
REGION1_DESCRIPTION	Contains the region description.	G
REGION1_FULL	Contains combined data from the following output fields: • *REGION1* • *REGION1_DESCRIPTION*	All engines
REGION1_NAME	Contains the fully spelled Region name.	All engines
REGION1_SYMBOL	Contains the abbreviated form of the Region name.	All engines
REGION2	Contains either the *REGION2_NAME* or the *REGION2_SYMBOL* output data based on the standardization option you set for region style. **USA:** Contains the county name. **Poland:** The district name (powiat).	G, U
REGION2_CODE	Contains the region code, which may be the ISO region code.	All engines
REGION2_DESCRIPTION	Contains the region 2 description.	G
REGION2_FULL	Contains combined data from the following output fields: • *REGION2* • *REGION2_DESCRIPTION*	G
REGION2_NAME	Contains the fully spelled Region 2 name.	G

Output field name	Description	Engine
REGION2_SYMBOL	Contains the abbreviation of the Region 2 name.	G
ENGINE_NAME	Contains the name of the engine that you select in the transform configuration.	All engines
REMAINDER_EXTRA_PMB_FULL	Contains combined data from the following output fields: • *REMAINDER_FULL* • *EXTRA1-2* • *PMB_FULL*	All engines
REMAINDER_FULL	Contains all remainder information, including data from the *ADDRESS_LINE_REMAINDER1-4* and *LASTLINE_REMAINDER1-4* output fields.	All engines
ROOM_FULL	Contains combined data from the following output fields: • *UNIT_DESCRIPTION* when it contains “room” or a variant • *UNIT_NUMBER*	All engines
ROOM_NUMBER	Contains the unit number for units that are variations of “room.” For example, RM, RMS, ROOM, ROOMS, RM., RMS, 号室, 室, 호.	All engines

Output field name	Description	Engine
ROW_ID	Contains an internal ID generated by the transform that uniquely identifies a row processed by that transform. Use this field to join the input and output to the applicable non-summary data quality statistics table. The software issues a warning if you select to generate any non-summary data quality statistics tables in the transform, and you don't include the *ROW_ID* output field. The software also issues a warning when you map this output field and don't select to generate any non-summary data quality statistics tables in the transform. Applicable for the following transforms: • Data Cleanse • Global Address Cleanse • Geocoder	All engines
SECONDARY_ADDRESS	Contains the block, floor, unit, stairwell, or wing data on one line.	All engines
SECONDARY_ADDRESS_NO_FLOOR_NO_ROOM	Contains combined data from all *SECONDARY_FULL* output fields except the following output fields: • *FLOOR_FULL* • *ROOM_FULL*	All engines
SECONDARY_ADDRESS_NO_FLOOR	Contains combined data from all *SECONDARY_FULL* output fields except *FLOOR_FULL*.	All engines
SECONDARY_ADDRESS_NO_ROOM	Contains combined data from all *SECONDARY_FULL* output fields except *ROOM_FULL*.	All engines
SINGLE_ADDRESS	Contains the full address and last line in one component.	All engines
STAIRWELL_DESCRIPTION	Contains the identifier for an entrance or stairwell of a building, such as, "Entrada" 1.	G

Output field name	Description	Engine
STAIRWELL_FULL	Contains combined data from the following output fields: • *STAIRWELL_DESCRIPTION* • *STAIRWELL_NAME*	All engines
STAIRWELL_NAME	Contains the name or number of an entrance or stairwell for a building, such as, Entrada "1."	G

Output field name	Description	Engine
STATUS	Contains the suggestion status generated as the result of looking up the current record and performing suggestion list processing. • *A*: Primary address-line suggestions available. • *B*: Primary and secondary ranges are invalid. • *C*: Address was not found. • *CS*: Country list is available. This is applicable when the *Enable Suggestion Lists* option is set to *YES*. This is for the Country ID engine only. • *F*: Floor range is invalid. • *L*: Lastline suggestions available. • *L1*: Locality1 list available (when the Enable Lastline Drilldown option is Yes; Global Address engine only). • *L2*: Locality2 list available (when the Enable Lastline Drilldown option is Yes; Global Address engine only). • *L3*: Locality3 list available (when the Enable Lastline Drilldown option is Yes; Global Address engine only). • *L4*: Locality4 list available (when the Enable Lastline Drilldown option is Yes; Global Address engine only). • *N*: No suggestions available. • *PC*: Postcode suggestions available. • *R*: Primary range is invalid. • *R1*: Region1 list available (when the Enable Lastline Drilldown option is Yes; Global Address engine only). • *R2*: Region2 list available (when the Enable Lastline Drilldown option is Yes; Global Address engine only). • *S*: Unit range is invalid. • *U*: Secondary address-line suggestions available.	All engines

Output field name	Description	Engine
STATUS_CODE	Contains the six-character status code that always starts with an S. This code describes what parts of the address changed during processing.	All engines
UNIT_DESCRIPTION	Contains the unit description, such as "Apartment" or "Flat." **Japan:** The unit description, such as gousitsu.	All engines
UNIT_FULL	Contains combined data from the following output fields: • *UNIT_DESCRIPTION* • *UNIT_NUMBER* • *UNIT_QUALIFIER*	All engines
UNIT_NUMBER	Contains the unit number, such as 100 in "Apartment 100."	All engines
UNIT_QUALIFIER	Contains the additional word that precedes or follows the unit information.	G
WING_DESCRIPTION	Contains the wing identifier within a building. For example, returns "Wing" from "West Wing."	G
WING_FULL	Contains combined data from the following output fields: • *WING_DESCRIPTION* • *WING_NAME*	All engines
WING_NAME	Contains the name or number of a wing within a building. For example, returns "West" from "West Wing."	G

Related Information

Global Address Cleanse status codes [page 812]
Global Address Cleanse quality codes [page 817]
Global Address Cleanse information codes [page 809]

3.5.9.19 Global Address Cleanse suggestion list output fields

Include the Global Address Cleanse output fields that have the *Field_Category* of SUGGESTION so that the transform outputs the suggestion list-specific information.

In the Global Address Cleanse transform, you enable components from the *Suggestion List Components* group of options. The software outputs the suggestion information for each component to the *Suggestion_List* output field in an XML string. If you do not include the *Suggestion_List* output field, the software does not output the specific components that you choose.

Field name	Description
Status	Contains the suggestion status generated as a result of suggestion list processing. • *A*: Primary address line suggestions available. • *AM*: Follow-up primary address-line suggestions available. • *B*: Primary and secondary ranges are invalid. • *C*: Address not found. • *F*: Floor range is invalid. • *L*: Lastline suggestions available. • *L1*: Locality1 list available. Applicable when you set the *Enable Lastline Drilldown* option to *Yes*. Applies only to Global Address engine • *L2*: Locality2 list available. Applicable when you set the *Enable Lastline Drilldown* option to *Yes*. Applies only to Global Address engine. • *L3*: Locality3 list available. Applicable when you set the *Enable Lastline Drilldown* option to *Yes*. Applies only to Global Address engine. • *L4*: Locality4 list available. Applicable when you set the *Enable Lastline Drilldown* option to *Yes*. Applies only to Global Address engine. • *N*: No suggestions available. • *PC*: Postcode suggestions available. • *R*: Primary range is invalid. • *R1*: Region1 list available. Applicable when you set the *Enable Lastline Drilldown* option to *Yes*. Applies only to Global Address engine. • *R2*: Region2 list available. Applicable when you set the *Enable Lastline Drilldown* option to *Yes*. Applies only to Global Address engine. • *S*: Unit range is invalid. • *U*: Secondary address-line suggestions available. • *UM*: Follow up secondary address-line suggestions available.

Field name	Description
Error	Contains the error status code generated by suggestion list processing. Possible output values are 0- 6: • *0*: No suggestion selection error. • *1*: Blank suggestion selection or entry. • *2*: Invalid suggestion selection. • *3*: Invalid primary range. • *4*: Invalid floor range. • *5*: Invalid unit range. • *6*: Too many possible results to generate a suggestion list. Provide more information, such as a postal code, region, or locality.
Count	Contains a nonnegative number that represents the number of individual suggestion selections generated from querying the current record. If the transform does not generate a suggestion list for the input record, the value is 0.
Suggestion_List	Contains a string of the combined suggestion list components that you enabled in the transform and the suggestion values. The format adheres to the *Output Style*, *Delimiter*, and *Field_Delimiter* values you set for *Suggestion List Components* options.
More_Suggestions_Available	Contains an indicator that there are more suggestions to view: • *Y*: There are more suggestions to view. • *N*: There are no more suggestions to view. A *Y* value indicates that more than one page of data is available for the current address. **Example** When *More_Suggestions_Available* = *Y*, the software enables a "page up" or "page down" mechanism to give the user access to all of the data. When the user selects page down, the address is reprocessed with a reply of `GOTOPAGE(2)`. The actual number of pages available depends on the following factors: • The number of suggestions the software finds for the address. • Your settings in the transform options *Max_Number_Address_Lines* and *Max_Number_Lastlines*.

3.5.9.20 Global Address Cleanse NW input fields

Use the NW input fields for working with data from SAP applications.

Ensure that you do not use non-NW fields with NW fields. Use the fields properly to avoid unexpected results in your data.

Keep in mind that the transform processes NW input fields as multiline fields, even though they appear as discrete fields. The software maps NW fields internally to *MULTILINE1-12* before it performs normal Global Address Cleanse processing. If you exclude an NW input field from your mapping, the software maps the multiline that would have been mapped to the missed NW field to the next available NW input field.

The following table contains an alphabetical list of the NW fields and the applicable engines. Engines include the following:

- Canada (C)
- Global Address (G)
- USA (U)

To view properties for each field, see the fields listed in the transform *Input* tab.

> ⚠ Caution
>
> Use the NW_ fields properly to avoid unexpected results in your data. For more information, see Mapping Global Address Cleanse NW input fields [page 599].

Input field	Description	Engine
NW_BUILDING	Contains the building information. **Requirement:** When you map this input field, also map the *NW_STREET* field.	All engines
NW_CITY1	Contains the locality. Required field. The transform requires that you map the *NW_CITY1* and *NW_CITY2* input fields in sequence.	All engines
NW_CITY2	Contains additional locality or district information.	All engines
NW_COUNTRY	Contains the country name. Required field.	All engines

Input field	Description	Engine
NW_FIRM	Contains the input firm name. Including a firm name in some addresses helps the software better assign and verify addresses. Also, some postal authorities require a firm name for specific discounts. For example, a firm name is required for France PO Box addresses that use the CEDEX postal code numbering system.	G
NW_FLOOR_NUM	Contains the floor number. **Requirement:** When you map this input field, also map the *NW_STREET* field.	All engines
NW_HOME_CITY	Contains additional locality information.	All engines
NW_HOUSE_NUM1	Contains the house number. The transform requires that you map the *NW_HOUSE_NUM1* and *NW_HOUSE_NUM2* fields in sequence. **Requirement:** When you map this input field, also map the *NW_STREET* field.	All engines
NW_HOUSE_NUM2	Contains additional house number information. The transform requires that you map the *NW_HOUSE_NUM1* and *NW_HOUSE_NUM2* fields in sequence. **Requirement:** When you map this input field, also map the *NW_STREET* field.	All engines
NW_LOCATION	Contains additional street information. **Requirement:** When you map this input field, also map the *NW_STREET* field.	All engines
NW_PO_BOX_CITY	Contains the locality. When you use one *NW_PO_BOX_*-type field, the transform requires that you include all *NW_PO_BOX_*-type fields in your mapping.	All engines

Input field	Description	Engine
NW_PO_BOX_COUNTRY	Contains the country. When you use one *NW_PO_BOX_*-type field, the transform requires that you include all *NW_PO_BOX_*-type fields in your mapping.	All engines
NW_PO_BOX_POSTCODE	Contains the postcode. When you use one *NW_PO_BOX_*-type field, the transform requires that you include all *NW_PO_BOX_*-type fields in your mapping.	All engines
NW_PO_BOX_REGION	Contains the state, province, or region. When you use one *NW_PO_BOX_*-type field, the transform requires that you include all *NW_PO_BOX_*-type fields in your mapping.	All engines
NW_PO_BOX	Contains the PO Box number. When you use one *NW_PO_BOX_*-type field, the transform requires that you include all *NW_PO_BOX_*-type fields in your mapping.	All engines
NW_POSTCODE	Contains the postcode. Required field.	All engines
NW_REGION	Contains the state, province, or region. Required field.	All engines
NW_ROOM_NUM	Contains the room number. Required field.	All engines
NW_STR_SUPPL1	Contains additional street information. The transform requires that you map the *NW_STR_SUPPL1-3* fields in sequence. **Requirement:** When you map this input field, also map the *NW_STREET* field.	All engines

Input field	Description	Engine
NW_STR_SUPPL2	Contains additional street information. The transform requires that you map the *NW_STR_SUPPL1-3* fields in sequence. **Requirement:** When you map this input field, also map the *NW_STREET* field.	All engines
NW_STR_SUPPL3	Contains additional street information. The transform requires that you map the *NW_STR_SUPPL1-3* fields in sequence. **Requirement:** When you map this input field, also map the *NW_STREET* field.	All engines
NW_STREET	Contains the primary street name. Required field.	All engines

Related Information

Mapping Global Address Cleanse NW input fields [page 599]

3.5.9.20.1 Mapping Global Address Cleanse NW input fields

To avoid unexpected results when you use NW input fields, ensure that you follow the specific rules.

The followint table contains rules or restrictions to know when you map the Global Address Cleanse NW input fields.

Rule or restriction	More information
Use NW input fields as a group.	Use the NW input fields with other NW input fields when cleansing address data from SAP applications. Do not mix with non-NW fields.

Rule or restriction	More information
Do not map multiline or address line fields with NW fields.	Although the NW input fields appear discrete, they behave like multiline fields, and the transform processes them as multiline fields. The software maps NW fields internally to *MULTILINE1-12* before it performs normal Global Address Cleanse processing. If you do exclude an NW input field from your mapping, the software maps the multiline that would have been mapped to the missed NW field to the next available NW input field.
The transform requires that you use specific NW input fields for every NW input field mapping.	Required NW input fields include the following: • *NW_CITY1* • *NW_COUNTRY* • *NW_POSTCODE* • *NW_REGION* • *NW_STREET*
Map multiple numbered NW fields in sequence.	The transform requires that you map the following NW input fields in sequence: • *NW_CITY1-2* • *NW_HOUSE_NUM1-2* • *NW_STR_SUPPL1-3*
Requirement for mapping *NW_PO_BOX_*-type input fields.	When you map any of the *NW_PO_BOX_*-type fields, include all *NW_PO_BOX_*-type fields in your mapping.
When you map any of the *NW_PO_BOX_*-type input fields, include specific NW street-level fields.	Include the following street-level input fields when you use any of the *NW_PO_BOX_*-type input fields: • *NW_CITY1* • *NW_COUNTRY* • *NW_POSTCODE* • *NW_REGION* • *NW_STREET*

→ Tip

To obtain a change significance of *N* in the *NW_PO_BOX_CHANGE_SIGNIFICANCE* output field, ensure that you set the *Format Assigned Data* option to *NO*.

CJK script and NW fields

For CJK script input, the transform concatenates several NW input fields into the last multiline field for internal processing.

If the transform finds a descriptor in *NW_HOUSE_NUM2*, the transform processes the multiline as follows (the + sign in the following code indicates that the fields are concatenated with no space between them):

```
NW_Street+NW_House_Num1+NW_House_Num2+<Descriptor>NW_RoomNumber
+<Descriptor>NW_Floor+NW_Building
```

If the transform does not fine a descriptor In *NW_HOUSE_NUM2*, the transform processes the multiline as follows (the + sign in the following code indicates that the fields are concatenated with no space between them):

```
NW_Street+NW_House_Num1+NW_RoomNumber<Descriptor>+NW_Floor<Descriptor>
+NW_Building NW_House_Num2
```

Related Information

Global Address Cleanse NW input fields [page 596]

3.5.9.21 Global Address Cleanse NW_PO_Box output fields

The Global Address Cleanse transform populates the *NW_PO_BOX_* output fields only when you map them to NW input fields.

Use *NW_PO_BOX_* output fields only for the PO Box address portion of SAP business suite software.

The following is a list of the available *NW_PO_BOX_* output fields. The content of each *NW_PO_BOX_* field is identical to its corresponding output field without the NW prefix.

> **Example**
>
> To find the description for the *NW_PO_BOX_ASSIGNMENT_INFO* field, see the output field description for *ASSIGNMENT_INFO*.

- *NW_PO_BOX_ASSIGNMENT_INFO*
- *NW_PO_BOX_ASSIGNMENT_LEVEL*
- *NW_PO_BOX_ASSIGNMENT_TYPE*
- *NW_PO_BOX_DELIVERY_INSTALLATION_FULL*
- *NW_PO_BOX_DELIVERY_POINT*
- *NW_PO_BOX_INFO_CODE*
- *NW_PO_BOX_ISO_COUNTRY_CODE_2CHAR*
- *NW_PO_BOX_ISO_SCRIPT_CODE*
- *NW_PO_BOX_LOCALITY1_FULL*
- *NW_PO_BOX_MATCH_BLOCK_NUMBER*
- *NW_PO_BOX_MATCH_BUILDING_NAME*
- *NW_PO_BOX_MATCH_COUNTRY*
- *NW_PO_BOX_MATCH_FLOOR_NUMBER*

- *NW_PO_BOX_MATCH_LOCALITY*
- *NW_PO_BOX_MATCH_LOCALITY2*
- *NW_PO_BOX_MATCH_POSTCODE1*
- *NW_PO_BOX_MATCH_PRIMARY_DIRECTIONAL*
- *NW_PO_BOX_MATCH_PRIMARY_NAME*
- *NW_PO_BOX_MATCH_PRIMARY_NAME2*
- *NW_PO_BOX_MATCH_PRIMARY_NUMBER*
- *NW_PO_BOX_MATCH_PRIMARY_TYPE*
- *NW_PO_BOX_MATCH_REGION*
- *NW_PO_BOX_MATCH_STAIRWELL_NAME*
- *NW_PO_BOX_MATCH_UNIT_NUMBER*
- *NW_PO_BOX_MATCH_WING_NAME*
- *NW_PO_BOX_NW_FORMATTED_POSTCODE*
- *NW_PO_BOX_NW_POSTCODE_IN_SUPPORTED_FORMAT*
- *NW_PO_BOX_POSTCODE_FULL*
- *NW_PO_BOX_POSTCODE_IN_VALID_FORMAT*
- *NW_PO_BOX_PRIMARY_ADDRESS*
- *NW_PO_BOX_PRIMARY_NUMBER*
- *NW_PO_BOX_PRIMARY_SECONDARY_ADDRESS*
- *NW_PO_BOX_REGION1*
- *NW_PO_BOX_REGION1_FULL*
- *NW_PO_BOX_REGION2*
- *NW_PO_BOX_REGION2_FULL*
- *NW_PO_BOX_STATUS_CODE*

Related Information

Global Address Cleanse output fields [page 554]

3.5.10 Global Suggestion List transform

The Global Suggestion List transform queries addresses with minimal data, and offers suggestions to complete the address.

Global Suggestion List information

Characteristic	Description
	Global Suggestion List icon
Use	The Global Suggestion List transform is A beneficial research tool for managing unassigned addresses from a batch process. Integrate this transform into your custom applications via the Web Service. If you are a programmer looking for details about how to integrate Global Suggestion List functionality, see the *Integrator Guide*. The Global Suggestion List transform requires the two character ISO country code on input. Therefore, you may want to place a transform, such as the Country ID transform, that outputs the ISO_Country_Code_2Char field before the Global Suggestion List transform. Use the Global Suggestion List transform with the Canada, Global Address, and USA engines. **i Note** Global Suggestion List does not support Chinese and Japanese addresses. **i Note** 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.
Content objects	Transform configurations, blueprints, and other content objects.

Global Suggestion List transform option groups [page 604]
Data Services groups the options for the Global Suggestion List transform into categories.

Global Suggestion List transform engines [page 605]
The Engines option group allows you to enable or disable individual engines of the Global Suggestion List transform.

Global Suggestion List transform options [page 606]
This option group contains all of the settings that you need to define when you process data with the Global Suggestion List transform.

Global Suggestion List transform Suggestion List options [page 607]
To determine the information to include in the suggestion list output, set components in the Suggestion List options group.

Global Suggestion List transform input fields [page 609]
Map the input fields to include in the transform processes.

Global Suggestion List transform output fields [page 611]
Map output fields to include in the output from the transform.

Related Information

Global Suggestion List transform option groups [page 604]
Cleanse address data transactionally
Data Quality transforms [page 341]
Download Data Quality blueprints and other content objects [page 340]

3.5.10.1 Global Suggestion List transform option groups

Data Services groups the options for the Global Suggestion List transform into categories.

The Global Suggestion List transform includes the following option groups.

> **i Note**
>
> The Global Suggestion List transform does not support Chinese and Japanese addresses.

Option group	Description
Common: Run as Separate Process	Splits the transform into a separate process.
Engines	Sets the engines for processing. The default value for each country engine is *Yes*. Select *No* to disable the country engine. Countries include: • Canada • Global Address • USA
Canada	Sets the directory path for your address cleanse reference files.
Global Address	Sets the directory path for your address cleanse reference files.
USA	Sets the directory path for your address cleanse reference files.
Reference Files	Specifies the location of your reference files. You can use a substitution variable.

Option group	Description
Options	Set options that define how to process data with the Global Suggestion List transform.
Suggestion List	Set options that define how the transform outputs suggestion lists as well as define the output fields where Data Services posts suggestion information.

Parent topic: Global Suggestion List transform [page 603]

Related Information

Global Suggestion List transform engines [page 605]
Global Suggestion List transform options [page 606]
Global Suggestion List transform Suggestion List options [page 607]
Global Suggestion List transform input fields [page 609]
Global Suggestion List transform output fields [page 611]

3.5.10.2 Global Suggestion List transform engines

The Engines option group allows you to enable or disable individual engines of the Global Suggestion List transform.

> **i Note**
>
> The Suggestion List option does not support Chinese and Japanese addresses.

Option	Description
Canada	Specifies if the engine is enabled or disabled for suggestion List processing. Choose one of the following: *Yes:* Enables the engine. *No:* Disables the engine.
Global Address	Specifies if the engine is enabled or disabled for suggestion List processing. Choose one of the following: *Yes:* Enables the engine. *No:* Disables the engine.
USA	Specifies if the engine is enabled or disabled for suggestion List processing. Choose one of the following: *Yes:* Enables the engine. *No:* Disables the engine.

Parent topic: Global Suggestion List transform [page 603]

Related Information

Global Suggestion List transform option groups [page 604]
Global Suggestion List transform options [page 606]
Global Suggestion List transform Suggestion List options [page 607]
Global Suggestion List transform input fields [page 609]
Global Suggestion List transform output fields [page 611]

3.5.10.3 Global Suggestion List transform options

This option group contains all of the settings that you need to define when you process data with the Global Suggestion List transform.

> **i Note**
>
> The Global Suggestion List transform does not support Chinese and Japanese addresses.

Option	Description
Default Country	Specifies a country to use if the input field is not defined or if the Country input field is blank. Enter a valid two-character country code, or enter *None* if you do not want to use a default country.
Enter Firm Data	Specifies whether to request firm data when the selected suggestion does not have data available. *Yes:* Requests firm data. *No:* Does not request firm data.
Return Single Item Lists	Specifies whether the transform should return suggestion lists that have only one item. *Yes:* Returns suggestion lists that only have one item. *No:* If a suggestion list has only one item, then the single item is automatically selected from the suggestion list and processing continues.

Parent topic: Global Suggestion List transform [page 603]

Related Information

Global Suggestion List transform option groups [page 604]

Global Suggestion List transform engines [page 605]
Global Suggestion List transform Suggestion List options [page 607]
Global Suggestion List transform input fields [page 609]
Global Suggestion List transform output fields [page 611]

3.5.10.4 Global Suggestion List transform Suggestion List options

To determine the information to include in the suggestion list output, set components in the Suggestion List options group.

For each component in the Suggestion List groups, select *Yes* or *No*:

- *Yes*: Outputs the component.
- *No*: Does not output the component.

Lastline Components group

Component	Description
Selection	Returns a unique index number that identifies this suggestion from the others in the returned list.
Locality1-3	Returns the city, town, or suburb. Additional locality information goes in Locality2.
City Addition	Returns unofficial city information that is associated with the locality. For example, there are two German cities named Frankfurt. The larger city of Frankfurt is called Frankfurt am Main and the smaller city is called Frankfurt (Oder). Locality1 would have Frankfurt for these two records, and City Addition would list (Oder) and am Main.
Region1	Returns the state, province, territory, or region of the address.
Postcode1	Returns the postal code or five-digit ZIP Code (USA).
Primary Names Available	Indicates whether or not street data is available for a locality: *Y*: Yes, there are streets. *N*: No, there are not streets.

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.

Option/Option group	Description
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.

Address Components

Option/Option group	Description
Selection	Returns a unique index number that identifies this suggestion from the others in the returned list.
Primary Side Indicator	Indicates if even, odd, or both values are valid. This applies to Street and PO box. *E*: The record covers the even-numbered values. *O*: The record covers the odd-numbered values. *B*: The record covers both the even- and odd-numbered values.
Firm	Returns the name of a firm, company, or organization.
Multiline1-6	Returns individual formatted address lines. This will not include country information, as it will be output in a separate field.
Postcode1	Returns the postal code.

Suggestion List options

Option/Option group	Description
Output Style	Specifies the format for the output suggestion list data. *Delimited:* Outputs the suggestion list data in a delimited text format, with the delimiters specified in the Delimiter and Field Delimiter options. *XML:* Outputs the suggestion list data as hierarchical XML. If you integrate suggestion lists via the web service, you are likely to use this option. You can then use the XML tools you own to parse through the suggestion list data.
Delimiter	Specifies a character to use to separate each suggestion in a suggestion list. This value is considered only if the Style option is set to Delimited. This value can be any character or string. Common delimiters include a pipe symbol (|), or a string of multiple asterisks (***). This value should differ from the Field Delimiter value.

Option/Option group	Description
Field Delimiter	Specifies a character to use to separate each field in a single suggestion. This value is considered only if the Style option is set to Delimited. Each suggestion can be made up of one or more fields. If you choose to retrieve multiple fields per suggestion, those fields are separated with the character specified here. This value can be any character or string. The default value is a pipe symbol (|). This value should differ from the Delimiter value.

Parent topic: Global Suggestion List transform [page 603]

Related Information

Global Suggestion List transform option groups [page 604]
Global Suggestion List transform engines [page 605]
Global Suggestion List transform options [page 606]
Global Suggestion List transform input fields [page 609]
Global Suggestion List transform output fields [page 611]

3.5.10.5 Global Suggestion List transform input fields

Map the input fields to include in the transform processes.

The following table contains input fields to include in the input mapping for the Global Suggestion List transform. The fields are listed alphabetically.

> **i Note**
>
> The Global Suggestion List transform uses all fields provided on input to select a match. If the fields you include do not exist in the data, the transform returns an error stating "no results." If you do not own address-level data for a country, do not include address-level fields in the input mapping. Remove address-level fields when address-level data is available but the address list states there are no results. Removing the address-level fields in this case allows you to view locality-level data for the address.

> **i Note**
>
> Global Suggestion List does not support Chinese and Japanese addresses.

Input field name (Global Suggestion List)	Description
Country	Specifies the country to look up in the query. This field contains the two-character country ISO code, not a country name. If your data does not contain the country code, place a transform such as the Country_ID transform before the Global Suggestion List transform to generate the country codes. 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 the software requires more information to complete the query. Reply1-5 can also contain the reply when you need to make a selection from a list. Possible types of generated suggestion lists are lastline, primary name, and address.
Start_Selection	Specifies the starting list number. If left blank, the default value is 1.
Script_Default	Sets the output script. This option is valid for Greece. If the input data is non-numeric, then the script type is determined by the script of the input data. **Example** When input data contains only numeric data for Greece, and this option is set to 2, the transform outputs the generated suggestion lists in Greek script. If the option is set to 1, the transform outputs the generated suggestion lists in Latin script.

Parent topic: Global Suggestion List transform [page 603]

Related Information

Global Suggestion List transform option groups [page 604]
Global Suggestion List transform engines [page 605]
Global Suggestion List transform options [page 606]
Global Suggestion List transform Suggestion List options [page 607]
Global Suggestion List transform output fields [page 611]

3.5.10.6 Global Suggestion List transform output fields

Map output fields to include in the output from the transform.

The following are fields to include in the output mapping for the Global Suggestion List transform. The fields are listed alphabetically.

> **i Note**
>
> Global Suggestion List does not support Chinese and Japanese addresses.

Output field name (Global Suggestion List)	Description
Country_Name	Returns the fully spelled country name, in English.
Data_Type	Returns a single-character code that indicates what type of additional data is required to complete processing. Possible output values are: *N:* No additional data is needed. *A:* Primary address data is needed. *F:* Firm data is needed. *R:* Primary range data is needed. *S:* Secondary range data is needed.
Error	Posts the error status generated as the result of looking up the current record and performing query processing. Possible output values are: *0:* There were no query errors. *1:* There was a system error while performing the query. *2:* The suggestion selection was invalid. For example, a selection of 8 is made and there are only 5 entries.
Firm	Specifies the firm name for the address.
Multiline1-6	Returns a line that may contain any data. The type of data in this line may vary from record to record. If you want to output the postal code, you cannot use one of these Multiline output fields. Use the Postcode1 field.
Postcode1	Returns the postcode. *Canada and Global Address:* Postal code. *USA:* Five-digit primary postal code (ZIP Code). Does not include the four-digit secondary postal code (ZIP4).

Output field name (Global Suggestion List)	Description
Status	Returns a code indicating query status generated as the result of processing the input record and performing Global Suggestion List processing. Possible output values are: *C:* Querying is complete. *D:* More data is needed. *E:* There was an error. *P:* A suggestion list was generated.
Suggestion_Count	Returns the number of individual suggestion selections generated as the result of querying the current record. A nonnegative value is output. If the input record did not generate a suggestion list, this field contains a value of 0. Your application developer uses this field to know how many suggestion selections must be displayed to users of your custom application.
Suggestion_List	Contains the list of suggestions based on the Suggestion List Option settings that you set in the Global Suggestion List transform.
Suggestion_Type	Returns a code indicating what type of suggestion list was generated. Possible output values are: *N:* No suggestion list was generated. *A:* An address suggestion list was generated. *L:* A lastline suggestion list was generated. *S:* A secondary suggestion list was generated.
System_Error_ Description	Posts the current Global Suggestion List system error as a descriptive string.

Output field name (Global Suggestion List)	Description
System_Error_ Number	Posts the current system error as a number. The values are: *0:* No error. *1:* Invalid postcode. *2:* Invalid street. *3:* Invalid town. *4:* More information needed. *5:* Street information needed. *6:* No input given. *7:* Postcode numeric. *8:* Town needed. *9:* Town or postcode needed. *10:* No street information available. *11:* Country blank. *12:* Invalid country. *13*: No results. *14:* Address needed. *15:* Premise needed. *16:* Firm needed.
Warning	Posts the warning status generated as the result of looking up the current record and performing query processing. The values are: *0:* There were no query warnings. *1:* An incomplete suggestion list was generated. The maximum number of elements that can be placed in a suggestion list is 200. *2:* An invalid premise was entered. *3:* An invalid unit was entered.

Parent topic: Global Suggestion List transform [page 603]

Related Information

Global Suggestion List transform option groups [page 604]
Global Suggestion List transform engines [page 605]
Global Suggestion List transform options [page 606]

Global Suggestion List transform Suggestion List options [page 607]
Global Suggestion List transform input fields [page 609]

3.5.11 Match transform

Use the Match transform to find duplicate records in your data sources and process them based on defined business rules.

Match information

Characteristic	Description
	Match icon
Use	Set rules for matching records, and establish thresholds and scores that the software uses to determine near matches and matching records. After the transform completes processing, it sends matching and unique records on to the next transform in the data flow. For best results, process the data in which you are attempting to find matches through data cleanse and Address Cleanses standardization processes. Best practice is to include other Data Quality transforms before the Match transform in your data flow.
Content objects	The Match transform has sample transform configurations that help you to set up your data flow. Use the Match Wizard to help set up matching options. Additionally, download blueprints from the SAP SCN page.

The Match transform is only one tool 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*.

> **i Note**
>
> Set options for the Match transform in the *Match Editor*. With the Match transform selected in your data flow, select *Tools* > *Match Editor.* The *Match Editor* contains options so that you can create fine-tuned matching criteria. Settings that you make in the *Match Editor* appear in the *Options* tab of the Match transform editor.

Match transform editor options [page 615]

To configure match processes, such as report generation, and match set formation, set options by accessing the Match editor through the *Tools* menu.

Match transform Data Salvage tab [page 617]

Set options in the *Data Salvage* tab in the Match editor to perform data salvage on your data for more complete matching results.

Match or Associate transforms Input Source options [page 618]
To select the input sources to track statistics, and to provide additional functionality, set options in the *Input Source* group .

Match transform Source Group options [page 621]
Include a Source Group in your Match transform set up to include additional statistics in certain Match reports.

Match transform match level options [page 629]
Set options that effect match processing at the match level.

Match transform match criteria table [page 633]
Use the match criteria table to navigate to a particular criteria by double-clicking a row in the table.

Match transform compare table [page 654]
Using compare tables enables you to compare records within sources, compare records across sources, or a combination of both.

Match and Associate transforms Post Match Processing [page 655]
Add operations to the *Post Match Processing* group or the *Post Association Processing* group such as generating statistics and creating a best record.

Match transform dynamic input field for [page 669]
Set up a match criteria using Geo Proximity to include the dynamic input field for the Match transform.

Match transform output fields [page 670]
The Match transform generates output fields for various types of operations run either pre- or post-match.

3.5.11.1 Match transform editor options

To configure match processes, such as report generation, and match set formation, set options by accessing the Match editor through the *Tools* menu.

Open the Match editor by selecting the Match transform in your data flow and then selecting *Tools* > *Match Editor.*

Match Editor option descriptions

Option	Description
Perform matching	Specifies whether SAP Data Services compares records and performs match-related operations. This option is located in the upper left corner of the *Match Editor*. *Perform matching* is selected by default in all Match transform configurations except the Base_Match transform configuration.

Option	Description
Match set name	Specifies a name for the match set currently open. Each Match transform in your data flow represents a match set. This option is located at the top of the main *Match Transform Options Editor* pane. If you use the Match Wizard to generate the Match transform, the Wizard populates the *Match set name*, but you optionally can rename it. Be sure that this name is unique within the data flow. The software uses the name in the match reports to differentiate data processed by one match set versus another.
Match engine	Specifies the match engine to use. Base your selection on the type of data in your source. If you use the Multinational strategy in the Match Wizard, the Wizard sets this option to *Latin1* for all match sets. *Chinese:* Specifies that the Match transform processes Chinese data in Chinese script. *Japanese:* Specifies that the Match transform processes Japanese data in Japanese script. *Korean:* Specifies that the Match transform processes Korean data in Korean script. *Latin1:* Specifies that the Match transform processes Latin1 data. Latin1 is the data used throughout the Americas, Western Europe, Oceania, and much of Africa. *Other_Non_Latin1:* Specifies that the Match transform processes non-Latin1 data, other than Chinese, Japanese, Korean, and Taiwanese, such as Russian, Greek, Hebrew, Arabic, and others. *Taiwanese:* Specifies that the Match transform processes data from Taiwan in Taiwanese script. For optimum accuracy and performance, filter your multinational data to separate match transforms with the appropriate match engine selected. If you use the Match Wizard, select the Multinational strategy. By default, the transform uses the Latin1 match engine. If you attempt to process data that is not latin1 with the Latin1 engine, the results are unpredictable.
Generate report data	Specifies whether to generate report data for this transform. This option is available in every transform that generates report data. *Yes:* Generates report data for this transform. *No:* Turns off report data generation. If you do not need to generate reports, set this option to *No*. For example, when you test data flows, disable reports by setting this option to *No*. Setting the option to *No* can improve match performance.
Logical source field	Specifies the field that contains the ID that identifies the logical source.

Option	Description
Physical source field	Specifies the field that contains the ID that identifies the physical source (reader).
Run as a separate process	Specifies whether to run the transform as a separate process.*Yes:* Runs the transform in a process separate from the rest of the data flow. *No:* Keeps the transform in same process as the rest of the data flow.

Related Information

Performance Optimization Guide: Distributing Data Flow Execution, Splitting a data flow into sub data flows

3.5.11.2 Match transform Data Salvage tab

Set options in the *Data Salvage* tab in the Match editor to perform data salvage on your data for more complete matching results.

Open the *Match Editor* by selecting the Match transform in your data flow and then selecting *Tools* > *Match Editor.*

Data Salvage tab option descriptions.

Option	Description
Match set name	Specifies a unique name for this match set. Each Match transform in your data flow represents a match set. If you used the Wizard_MatchBatch transform configuration to generate the transform, the Wizard automatically populates this option with a name. The name must be unique within the data flow. This name is used in the match reports to differentiate data processed by one match set versus another.
Enable data salvage	Specifies to perform data salvaging. If two records match, data salvaging temporarily copies data from a passenger record to the driver record after comparing the two records. The copied data is data that is found in the passenger record but is missing or incomplete in the driver record. Data salvaging prevents blank matching or initials matching from matching records that you may not want to match.
Perform data salvage default	Specifies whether to perform data salvage if the data record does not contain a field with this value. *YES*: Performs the data salvage on the driver record after it matches a passenger record. *NO*: Does not perform the data salvage on the driver record after it matches a passenger record.
Specify data salvage by field	Controls data salvaging by means of a value in a field.

Option	Description
Perform data salvage field	Specifies the field that contains the indicator for performing the data salvage operation. Use this option setting to override the default value.
Specify data salvage by source	Select to control data salvaging per source.
Source	Select a source from the dropdown list. The dropdown list contains sources that you previously added to the *Logical Input Source* dialog box.
Perform data salvage	Specifies whether to perform data salvaging on a source.

Related Information

Data Salvage

3.5.11.3 Match or Associate transforms Input Source options

To select the input sources to track statistics, and to provide additional functionality, set options in the *Input Source* group .

The *Input Source* options appear in both the Associate and the Match transform editors.

Before you define your input sources, map a field that contains the value that identifies the input source. For details, read about input sources in the Match section of the *Designer Guide*.

To access the input sources options, open the *Match Editor*, right-click the *Transform Options* node in the left pane, and select *Add* > *Input Source.* The *Input Sources Editor* appears in the main pane.

Input Source option descriptions

Option	Description
Value field	Specifies an input field to contain the value for your source. **i Note** Map the input field before you define the input source.
Source name	Specifies a name for your input source.
Source value	Specifies a value from the input data that identifies records that belong to the *Source name*. The software creates sources by matching the *Source value* to the *Value field*.

Option	Description
Source type	Specifies a source type from the dropdown list. • *NORMAL*: Contains good or eligible records. • *SUPPRESS*: Contains disqualified records that the software does not use. **Example** For example, if you're using the software to refine a mailing list, a Suppress source removes the following records from the mailing list: • DMA Mail Preference File • American Correctional Association prisons/jails lists • No pandering or non responder lists • Credit card or bad-check suppression lists • *SPECIAL*: Contains good or eligible records that the software uses only to determine whether a match group is single source or multisource. The software does not count the record towards the multisource status. **Example** A special source contains "seed" records. Seed records are for individuals who report back to the mailer when they receive advertising mail. The mailer then uses the information to measure mail delivery. You label the seed source as a special source type so that the software does not count records from the seed source towards the multisource status.
Default source name	Specifies a source to which the software assigns records that do not belong to a predefined source. The dropdown list contains only sources that exist. If the *Source value* is blank or if the job has reached the maximum number of sources (maximum of 10,000), the software uses the *Default source name*.

Option	Description
Auto generate sources	Specifies whether to automatically generate sources for each unique entry in the *Value field*. Auto generating sources saves you time because you do not have to manually define your input sources. The name of an automatically generated source is the same as the value in the *Value field*. As the transform processes each record, the Match transform performs the following checks: • Checks to see if the record belongs to a predefined source. If it does, the transform assigns that record to the predefined source. • If the record does not belong to a predefined source, then the transform checks to see if the record belongs to an auto generated source. • If the record belongs to an auto generated source, the transform uses the automatically generated source. • If the input source is not defined, the transform adds the definition to the list of defined sources. • If the transform reaches the maximum number of source definitions, then instead of adding a new source definition, the transform uses the *Default source name*. **i Note** When the transform is set to *Auto generate sources*, the Source ID is case sensitive but the *Type field* is not case sensitive. The transform accepts the following *Type field* values in the *Auto generate sources*: N, n, P, p, S, or s.
Default type	Specifies a source type when the source does not have a type defined in the *Type field*.
Type field	Specifies an input field that contains the input *Source type* value. The type field must have a value of: *N*: Normal *P*: Suppress (or Purge) *S*: Special If you do not select a field, or if the field value is not N, n, P, p, S, or s, the transform uses the *Default type*.

Related Information

Using sources

3.5.11.4 Match transform Source Group options

Include a Source Group in your Match transform set up to include additional statistics in certain Match reports.

For complete information about creating source groups so that you can access the Source Group options, read about source groups in the Match section of the *Designer Guide*.

Option	Description
Source groups	Specifies source groups. Double-click the checkbox in the first row and type a name for the source group.
Sources	Specifies the sources for the source group. The listed sources are the sources that are defined in the *Input Source* options group.
Undefined action	Specifies what the transform does when an input source does not appear in a source group. *Ignore*: Processes the input source as one that does not belong to any source group. *Default*: Processes the input source as one that belongs to the default source group specified in the *Default source group* option. *Auto*: Processes the input source as follows: • If the *Source group field* option is not defined, then the transform includes the input source in a source group of the same name as the input source. If it does not exist, the transform creates the source group. • If the *Source group field* option is defined, then the transform includes the input source in the source group that you designated in the *Source group field* option. If it does not exist, the transform creates the source group. • If the *Source group field* field is blank, then the transform processes the input source with the *Ignore Undefined action*.
Default source group	Specifies a default source group name. Select a source group from the dropdown list. If you choose *Default* for *Undefined action*, this option is required.
Source group field	Specifies the field that contains the value for your source groups.

Match transform groups [page 622]
To group and prioritize records for better match accuracy and efficiency, create match groups.

Match and User Defined transform Break Group options [page 623]
To define common field values for match break groups, set break group options.

Match and User Defined transform candidate selection options [page 625]
To append records from a relational database to an existing data collection before matching, set options in the Candidate Selection editor.

Match and Associate transforms Priority Order options [page 369]
To control group priority order in post processing, set options in the *Priority Order* group.

Match and Associate transforms Record Completeness [page 370]
To set the priority of output data based on record completeness, configure options in the *Record Completeness* tab.

Related Information

Creating source groups

3.5.11.4.1 Match transform groups

To group and prioritize records for better match accuracy and efficiency, create match groups.

Create break groups

Create break groups that contain records based on common field values such as postal code. The transform compares only the records that share the same break group value.

Use break groups to lower the number of comparisons needed and to increase the efficiency of the matching process.

Candidate selection

Candidate selection appends records from a relational database to an existing break group for processing.

For real-time jobs, candidate selection pulls a candidate set of records based on a single record or many records.

> **i Note**
>
> Candidate selection works with relational databases only; it does not work with flat files.

Group prioritization

Use group prioritization to ensure that your most complete and accurate records drive the comparison process.

Related Information

Match and User Defined transform Break Group options [page 623]
Match and User Defined transform candidate selection options [page 625]
Match and Associate transforms Priority Order options [page 369]

3.5.11.4.2 Match and User Defined transform Break Group options

To define common field values for match break groups, set break group options.

Find the *Break Group* options in the Match Editor or the User-Defined Editor.

Break group option descriptions

Option	Description
Split records into break groups	Specifies that the transform forms break groups to reduce the total number of comparisons made. This option is selected by default. The most common case for not selecting this option is for real-time jobs. If data in a real-time job forms one break group, do not select this option. Instead, for optimal real-time matching, use candidate selection. Candidate selection selects a limited number of records from a relational database. ⚠ Caution Deselect this option with caution within a batch data flow. The size of a break group may not exceed 2 GB. If you use this option in a batch data flow, also set the *Maximum allowable break group size (in records)* option so that the collection does not exceed the size limit. If it does exceed the limit, the software stops the data flow execution. → Tip Calculate break group size by multiplying the record length by the number of records in the break group.
Field	Specifies a mapped input field to include in the break key. For a more complex break key, define that field using an upstream Query transform and select the field in this option.
Add Row	Select to add a new row to the Break key table.
Remove Row	Click to remove a row from the Break key table.
Start Position	Specifies the start position of the field. Valid values for a field length of n are 1 to n and -1 to -n. Negative start values signify that the transform counts the start position from the right. Example For example, a field with a length of 7 contains JOHNSON. A start position of 2 means start with "O." A start position of -4 means start with the first "N." If the field has a length of 20, the software still counts the start position of -4 at the letter "N" because the negative start value starts from the actual length of the string, not at the start of the field.
Length	Specifies the number of characters in the field you want included in the break key.

Option	Description
Break key case sensitive	Specifies whether to treat the break key as case sensitive. • *Yes:* Treat the break key as case sensitive. • *No:* Do not treat the break key as case sensitive. **Example** You create a break key using the Primary_Name (street) field. You set *Break key case sensitive* to *Yes*. Input consists of: • One input record contains Primary_Name = Main • Another input record contains Primary_Name = main The software forms separate break groups with the following input values: 1. Break group 1: Main 2. Break group 2: main
Replace NULL with empty string	Specifies whether to convert NULL values with an empty string in the break key. • *Yes:* Convert NULL to an empty string. • *No:* Do not convert to an empty string.
Right pad fields with blanks	Specify whether to right pad fields with blanks. • *Yes:* Right-pad fields with blank spaces. • *No:* Do not right-pad fields. Because the transform uses the break key for sorting and aggregating, the transform is sensitive to the position in which data is placed. By right-padding the break key fields you can ensure that the software forms break groups properly. If the *Replace NULL with empty string* option is set to *Yes* and this option is set to *Yes*, then the transform replaces fields with NULL values with all spaces to the length of the field.
Input already sorted	Specifies that the input data has already been sorted, and the transform should not sort it again. • *Yes:* The transform does not resort the input data. • *No:* The transform sorts the break keys at runtime before forming break groups. **→ Tip** if you require a more complex break key, use a Query transform to create it, and use the ORDER BY operation to order your data.
Maximum allowable break group size (in records)	Specifies the maximum number of records allowed in a break group. An empty value or zero means that there is no limit on the break group size. Control the amount of memory used during processing by specifying the number of records processed at one time. If more records make it into a single break group than the number that you specify, then the software issues an execution error and stops the job.

Related Information

Break keys and candidate selection

3.5.11.4.3 Match and User Defined transform candidate selection options

To append records from a relational database to an existing data collection before matching, set options in the Candidate Selection editor.

Candidate selection speeds up match processing. The Candidate Selection editor appears in both the Match Editor and the User-Defined Editor.

Candidate Selection Editor option descriptions

Option	Description
Datastore	Specifies the datastore from which you load candidate set. The dropdown list is populated with all valid SQL and persistent cache datastores. For persistent cache datastores, consider using persistent cache when the datastore doesn't often change. In addition to using persistent cache datastores, also use persistent cache datastores created from a flat file. **i Note** Create persistent cache datastores for candidate selection using a data flow in double-byte mode. To change to double-byte mode: 1. Change the locale setting in the Data Services Locale Selector. Set the code page to UTF-8. 2. Run the job to generate the persistent cache. 3. After the job completes, change the code page back to the original setting. If you choose a persistent cache datastore, you cannot use custom SQL.
Cache type	Specifies the type of cache the datastore uses. • *No_Cache:* Specifies that the transform sends each query to the database. No_Cache captures data at a point in time. The data doesn't change until the job restarts. • *Pre_Load_Cache:* Specifies that the entire secondary table is cached to a local disk or memory. Choose this option when the data doesn't often change. Set this option to improve performance. However, a trade-off may be that the cache uses more memory.
Auto-generate SQL	Specifies whether the transform automatically generates the SQL. This option allows you to query a simple single table. If you join tables or create a complex WHERE clause, select the *Create custom SQL* option.

Option	Description
Table	Specifies a table from the datastore.
Use break column from database	Select this option if your database already contains a column that corresponds to the break key field.
Break key field	Select the column from the secondary table that contains the break key field.
Create custom SQL	Select to create custom SQL.
Launch SQL Editor	Opens the SQL editor. This button is only enabled if you select the *Create custom SQL* option.
Use constant source value	Select to assign records to a physical source for generating appropriate statistics.
Physical source value	Type a value for your physical source. This transform places the value in the physical source field you select.
Physical source field	Select the mapped field that contains the physical source name.
Add DB columns to mapping table	If you are using the *Create custom SQL* option, clicking this button adds only the database columns that appear in the SELECT statement and in the order that they appear in the SELECT statement. If you are using the *Auto-generate SQL* option, clicking this button adds ALL database columns, in the order that they appear in the table schema. **i Note** If you do not associate an input field to any of these columns in the column mapping table, the transform removes the columns when you close the window.

Column mapping table

The Column mapping pane allows you to specify which mapped field in the data flow each database selected field is assigned to.

Column	Description
Break key	Specifies whether this field is used as part of your break key.
Field	Each cell contains a list of the mapped names from the input fields in the transform.
DB column	Each cell contains a list of the column names in your database table or the selected columns from a custom query. Match the data of a column in your database to the data of a mapped field.

Related Information

Break keys and candidate selection

3.5.11.4.4 Match and Associate transforms Priority Order options

To control group priority order in post processing, set options in the *Priority Order* group.

The *Priority Order* options appear in both the *Match Editor* and the *Association Editor* based on which transform you are working with.

In the *Post Association Processing* or *Post Matching Processing* dialog of the `<transform_name>` *Editor*, select to add a *Group Prioritization* operation to control group order in post processing.

For more information about priority order, see the *Designer Guide*. Search for "priority order" or "group prioritization".

Group forming prioritization

Use the *Group Prioritization* operation to order records within each break group. This order 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. Ordering records controls 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. A *Group Prioritization* operation controls the destination of data that is being propagated from other records to form a best record.

Group Prioritization Editor option description

Option	Description
Prioritization name	Specifies the name for this Group Prioritization operation. If you have multiple operations in this Match or Associate 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, such as a dollar amount or a date. Use the buttons to add, remove, and order rows. Place the primary sort field at the top of the list. Keep the rest of the fields in the order that they are positioned to determine the sub-sort that occurs.

Option	Description
Input field	Specifies a field on which to sort your records.

Option	Description
Field order	Specifies the order in which records are sorted.

3.5.11.4.5 Match and Associate transforms Record Completeness

To set the priority of output data based on record completeness, configure options in the *Record Completeness* tab.

Record Completeness option descriptions

Option	Description
Prioritization name	Specifies the name for this *Group Prioritization* operation. If you have multiple operations in the Match transform, use a unique name.
Order records based on completeness of data	Specifies whether to apply priority and blank penalty points to records to help control the order of your records.
Define only field penalties	Specifies whether the transform assesses penalties based on blank fields.
Define priority and penalty fields	Indicates that you have specific fields that contain the actual integer values for priority and blank penalty.
Record priority field	Specifies the field that contains priority values. This field must contain an integer.
Apply blank penalty field	Specifies the field that contains the indicator Y or N for applying blank penalty points to a record.
Define priority and penalty based on input source	Specifies whether to have your record priority and blank penalty indicator Y or N determined by membership in a given source.
Source Name	Specifies an input source. Select an input source from the list of sources in the *Source Name* column. The sources listed here are defined in the *Input Source* operation.
Priority	Specifies a priority value (an integer) in the *Priority* column. The lower the priority score, the higher the priority.
Apply Blank Penalty	Specifies whether to apply a blank penalty to a record based on membership to this source.
Default record priority	Specifies the default value for the record priority if: • The record does not contain a field with this value. • The field is blank for a record. • A record does not belong to any of the sources specified. The lower the priority score, the higher the priority.

Option	Description
Default apply blank penalty	Specifies the default indicator to add blank penalty points to records with blank fields. The transform uses the indicator under the following circumstances: • When a record does not have a field that carries this indicator. • If that field is blank or has invalid data. • If a record does not belong to any of the sources specified. Options are: • *Yes*: The transform adds each blank penalty for a record to the record priority to generate an adjusted record priority score. The lower the score, the higher the priority. • *No*: The transform does not apply a penalty when the fields are blank.
Input field	Displays the input fields that are available to assign a blank penalty score to.
Blank penalty	Specifies a penalty value (an integer) to apply to a field when the specified field is blank in a record.

3.5.11.5 Match transform match level options

Set options that effect match processing at the match level.

Match level Person option descriptions

Option	Description
Match level name	Specifies a name for this match level. Use a name that is unique within the Match transform.
Weighted match score	Specifies the weighted match score for this level. When your matching method includes weighted scoring, the transform considers records as matches when the total contribution score is greater than or equal to the value that you enter for *Weighted match score*.
Number of names that must match	Specifies the number of names that must match. This option requires that you have criteria of Person1_Given_Name1, Person1_Family_Name1, and so on. • *One*: Specifies that records are a match when at least one of the names meets the criteria. • *All*: Specifies that records are a match only when all of the names meet the criteria.

Option	Description
Match on hyphenated family name	Specifies whether a single family (last) name in one record matches a hyphenated family name in another record. The transform performs this comparison only when one field has a hyphen and the other field doesn't have a hyphen. • *Yes*: The family names match as long as the single family name in one record matches one of the hyphenated family names in another record. • *No*: The transform considers the hyphenated family name as a single family name and determines the records don't match. The comparison results in a low similarity, which may not meet your family name criteria. **Example** With the *Match on hyphenated family name* set to *Yes*, the transform determines that the following records match: **Given** / **Family** Laura / Smith Laura / Albers-Smith This option works on a criteria named Family_Name1, Family_Name2, or Family_Name3.

Option	Description
Compare Given Name1 to Given Name2	Specifies whether the first name (Given Name1) of one record is compared to the middle name (Given Name2) of another record. • *Yes*: The transform compares the Given_Name1 field of one record to the Given_Name2 field of another record. • *No*: The transform doesn't compare the Given_Name1 field of one record to the Given_Name2 field of another record. **Example** With the *Compare Given Name1 to Given Name2* set to *Yes*, the software determines that the following records could be duplicates: **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.
Ignore family name when female	Specifies whether an adjustment occurs for family names when the given name is a female. • *Yes*: The transform ignores the Family_Name1 criteria when the given name gender is a female (Gender=5). For example, Laura Smith may match Laura Albers. • *No*: The transform doesn't use gender and it performs the matching process as usual. To use this option, you must have at least these three criteria: Given_Name1, Family_Name1, and Gender.

Address Options descriptions

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. • *Yes*: The transform considers records a match if the Boxes match. If the Boxes don't match, then the address and rural route address must pass the match criteria settings. • *No*: The transform requires that all forms of the address (street, rural route, and PO Box) match.
Address matches blank if Firms match	Specifies whether to match on firm data when other address data doesn't match. This option affects only records where one record has street information and the other record has PO Box information. If both records have Street information that don't match, or if both have PO Box information that don't match, the transform doesn't consider the records as duplicates. • *Yes*: If firm data matches and neither firm field is blank, blank matching is allowed for all address components. • *No*: If firm data matches but address data in one of the records is blank, the transform doesn't consider the records a match, unless you enable blank matching for the address components.

Example

firm	number	street	suffix	postal number	post-code
Acme Hard-ware	100	Elm	Ave	200	02961
Acme Hard-ware	123	Main	St	200	02961

Example

firm	number	street	suffix	postal number	postcode
Acme Hardware	100	Elm	Ave		02961
Acme Hardware				300	

Option	Description
Unique on resident if RR, but no Box	Specifies whether to match on Rural Route when an input record 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. • *Yes*: The transform places all records with this type of name data into the same match group. • *No*: The transform doesn't place records with this type of name data into the same match group.
Ignore Firm if Name matches	Specifies whether to consider as matching names at the same address even if the firm names don't match. • *Yes*: Indicates that matching names at the same address are matches, even if the firms don't match. Selecting *Yes* enables the transform to catch the following match, which it may have been missed: Example Rita Terranova Greenco 100 Bren Rd 55343 Rita Terranova Eco Technologies 100 Bren Rd 55343 • *No*: Both the firm criteria and the address criteria must meet the minimum similarity threshold to match.

Related Information

Weighted-scoring method

3.5.11.6 Match transform 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.

Match transform criteria fields [page 634]
Choose fields in the *Criteria Fields* tab to map fields to input fields and set match criteria in other tabs.

Match transform match criteria options [page 637]
Use the *Criteria Fields* tab to set precomparison options and comparison rule options for the criteria field that you choose.

Match transform Multiple Field Comparisons tab [page 650]
Set options for comparing the criterion field against all selected fields in other records or to the same fields in other records.

3.5.11.6.1 Match transform criteria fields

Choose fields in the *Criteria Fields* tab to map fields to input fields and set match criteria in other tabs.

Choose fields in the Criteria Fields tab to map fields to input fields and set match criteria in other tabs.

Category	Fields
Geographic	*ADDRESS_DATA1-5*: Use for address data that is not accounted for in other address-based criteria in the Geographic category. Also use the criteria for fields that you know contain address data, but you're not sure which type it contains, or use the criteria for international data that has not been parsed.
	ADDRESS_POST_OFFICE_BOX: Post Office box number.
	ADDRESS_PRIMARY_NAME: Street name data.
	ADDRESS_PRIMARY_NUMBER: Street number data.
	ADDRESS_PRIMARY_POSTFIX: Address data that comes at the end of a street name, such as a directional.
	ADDRESS_PRIMARY_PREFIX: Address data that comes at the beginning of a street name, such as a directional.
	ADDRESS_PRIMARY_TYPE: Data that tells what type of street it is (street, boulevard, lane, and so on).
	ADDRESS_PRIVATE_MAIL_BOX: A private mail box (PMB) number. These are mail boxes that are not run by a postal authority.
	ADDRESS_RURAL_ROUTE_BOX: Rural-route box number (number only, without "Box" prefix).
	ADDRESS_RURAL_ROUTE_NUMBER: Rural route number.
	ADDRESS_SECONDARY_NUMBER: The number of a unit, building, floor, or room.
	COUNTRY: Country name.
	LOCALITY: City, town, locality, or suburb.
	LATITUDE_LONGITUDE: Latitude and longitude.
	POSTCODE1: Primary postal code.
	POSTCODE2: Secondary postal code.
	REGION: Region data, such as state or province.
Firm	*FIRM:*: Firm name.
	FIRM_DATA1-3: Use for firm data that is not accounted for in other firm-based criteria. Also use the criteria for fields that you know contain firm data, but you're not sure which type it contains. Also use the criteria for international data.

Category	Fields
	FIRM_MATCH_STD1-6: Firm match standards. The data in these fields is generated by the Data Cleanse transform or other pre match transforms.
	FIRM_LOCATION: A location within a company or organization.
	FIRM_LOCATION_MATCH_STD1-6: Match standards for a location within a company or organization.
Person	*NAME_DATA1-3*: Use for name data that is not accounted for in other name-based criteria. Also use the criteria for fields that you know contain name data, but you're not sure which type it contains.
	PERSON1-3_GIVEN_NAME1: The given name1 (first name) of the persons.
	PERSON1_GIVEN_NAME1_MATCH_STD1-6: Given_Name1 (first name) match standards for the first person.
	PERSON2_GIVEN_NAME1_MATCH_STD1-6: Given_Name1 (first name) match standards for the second person.
	PERSON3_GIVEN_NAME1_MATCH_STD1-6: Given _Name1 (first name) match standards for the third person.
	PERSON1-3_GENDER: Gender.
	PERSON1-3_FAMILY_NAME1 Family (last) name.
	PERSON1-3_FAMILY_NAME1_MATCH_STD1-6: Family_Name1 match standards for the persons in your data record.
	PERSON1-3_FAMILY_NAME2: Family (last) name. Use this field when family name data is split into two fields. For example, some cultures store the paternal family name and the maternal family name in different fields.
	PERSON1-3_FAMILY_NAME2_MATCH_STD1-6: Family_Name2 match standards for the persons in your data record.
	PERSON1-3_MATURITY_POSTNAME: Maturity postname. For example, Sr. or Jr. (one standard per person).
	PERSON1-3_MATURITY_POSTNAME_MATCH_STD1-6: Maturity postname match standards for the persons in your data record.
	PERSON1-3_GIVEN_NAME2: Given name2 (middle name).
	PERSON1_GIVEN_NAME2_MATCH_STD1-6: Given_Name2 (middle name) match standards for the first person.
	PERSON2_GIVEN_NAME2_MATCH_STD1-6: Given_Name2 (middle name) match standards for the second person.

Category	Fields
	PERSON3_GIVEN_NAME2_MATCH_STD1-6: Given_Name2 (middle name) match standards for the third person.
	PERSON1-3_HONORARY_POSTNAME: Honorary postname for up to three persons indicating certification, academic degree, or affiliation. For example, CPA.
	PERSON1-3_HONORARY_POSTNAME_MATCH_STD1-6: Honorary postname match standards.
	PERSON1-3_PRENAME: Prename (for example, Mr. or Mrs.) for up to three persons.
	PERSON1-3_PRENAME_MATCH_STD1-6: Prename match standards.
	PERSON1-3_TITLE: Job or occupational title of each person. For example, Manager.
	PERSON1-3_TITLE_MATCH_STD1-6: Title match standards for each person.
	SOCIAL_SECURITY_NUMBER1-3: Social Security numbers for up to three people in a record.
Other	*DATE1-3*: Date data. For example, birthdate data.
	PHONE: Phone number.
Custom	Use custom fields to match data that does not qualify for any of the specifically named criteria. If you prepared fields for matching through a custom Universal Data Cleanse solution, the Data Cleanse dictionary names appear in the Custom category.

Criteria field mapping

After you select a field from the *Available criteria* pane, match the field to a mapped input field in the *Criteria field mapping* pane.

Option	Description
Criteria field	Contains the criteria field that you have selected in the *Available criteria* pane.

Option	Description
Input field mapped name	Specifies the input field mapped name for the criteria field. The dropdown list contains only existing input fields of the correct type. If you choose any of the following input fields, the Match transform automatically adds the appropriate match standard fields to the list of criteria fields. The fields that the software displays vary depending on the value that you choose for *Compare data using* on the *Options* tab. • Firm • Firm_Location • Person*_Given_Name1 • Person*_Given_Name2 • Person*_Honorary_Postname • Person*_Maturity_Postname • Person*_Prename • Person*_Title Choose to include or exclude any of these from mapping. **i Note** If you enable multiple field matching in the *Multiple Field Comparisons* tab, the software removes any appropriate match standard fields. If you want to include them in the match process, add them in the *Additional fields to match* table in the *Multiple field matching* section.

3.5.11.6.2 Match transform match criteria options

Use the *Criteria Fields* tab to set precomparison options and comparison rule options for the criteria field that you choose.

The options that you set in the *Options* tab apply to the node that you select under *Match Criteria* in the Explorer pane.

There are four sections in the *Options* tab. The following tables describe the options in each section.

Compare data using

Choose a comparison method from the *Compare data using* list. Your selection specifies how Match compares fields, when more than one word commonly exists in the field. Your selection determines the options available in the Comparison rules section of the Options tab.

The Match transform refers to the comparison rules as algorithms. The transform warns you if you choose a comparison that isn't compatible with the chosen criteria field.

Compare data using

Option	Description
Field Similarity	Compares all data in the field as a single string. This algorithm is more efficient when you use it for fields that typically have one word.
Geo Proximity	Compares latitude and longitude information from different records for geographic proximity. Determines that the records are close enough to be considered duplicates.
Numeric Difference	Compares numeric information from different records based on numerical difference. Determines that the records are close enough to be considered duplicates.
Numeric Percent Difference	Compares numeric information from different records based on percentage of numerical difference. Determines that the records are close enough to be considered duplicates.
Word Similarity	Parses the data in the field into words, then compares the words. *Word Similarity* is less efficient than *Field Similarity*. However, *Word Similarity* does a better job comparing data with more than one word. Many criteria options require that you set the comparison to *Word Similarity*.

Pre-comparison options

Use the options in the *Pre-comparison options* section to alter the data in the criteria field for better comparison. These options don't alter the data that is output from the Match transform.

Pre-Comparison option descriptions

Option	Description
Field compare length	Specifies the number of characters in the field to compare. **Note** If the input field mapped to the main criteria (not a match standard) isn't a varchar data type, *Field compare length* isn't editable.
Remove punctuation	Specifies whether to remove punctuation from your data to help provide more accurate matches. Select *Yes* to remove punctuation. **Limitations:** • Valid only for Latin1 data. • Match doesn't remove a dash when it appears in a value in the `<Family_Name*>` field. **Note** If you use the same mapped input field for other criteria, changes you make here affect all other criteria related to the mapped input field. Therefore, before you set this option, check that there are no other criteria using the same mapped input field. **Caution** Setting this option and the *Convert text to numbers* option to *Yes* can produce undesirable results. **Example** The data criteria field contains `1.23` as a value. If you set *Remove punctuation* to *Yes* the transform converts `1.23` to `123`. With this change, the number `123` can then match another value of `123`. If you also select *Yes* for *Convert text to numbers*, `123` matches `one hundred twenty-three`.

Option	Description
Convert to uppercase	Specifies whether to convert all data to uppercase for matching purposes only. This option is valid only for Latin1 data. • *Yes:* Converts the data to uppercase for the comparison process. • *No:* Preserves the case of the data for the comparison process, and compares “A” to “a” as different values. **Note** If you use the same mapped input field for other criteria, changes you make here affect all other criteria related to the mapped input field. Therefore, before you set this option, check that there are no other criteria using the same mapped input field.
Convert diacritical characters	Specifies whether to include diacritical characters in the matching process. • *Yes:* Converts diacritical characters to the closest English ASCII equivalent for matching purposes. For example, “ä” converts to “a”. • *No:* Preserves diacritical characters in the matching process, treating “ä” and “a” as different characters. **Additional information:** • Valid for all match engine options. • Works only on upper Latin1 characters, values between 128 and 255. **Example** If you process Japanese data, you might have some Latin1 characters mixed in. Setting this option to *Yes* converts any diacritical characters in the Japanese data. In these cases, you can convert diacritical characters. **Note** If you use the same mapped input field for other criteria, changes you make here affect all other criteria related to the mapped input field. Therefore, before you set this option, check that there are no other criteria using the same mapped input field.

Option	Description
Convert text to numbers	Specifies whether numbers represented as text, such as `one`, `two`, `three`, are converted to numbers. *Yes:* Converts numbers represented as text to numbers. • Converts in cardinal format, such as `one` = `1`. • Converts in ordinal format, such as `first` = `1st`. *No:* Leaves any numerical text intact. **i Note** If you use the same mapped input field for other criteria, changes you make here affect all other criteria related to the mapped input field. Therefore, before you set this option, check that there are no other criteria using the same mapped input field. **⚠ Caution** Setting this option and the *Convert text to numbers* option to *Yes* can produce undesirable results. **Example** The data criteria field contains `1.23` as a value. If you set *Remove punctuation* to *Yes* the transform converts `1.23` to `123`. With this change, the number `123` can then match another value of `123`. If you also select *Yes* for *Convert text to numbers*, `123` matches `one hundred twenty-three`.
Locale	Specifies the locale setting for this criterion field. If nothing is specified, or you specify *DEFAULT*, the transform uses the default system locale. Best practice is to set this option when you also set the *Convert text to numbers* option to *Yes*.

Comparison rules

Comparison rules option descriptions

Option	Description
Approx substring adjustment score	Specifies the score for words that aren't matched to the other words in a compared string. Enter a value from 0 (default) to 100. Enter a value of 0 to disable the option. Use this option to compare family names that have varying representations for which the *Substring adjustment score* option is too strict to compare. For example, "Cruz Rodríguez" and "C. de Rodríguez".
Abbreviation adjustment score	Controls matching whole words to abbreviations. For example, long firm names are abbreviated by removing letters. International Health Providers might be abbreviated to Intl Health Providers. • Enter a value of 0 (zero) to disable abbreviation checking. • Enter a value greater than 0 to enable this option. • Enter a value of 100 if you want the transform to consider abbreviations and longer words as a perfect match. The transform considers abbreviation as: • The first letter of the shorter word matches the first letter of the longer word. • 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. **Example** Full word: Business — Possible abbreviations: Bus, Bsnss, Bss Full word: Database — Possible abbreviations: Dat, Db, Dse **Note** For this option to work, set the *Compare data using* option to *Word Similarity*.

Option	Description
Both fields blank operation	Specifies whether to use the criteria field when both of the fields in a record for the criteria field are blank. • *Eval*: The transform uses the value that you enter in the *Both fields blank score* option as the similarity score for the criteria field. • *Ignore*: The transform ignores the criteria field in the comparison process. The transform proportionally distributes this criterion contribution to the weighted score among the remaining criteria, therefore negating any impact the contribution score has.
Both fields blank score	Specifies the similarity score if both of the fields are blank and *Both fields blank operation* is set to *Eval*. Enter a value from 0 to 100.
Check for transposed letters	Specifies whether the transform adjusts the match score for any transposed characters encountered. • *Yes*: The transform deducts half as many points for transposed characters as it deducts for other non-matching characters. Example **Comparison:** Smith—Simth **Finding:** Words differ by one transposition, penalty of 1 correction. **Percentage alike:** 90% • *No:* The transform handles transposed characters the same way it handles any non-matching characters. Example **Comparison:** Smith—Simth **Finding:** Words differ by one transposition, penalty of 1 correction. **Percentage alike:** 80%
Contribution to weighted score	Specifies the contribution value, when you use the weighted or the combination scoring method. Type a value between 0 and 100. If no single criteria decides a match or no-match, the transform calculates the contribution score by summing the products of each criteria's score by each criteria's weight.

Option	Description
Distance Unit	Specifies the type of distance unit used to calculate the distance between two Latitude, Longitude pairs. This option is only available when the *Geo Proximity* option is selected. Select one of the following: • *Feet* • *Kilometers* • *Meters* • *Miles*
Enable inter-script matching	Enable this option if you have the same data in different scripts (writing systems). For example, one record contains Latin1 and the other contains Katakana. One record contains Latin and the other contains Cyrillic.
Ext abbreviation adjustment score	Specifies an adjustment to the similarity score for variations of the *Abbreviation adjustment score* option. Enter a number that adjusts the similarity score for these types of abbreviations. **Example** For example: • Enter a value of 0 (default) to disable abbreviation checking. • Enter a value greater than 0 to enable abbreviation checking. The following rules apply when you use this option: • The first letter of the short word matches the first letter of the first word in a multiple word string • The remaining letters of the short word are in the same order in the multiple word string. • The transform assigns a score of 100 to letters that match. The transform assigns the score that you specify to the remaining letters. • The transform proportionally combines the scores to render the overall score. **Note** For this option to work, set the *Compare data using* option to *Word Similarity*.

Option	Description
Initials adjustment score	Specifies whether you want initials or acronyms to match whole words. For example, the firm name International Health Providers could match IHP. Enter a value from 0 to 100. Enter a value of 0 (default) to disable initial checking. Remember the following rules when using this option: • The initial must match the first letter of the word. • The transform assigns the score of 100 to letters that match. The transform assigns the score that you specify to the remaining letters (1–100). • The two scores are proportionally combined to render the overall score. If there are other words in the field that aren't shortened, they are scored the usual way. For example, New York Police Department can be shortened to New York PD and still match. **i Note** For this option to work for multiple-word abbreviations, set the *Compare data using* option to *Word Similarity*. For this option to work for single-word abbreviations such as Maria = M, set the *Compare data using* option to either *Word Similarity* or *Field Similarity*.
Match score	Specifies the minimum similarity score needed for the records to be considered a match based on this criterion. Type a value from 0 to 101. A value of 101 ensures that this criterion alone isn't enough to consider two records a match and that you want to consider other criteria in the comparison process. **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.

Option	Description
No match score	Specifies the maximum similarity score needed for the records to be considered not a match based on this criterion. Type a value from -1 to 100. A value of -1 ensures that this criterion isn't enough to consider two records a no-match and that you want to consider other criteria in the comparison process. **Example** A value of 49 means that if the similarity between the data in two records is less than 50%, the records don't match.
Max Difference	Specifies the maximum difference allowed in a numeric range. Type a value 0–2147483647.
Max Difference Score	Specifies what score to generate when the difference is the same as the *Max Difference*. Valid values for this required attribute range from 0 to 100. • Any difference larger than the *Max Difference* receives a score of 0. • A difference equal to *Max Difference* receives a score of *Max Difference Score*. • Any difference less than *Max Difference* receives a proportional score between *Max Difference Score* and 100.
Max Distance	Specifies the maximum distance allowed when calculating the distance between two Latitude, Longitude pairs. Type a value from 0 to 4294967295.0.
Max Distance Score	Specifies what score to generate when the distance is the same as *Max Distance*. Type a value from 0 to 100. • Any distance larger than the *Max Distance* receives a score of 0. • A distance equal to *Max Distance* receives a score of *Max Distance Score*. • Any distance less than *Max Distance* receives a proportional score between *Max Distance Score* and 100.
Max Percent Difference	Specifies the maximum difference allowed as a percent of the absolute value. Type a value from 0 to 100.

Option	Description
Max Percent Difference Score	Specifies what score to generate when the difference is the same as *Max Percent Difference*. Valid values for this required attribute range from 0 to 100. • Any difference larger than the *Max Percent Difference* receives a score of 0. • A difference equal to *Max Percent Difference* receives a score of *Max Percent Difference Score*. • Any difference less than *Max Percent Difference* receives a proportional score between *Max Percent Difference Score* and 100.
Numeric words match exactly	Specifies how to match data that contains both numbers and letters. • *None*: Numeric words aren't required to match exactly to be considered a match. • *Any_Position*: Numeric words aren't required to be in the same position in two different strings to be considered a match. • *Any_Position_Consider_Punctuation*: Works the same as the *Any_Position* value. However, the transform takes the position of a decimal separator such as a comma or a period within the numeric words into consideration. • *Any_Position_Ignore_Punctuation*: Works the same as the *Any_Position* value. However, the transform ignores decimal separators such as comma or period. • *Same_Position*: Numeric words must match exactly and be in the same position in the string to be considered a match. **i Note** For this option to work, set the *Compare data using* option to *Word Similarity*.
One field blank operation	Specifies whether to use this criterion if a field in one record is populated and the field in a different record is blank. • *Eval*: The transform uses the value in the *One field blank score* option as the similarity score for this criterion. • *Ignore*: The transform ignores this criterion in the comparison process, and the transform distributes the contribution to the weighted score proportionally among the remaining criteria.
One field blank score	Specifies the similarity score to use if one of the fields is blank and the *One field blank operation* option is set to *Eval*. Type a value from 0 to 100.

Option	Description
Substring adjustment score	Allows matching longer strings of words to shorter strings. **Example** Long firm names are sometimes 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. Before you set this option, keep in mind the following information: • Enter a value of 0 (default) to disable substring checking. • Enter a value of 100 if you want substrings and longer strings to be considered a match. After processing, the transform assigns scores as follows: • Assigns letters that match a score of 100. • Assigns the remaining letters the score you specify (1–100). • Combines the two scores proportionally to render the overall score. To qualify as a substring match, the shorter string must exactly match the first part of the longer string. **Example** Matching substrings: Mayfield Mayfield Painting Mayfield Painting and Sand Substrings that don't match: Mayfield Sand Blasting Painting and Sand Blasting Alternate spellings in any of the words also disqualify the substrings as a match. For example, the values "Murphy Painting and Sand Blasting" don't match. The comparison has a similarity score of 85% without setting the *Substring adjustment score*. **Note** For this option to work, set the *Compare data using* option to *Word Similarity*.

Option	Description
Use in weighted score if greater than	Specifies the minimum similarity score required to qualify this criterion to contribute to the *Weighted Match Score*. **Example** If you set this option to 59 for a given name criterion, and the given names between two records are less than 60% similar, then the transform ignores the given name criteria and it distributes the contribution value proportionally among the remaining criteria.
Zero weighted score if less or equal	Specifies the minimum similarity score needed for this criterion to qualify for contributing a value other than zero to the weighted match score. **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.

Set Scores buttons

We provide you with default matching scores for every criterion in the *Comparison rules* options. The Set Scores buttons allow you to try different base levels of matching 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.

Set Scores button descriptions

Button	Descriptions
Set Scores to Exact	Data must be exactly the same to be considered a match. The transform doesn't allow any variation. There are fewer matches in a match group but the quality of the matches is high.
Set Scores to Tight	Data must have a high level of similarity. The transform allows a small amount of variation.
Set Scores to Medium	Data can have a medium level of similarity. The transform allows a medium amount of variation.

Button	Descriptions
Set Scores to Loose	Data can have a lower level of similarity. The transform allows a greater amount of variation. There are more matches in a match group, but the quality of the matches is poor.

Related Information

Numeric data matching
Extended abbreviation matching
Matching methods
Unicode matching

3.5.11.6.3 Match transform Multiple Field Comparisons tab

Set options for comparing the criterion field against all selected fields in other records or to the same fields in other records.

Multiple field mapping

The following table describes the options at the top of the *Multiple Field Comparisons* tab.

Multiple Field Comarisons tab option descriptions

Option	Description
Compare multiple fields	Enables multiple field matching. When you enable this option, the remaing options are editable.
All selected fields in other records	Specifies to match each field against all fields selected in the table in each record.
The same field in other records	Specifies to match each field only against the same field in each record.

Additional fields to compare

The following table contains descriptions for the columns in the *Additional fiels to compare* table.

Additional fields to compare table option descriptions

Option	Description
Criteria field	Specifies an input field that contains the data to compare.
Custom name	Specifies the name for the custom field.
Input field mapped name	Assigns the existing mapped input field name for the *Criteria field*. The transform displays fields based on the value set in the *Compare data using* option in the Options tab.

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.

Pre-Comparison option descriptions

Option	Description
Field compare length	Specifies the number of characters in the field to compare. **i Note** If the input field mapped to the main criteria (not a match standard) is not of type varchar, this option is disabled.

Option	Description
Remove punctuation	Specifies whether to remove punctuation from your data to help provide more accurate matches. • *Yes:* Removes punctuation. • *No:* Keeps punctuation in your data Before you set this option, be aware of the following information: • This option is valid only for Latin1 data. • Match does not remove a dash when it appears in a value in a `<Family_Name*>` field. **i Note** If you use the same mapped input field for other criteria, changes you make here affect all other criteria related to the mapped input field. Therefore, before you set this option, check that there are no other criteria using the same mapped input field. **⚠ Caution** Setting this option and the *Convert text to numbers* option to *Yes* may produce undesirable results. **Example** Suppose that you have `1.23` as data in a criteria field. If you set *Remove punctuation* to *Yes* the transform converts this number to `123`. The number `123` could then match another value of `123`, or, in the case of enabling *Convert text to numbers*, this number could match a value of "one hundred twenty-three."
Convert to uppercase	Specifies whether to convert all data to uppercase for matching purposes only. This option is valid only for Latin1 data. • *Yes:* Converts the data to uppercase for the comparison process. • *No:* Preserves the case of the data for the comparison process, and compares “A” to “a” as different values. **i Note** If you use the same mapped input field for other criteria, changes you make here affect all other criteria related to the mapped input field. Therefore, before you set this option, check that there are no other criteria using the same mapped input field.

Option	Description
Convert diacritical characters	Specifies whether to include diacritical characters in the matching process. • *Yes:* Converts diacritical characters to the closest English ASCII equivalent for matching purposes. For example, “ä” converts to “a”. • *No:* Preserves diacritical characters in the matching process, treating “ä” and “a” as different characters. When you set this option, be aware of the following information: • This option is valid for all match engine options. • This option works only on upper Latin1 characters, values between 128 and 255. **Example** For example, if you process Japanese data, you may have some Latin1 characters mixed in. Enabling this option converts any diacritical characters in the Japanese data. In these cases, you can convert diacritical characters. **Note** If you use the same mapped input field for other criteria, changes you make here affect all other criteria related to the mapped input field. Therefore, before you set this option, check that there are no other criteria using the same mapped input field.
Convert text to numbers	Specifies whether numbers represented as text, such as one, two, three, are converted to numbers. If you choose Yes, numbers are in cardinal (one = 1) or ordinal (first = 1st) format. *Yes:* Converts numbers represented as text to numbers. The transform converts in cardinal format such as one = 1, or ordinal format such as first = 1st. *No:* Leaves any numerical text intact. **Note** If you use the same mapped input field for other criteria, changes you make here affect all other criteria related to the mapped input field. Therefore, before you set this option, check that there are no other criteria using the same mapped input field. **Caution** Setting this option and the *Remove punctuation* option to *Yes* may produce undesirable results. **Example** Suppose that you have `1.23` as data in a criteria field. If you set *Remove punctuation* to *Yes* the transform converts this number to `123`. The number `123` could then match another value of `123`, or, in the case of enabling *Convert text to numbers*, this number could match a value of "one hundred twenty-three."

Option	Description
Locale	Specifies the locale setting for this criterion field. If nothing is specified, or you specify *DEFAULT*, the transform uses the default system locale. Best practice is to set this option when you also set the *Convert text to numbers* option to *Yes*.

3.5.11.7 Match transform compare table

Using compare tables enables you to compare records within sources, compare records across sources, or a combination of both.

The transform uses the compare table to determine which record pairs qualify to be compared, or which sources should be compared. If you do not include a Compare table in the Match transform, the transform compares a driver record to all remaining passenger records in a 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, 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 in the Compare actions table for both source A and source B. Set the *Default action* options for those sources to *No_Match*.

Option	Description
Default action	Specifies the action assigned to each cell of the compare table before any *Compare actions* table rows are applied. *Compare:* Compares all of the logical sources specified in the compare table. You can then specify a pair that the transform should not compare to the *Compare actions* table. *No_Match:* Does not compare all logical sources in the compare table. You can then specify a pair to be compared in the Compare actions table.
Default source	Specifies the source that a record belongs to if that record has no field to identify it. If there are no pre defined sources, type a field name.
Define compare actions using field values	Specifies whether to use values in a field as the criteria for comparison, rather than membership in an input source. Enter field values in the *Compare Actions* table to enable this option. If you do not have any input sources defined, this option is selected by default.
Logical source field	Specifies the field that contains the logical source value (name).

Option	Description
Default logical source value	If you are not passing in a field from another location that contains the logical source value, or there is no value in the field, then specify the default value for this option. This option specifies which value defined in the *Compare actions* table is the default value.

Compare actions table

Use this table to set the action of driver records and the records they are compared with (passenger records). These entries can override the value in the *Default action* option.

Compare actions table column descriptions

Option	Description
Driver value	Specifies the source contained in the driver record. If you leave this blank, the transform assumes that all driver records are the first in a break group.
Passenger value	Specifies the value contained in the passenger record. If you leave this blank, the transform assumes that all passenger records are any record that is not the first in the break group.
Action	Specifies the action to take when a record in the *Driver value* column is to be compared with a record from the *Passenger value* column. • *Compare*: Compares the two records. *No_Match*: Does not compare the two records. This option overrides the *Default action* option. For example, if you set the *Default action* option to *No_Match*, specify that you want this pair to be compared by setting *Action* to *Compare*.

Related Information

Compare tables

3.5.11.8 Match and Associate transforms Post Match Processing

Add operations to the *Post Match Processing* group or the *Post Association Processing* group such as generating statistics and creating a best record.

Both the *Match Editor* and the *Associate Editor* contain post processing options. Open the editor by selecting the transform in your workspace and selecting *Tools* > *Match Editor* or *Associate Editor* as applicable.

To add operations to post matching:

1. Right click the *Post Match Processing* node.
2. Click *Add* to open the dropdown list.
3. Select the applicable operation.

Post Match Processing option descriptions

Operation	Description
Group Prioritization	Orders records for processing by other post match operations.

Group Statistics	Generates statistical information about your group of matching records.
	
	Group statistics are essential for generating data for match reports.
	
	Generated statistics include the following information:
	
	• Number of records within the match group.
	• Sequential group order number.
	• Group rank:
	  • Group rank flags one record in each group of matching records as the Master record.
	  • Group rank groups all other records in the group as Subordinate records.
	• Records in a match group belong to more than one source.

Best Record	Salvages data from matching records that are members of match groups. Consolidates or posts salvaged data to a best record or to all matching records.
Unique ID	Assigns sequential identification numbers to each new record of a source, starting with the highest unique ID assigned to the previous source plus 1. **Example** The highest unique ID from Source_A is 2999. The transform starts assigning unique ID numbers for Source_B beginning with 2999 plus 1. The transform continues assigning unique ID numbers sequentially for the remaining records in Source_B.

Match and Associate transforms Best Record tab [page 657]
Use the best record post-match processing operation to specify how the software updates your records with information from other records in a match group.

Match and Associate transforms Destination Protection tab [page 366]
Use the options in the *Destination Protection* tab to protect data in particular records or data in records from particular input sources from being overwritten.

Match and Associate transforms Group Statistics Editor [page 661]
: Use the options in the *Group Statistics Editor* to set up an operation for post association or post match processing.

Match and Associate transforms Unique ID options [page 371]
: Use the Unique ID options to assign unique sequential identification numbers to each new record that you add to a data warehouse.

Match and Associate transforms Destination Protection tab [page 374]
: Use the *Destination Protection* tab to control whether a unique ID for a record is protected based on the source that the record belongs to.

Match transform Output Flag Selection options [page 668]
: Add an *Output Flag Selection* operation to match levels in Post Match Processing to flag specific record types for evaluation or routing downstream in your data flow.

Related Information

Match and Associate transforms Best Record tab [page 657]
Match and Associate transforms Group Statistics Editor [page 367]
Match transform groups [page 622]
Match transform Output Flag Selection options [page 668]
Best record
Unique ID

3.5.11.8.1 Match and Associate transforms Best Record tab

Use the best record post-match processing operation to specify how the software updates your records with information from other records in a match group.

The Best Record options appear in both the *Match Editor* and the *Association Editor* based on which transform you are working with.

Best Record post processing operation option descriptions

Option	Description
Best record name	Specifies a unique name for the Best Record operation. Make sure that you use a name that is unique within this Match transform. **i Note** This is the same option as the *Best record name* option in the *Destination Protection* tab. When you enter a name here, the software populates the *Best Record name* field in the *Destination Protection* tab.

Option	Description
Best record strategy	Specifies the strategy to determine whether any action is taken on records in a match group. After you choose the strategy, priority, and field that you want to work with, the transform automatically generates the Python code for you, except when you choose *Custom*. • *Custom:* Bases the best record strategy on Python code that you create. *Custom* enables you to open the Python Expression editor and create custom Python code. • *Date:* Bases the best record strategy on a date field. • *Length:* Bases the best record strategy on the length of data in a specified field. • *Non_Blank:* Bases the best record strategy on the completeness of data in a specified field. • *Priority_Number:* Bases the best record strategy on a number. • *Priority_String:* Bases the best record strategy on string data in a specified field.
Strategy priority	Specifies priorities for the best record strategy. *Strategy priority* does not control priority for the *Non_Blank* and *Custom* best record strategies. **Date** • *Newest*: The newest date in the field causes an action to take place. • *Oldest*: The oldest date in a field causes an action to take place. **Length** • *Longest:* The longest string in a field causes an action to take place. • *Shortest:* The shortest string in a field causes an action to take place. **Priority_Number** • *Highest:* The highest number in a field causes an action to take place. • *Lowest:* The lowest number in a field causes an action to take place. **Priority_String** • *Ascending:* The string with the most ascending string order causes an action to take place. • *Descending:* The string with the most descending string order causes an action to take place.
Strategy field	Specifies a field that contains data required to execute your strategy.
Posting destination	Specifies the destination record. • *Master*: Posts only to a master record. • *Subs*: Posts only to subordinate records. • *Master to Subs*: Pushes information from the master record and posts it to each subordinate record. • *All*: Posts to both the master and subordinate records.

Option	Description
Post only once per destination	Specifies to post once per destination. • *Yes:* Posts once per destination record. After data is posted to the destination record, the transform stops the operation. Select *Yes* when you use the *Non_Blank* best record strategy. • *No:* Posts more than once per destination. After data is posted to the destination record, the transform continues the operation and populates the destination record again with the next value. Select this option when accumulating values such as total sales. Select *No* when you use the Longest, Shortest, Newest, Oldest, Ascending, or Descending priorities. **i Note** The transform ignores the setting in this option when it uses the *Master to Subs* posting destination. With this posting destination, information can be posted only once.
View/Edit Python	Opens the Python Expression editor. Select the *Custom Best record strategy* to create custom python. If you choose any other strategy, the Python in the editor is read only.

Best record action fields table

Use the *Best record action fields* table to define actions on fields based on your selected best record strategy.

Best record action fields option descriptions

Option/Option group	Description
Source field	Specifies the name of the source field in the input record.
Destination field	Specifies the name of the destination field in the output record, or the destination of this best record action. Either set the action to post to the same input field or set the action to post to a different field.
Custom	Specifies whether you create custom Python code for the action to perform on the destination field. • *Yes:* Specifies to create custom Python code to perform an action on the destination field. • *No:* Specifies to use the same source and destination fields. When this option is set to *No*, the transform copies the contents of the source field to the destination field.
Editor	When you choose *Yes* in the *Custom* column, the word "PYTHON" appears in the *Editor* column and the software activates the *Edit Python* button.
Edit Python	Opens the *Python Editor for Best Record* dialog box. Compose your custom Python code using the options and tools in the editor.

Related Information

Best record

3.5.11.8.2 Match and Associate transforms Destination Protection tab

Use the options in the *Destination Protection* tab to protect data in particular records or data in records from particular input sources from being overwritten.

Destination Protection tab option descriptions

Option	Description
Best record name	Specifies a name for the best record operation. Make sure that you use a name that is unique within this Match transform. **i Note** This is the same option as the *Best record name* option in the *Best Record* tab. If you entered a name for *Best record name* in the *Best Records* tab, the name also appears here. If you change the name here, the software changes the name shown in the *Best Record* tab.
Enable destination protection	Enables the transform to protect records from best record operations that may overwrite the record contents or data in records from particular input sources.
Default destination protection	When you set destination protection for records based on fields: Specifies whether a destination is protected when the field that you specify in the *Destination protection field* does not have a valid value. • *Yes*: Protects a destination when the destination protection field does not have a valid value of Y or N. • *No*: Disables this type of protection. When you set destination protection for records based on input source membership: Specifies whether a destination (input source) is protected when you do not specifically define the source in the table.
Specify destination protection by field	Specifies to enable destination protection through a value in a field. The value must be Y or N. When you select this option, select a field from the *Destination Protection field* dropdown list, or *Unique ID protected field*.
Destination protection field	Specifies the field that is protected. The field must contain a value of Y or N. Any other value, including blank, causes the transform to use the *Default destination protection* value.
Specify destination protection by source	Specifies to enable destination protection through membership in a particular source. When you select this option, complete the table with source names and destination protection settings.

Option	Description
Source Name	Choose the name of the source from the dropdown list. The transform populates the list with defined sources and source groups from the *Input Sources Editor* window of the Match Editor. If there are no names listed in the dropdown list, or you do not see a source that you expected to see, add an input source through the *Input Sources Editor*. To access the *Input Sources Editor*: 1. Click the *Transform Options* node. 2. Click *Add* at the bottom of the pane at left. 3. Click *Input Sources*.
Destination protected	Specifies whether to protect records from the source in the *Source Name* column. • *Yes*: Protects records from the source from being overwritten. • *No*: Does not protect records from the source from being overwritten. If you do not specify an option for every source, the transform uses the default value in the *Default Destination Protection* option

Related Information

3.5.11.8.3 Match and Associate transforms Group Statistics Editor

Use the options in the *Group Statistics Editor* to set up an operation for post association or post match processing.

Create group statistics in the post match processing portion of a match or associate level. Open the *Group Statistics Editor* in either the Match Editor or the Associate Editor. For instructions to create group statistics, see the *Designer Guide*.

Group Statistics Editor option descriptions

Option	Description
Group statistics name	Specifies a name for the group statistics. Make sure that you use a name that is unique within this Match transform.
Generate only basic statistics	Generates statistics about match groups, such as how many records in each group. Basic statistics don't include statistics about input sources.
Generate source statistics from input sources	Generates basic statistics and statistics about input sources. To enable this option, define one or more input sources in the *Match Editor*.

Option	Description
Generate source statistics from source values	Generates source statistics based on source values in a field instead of defined input sources. When you select this option, the following options become editable: • *Logical source field* • *Default logical source value* • *Count all sources* • *Choose sources to count*
Logical source field	Specifies the input field that holds the value for your logical sources. **i Note** Only mapped input fields appear in the list.
Default logical source value	Specifies a value to use if the field specified for the *Logical source field* option is blank. If a record has a blank value in the logical source field, the transform uses the value from the *Default logical source value* option.
Count all sources	Counts all sources, regardless of the value in the field specified in *Logical source field*.
Choose sources to count	Specifies the sources to count based on values in the field specified in *Logical source field*. When you select this option, the following options become editable: • *Default count flag* • *Auto generate sources* • *Predefined count flag field* (editable only when you also select *Auto generate sources*) • *Manually define logical source count flags*
Default count flag	Specifies the value to use when the field specified in *Predefined count flag field* is invalid. For example, if the value in the field specified for *Predefined count flag field* isn't Y, N, or blank. • *YES*: Counts all of the records in the source. • *NO*: Doesn't count any of the records in the source.
Auto generate sources	Generates sources based on the value in the field specified for *Predefined count flag field*.

Option	Description
Predefined count flag field	Specifies the field that contains the Y or N indicator value. The transform behaves as follows when the specified field contains either a Y or an N: • If the specified field contains a Y, the transform counts the source for statistics. • If the specified field contains an N, the transform doesn't count the source. You must check *Auto generate sources* to enable this field.
Manually define logical source count flags	Complete the columns in this table when you select *Choose sources to count*. If you don't complete both columns, the transform disregards the selection of *Choose sources to count*.
Source value	Specifies the value in the field that identifies the logical source. This value is case-sensitive: You must enter the value using the same casing as in the field.
Count	Specifies whether the transform uses the source, based on the value you entered in the *Source value* column, in the count. • *YES*: If the transform finds the value specified in the *Source Value* column, include the source in the count. • *NO*: If the transform finds the value specified in the *Source Value* column, don't include the source in the count.

3.5.11.8.4 Match and Associate transforms Unique ID options

Use the Unique ID options to assign unique sequential identification numbers to each new record that you add to a data warehouse.

The Unique ID post-match processing operation enables you to begin numbering where the highest unique ID from the previous run ended.

> **Example**
>
> The transform can carry over the largest number assigned in a particular project as the beginning identification number plus 1. The transform uses the new beginning identification number in the assignment of new sequential ID numbers. The transform assigns new sequential numbers when the software processes the next source against the data warehouse file.

In addition to configuring the transform to assign unique ID numbers to individual records, configure the transform to assign the same unique ID to every record in a match group.

Add the *Unique ID* operation as a *Post Match Processing* operation in the *Match Editor*.

Unique ID options descriptions

Option	Description
Unique ID name	Specifies a name for the Unique ID operation. Make sure that you use a name that is unique within this Match transform. For example, use the match transform name and the match level in the unique ID name. **i Note** This field is the same as the *Unique ID name* field in the *Destination Protection* tab.
Processing operation	Specifies the type of unique ID processing operation to perform. Valid values include the following: • *Assign*: Assigns a new ID to unique records that need one, or assigns a new ID to all members of a group that don't have an ID. In addition, the assign operation uses an existing ID when a member of a match group already has an ID. All match group members must appear consecutively in one collection and must be in priority order (high to low). • *AssignCombine*: Performs both an assign operation and a combine operation. In addition to the Assign process, the transform combines the ID numbers 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). • *Combine*: Combines the IDs of a match group when more than one ID is represented. All match group members must appear consecutively in one collection and must be in priority order (high to low). • *Delete*: Removes unique ID numbers from records that already have an ID, unless the record is protected. • *Split*: Splits the ID numbers of an ID group when more than one match group is represented. All ID group members must appear consecutively in one collection and must be in priority order (high to low).
Recycle unique IDs	Specifies whether the software should reuse the unique IDs that were discarded during the delete operation in different records. If you have a limited amount of unique ID numbers available, you may want to recycle them. Valid values include: *Yes*: Recycle discarded unique IDs. *No*: Do not recycle discarded unique IDs.
ID field	Specifies a field that holds a previously assigned unique ID. If you do not specify a field, then the transform assumes that no records have a unique ID.
Starting unique ID source group	

Option	Description
Field	Specifies an input field from which the transform obtains the starting unique ID. For the field to appear in the dropdown list, map it from an upstream transform.
Starting unique ID field	Specifies the field that passes the starting unique ID. If the transform does not receive a Unique ID, it uses 1 as the starting number.
Constant value	Specifies that the starting ID is a positive whole number in the *Starting value* option.
Starting value	Indicates the starting unique ID value. Valid values range from 1 to UINT_MAX (unsigned integer max). The default value is 1.
Value from file	Specifies to read the starting Unique ID from the file specified in the *File* option.
File	Specifies the path and name of the file that manages unique ID numbers. Specify a value for this option only when the *Value from file* option is selected.
GUID	Specifies to use the Globally Unique Identifier (GUID) as the unique ID. GUID is also known as the Universal Unique Identifier (UUID). The UUID variation used for unique ID is a time-based 36-character string with the format: `TimeLow-TimeMid-TimeHighAndVersion-ClockSeqAndReservedClockSeqLow-Node`.
Save ending ID to file and reclaim recycled IDs	Specifies whether to save the last ID that was assigned to a file. Additionally, specifies whether to reclaim recycled IDs.
File	Specifies the file to which the transform writes the last assigned ID.
Allow multiple Match transforms to access unique ID file	Specifies to allow multiple Match transforms to access a shared unique ID file. When enabled, multiple data flows access the same unique ID file, and single Match transforms run in more than one process when the DOP setting is greater than 1. In addition, this option allows multiple Match transforms within a single data flow to share a single unique ID file.

Option	Description
Number of IDs to get when accessing file	Specifies the number of IDs to retrieve from the unique ID file during each access. For example, with a setting of 100, the first process accesses the file and retrieve IDs numbered 1-100. The next process with retrieve IDs numbered 101-200. If a process uses less than the number of retrieved IDs, the remaining IDs are written back to the unique ID file as recycled IDs. **i Note** A setting greater than 1 improves performance when sharing a unique ID file between multiple processes by reducing the number of times the file must be accessed. However, integer numbers may not be assigned in sequential order.
Group number field	Specifies the field that holds a group number. The group number is used to assign the same unique ID to more than one record. If this field is omitted, then it is assumed that each record is unique and should have its own number.

Related Information

Unique ID
Assigning unique IDs using a file

3.5.11.8.5 Match and Associate transforms Destination Protection tab

Use the *Destination Protection* tab to control whether a unique ID for a record is protected based on the source that the record belongs to.

Destination protection helps prevent IDs from being assigned to a suppression or a rented source.

Destination Protection table option descriptions

Option	Description
Unique ID name	Specifies a name for this unique ID operation. Make sure that the name is unique within this match transform. If you use other unique ID operations in this Match transform, use both the match transform name and the match level in the Unique ID name to distinguish it from other unique ID operations. **i Note** This field is the same as the *Unique ID name* field in the *Unique ID* tab.
Enable destination protection	Enables the protection of the source record unique ID numbers from being overwritten with the unique ID numbers of matching records.
Default destination protection	When you set destination protection for records based on fields: Specifies whether a destination is protected when the field that you specify in *Unique ID protected field* option does not have a valid value. • *Yes*: Protects a destination when the destination protection field does not have a valid value of Y or N. • *No*: Disables this type of protection When you set destination protection for records based on input source membership: Specifies whether a destination (input source) is protected when you do not specifically define the source in the table.
Specify destination protection by field	Specifies to enable destination protection through a value in a field.
Unique ID protected field	Specifies an input field 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) causes the transform to use the setting in *Default destination protection* option.
Specify destination protection by source	Specifies to control destination protection through membership in a particular source. Complete the table with source names and destination protection settings.

Option	Description
Source name	Choose the name of the source from the dropdown list. The transform populates the list with defined sources and source groups from the *Input Sources Editor* window of the *Match Editor*. If the dropdown list is empty, or you don't see the specific source in the list, add the source in the *Input Sources Editor*. To access the *Input Sources Editor*: 1. Click the *Transform Options* node at left. 2. Click *Add* at the bottom of the pane at left. 3. Click *Input Sources*.
Unique ID protected	Specifies whether to protect the source in the *Source name* column. • *Yes*: The source is protected. • *No*: The source is not protected.

3.5.11.8.6 Match transform Output Flag Selection options

Add an *Output Flag Selection* operation to match levels in Post Match Processing to flag specific record types for evaluation or routing downstream in your data flow.

Adding this operation generates the `<Select_Record>` output field that you include in the output schema. The transform populates this field with a Y or N based on the type of record you select in the operation.

> **i Note**
>
> The Output Flag Selection operation does not appear in the Post Match Processing dropdown list until you map output fields in the *Output* tab of the transform.

Add multiple Output Flag Selection operations to the match level if required.

Record type	Description
Output flag selection name	Specifies a unique name for this operation that enables you to identify it in a report and in the `<Select_Record>` output field. You could, for example, use the match level and the operation name as the unique name. If you have two *Output Flag Selection* operations in one match level named DMA_Matches and Mail_List, the unique names appear as: • <match level name>_DMA_Matches_Select_Record • <match level name>_Mail_List_Select_Record
Unique	Flags records that are not members of any match group, or records with no matching records. Unique records can be from normal-type or special-type sources.
Single source masters	Flags the highest ranking member of a match group whose members came from the same source. Single source master records can be from normal-type or special-type sources.

Record type	Description
Single source subordinates	Flags records that came from a normal-type or suppress-type source and is a subordinate member of a match group.
Multiple source masters	Flags the highest ranking member of a match group whose members came from two or more sources. Multiple source master records can be from normal-type or special-type sources.
Multiple source subordinates	Flags records that are the subordinate record of a match group that came from a normal-type or suppress-type source whose members came from two or more sources.
Suppression matches	Flags a subordinate member of a match group that includes a higher-priority record that came from a suppress-type source. Suppression matches can be from normal-type or special-type sources.
Suppression uniques	Flags records that came from a suppress-type source, and for which no matching records were found.
Suppression masters	Flags records that came from a suppress-type source and are the highest ranking member of a match group.
Suppression subordinates	Flags records that came from a suppress-type source and are a subordinate member of a match group.

3.5.11.9 Match transform dynamic input field for

Set up a match criteria using Geo Proximity to include the dynamic input field for the Match transform.

The following field appears automatically in the Input tab of the Match transform when you configure a match criteria and you select the Geo Proximity as the *Compare data using* setting.

Input field name (Match)	Description
Option_Field_Algorithm_Geo_Proximity_<logical_name>_Max_Distance	The distance used in proximity matching. This setting is dynamic. If you change this setting, you do not have to terminate and reinitialize the transform in order for the new configuration to be recognized. <logical-name> is a name used to reference option groups. Some option groups can be repeated and the transform needs a way to uniquely identify each option group. After you configure Geo Proximity criteria in the Match Editor, the software automatically adds the input field to the Input tab. The software also adds a new option group to the Options tab named *Field Algorithm Geo Proximity Group*. The software names the field using the field name with <_GP> added to the end. For example, if you set the Geo Proximity criteria that uses the input field <ADDRESS_PRIMARY_NAME>, the field name in the options tab is <ADDRESS_PRIMARY_NAME_GP>. The input field name on the Input tab is <Option_Field_Algorithm_Geo_Proximity_Address_Primary_Name_GP_Max_Distance.>

Related Information

Designer Guide: Set up Geographic Proximity matching - Criteria options

3.5.11.10 Match transform output fields

The Match transform generates output fields for various types of operations run either pre- or post-match.

The following tables list the output field in the following groups:

- Match level
- Input source operation
- Group statistics post-match, generate source statistics from input sources
- Group statistics post-match, generate source statistics from source values
- Source group operation
- Group statistics operation
- Unique ID operation
- Group prioritization operation
- Output flag election

Match-level output fields

The transform generates the following output fields for each match level. Use these fields when you map your output schema.

Match level output fields

Output field name (Match)	Description
ALL_MATCH_CRITERIA *ALL_MATCH_RECORDS*	For internal use only
BREAK_KEY	Contains the break key generated for this record.
CONFLICT_RECORD	Contains the sequence number of the record that has a conflict with this record. A conflict happens when records that do not match directly are in the same match group. **Example** Records A, B, and C are in the same match group. Records A and B match record C Records A and B do not directly match each other Records A and B are conflict records The field is blank when there are no conflict records for this record. **! Restriction** Pattern matching is currently not supported. However, when pattern matching is supported, the following restriction exists for this field: Select this field for output only when you create your job with Information Steward Data Cleansing Advisor and you have the Match transform configured for pattern matching. If you use this field under any other condition, the output is blank.

Output field name (Match)	Description
FIRST_MATCH_RECORD	Contains the sequence number for the first record that matched to this record. Contains a value of Null when this record is unique. **! Restriction** Currently, the *MATCH_MODE* option is not available for edit. However, when *MATCH_MODE* is supported, the *FIRST_MATCH_RECORD* output field is useful when you set *MATCH_MODE* to *MULTI_DRIVER* to determine the first match record. **! Restriction** Pattern matching is currently not supported. However, when pattern matching is supported, the following restriction exists for this field: Select this field for output only when you create your job with Information Steward Data Cleansing Advisor and you have the Match transform configured for pattern matching. If you use this field under any other condition, the output is blank.
GROUP_NUMBER	Contains the records that belong to the same match group and that share the same group number. The group numbers start with the number 1. Unique records have a blank group number. If you use association matching in your job, map *GROUP_NUMBER* on output because the Associate transform uses it.
MATCH_CRITERION	Contains the name of the criteria or pattern that decided that the records match. Applicable when *MATCH_TYPE* contains *R*. The field is blank if the *MATCH_TYPE* output field does not contain *R*. **i Note** Pattern matching currently not available.
MATCH_LEVEL	Contains the name of the match level the transform used for this record.

Output field name (Match)	Description
MATCH_PAIR_COUNT	Contains the number of records that matched this record. • Unique record has a value of 0. • Record that matched all other records in a match group has a value that is one less than the match group count. • If you set *Match Mode* to *Single Driver*, then the driver record has a match pair count equaling one less than the match group count. The passengers have a match pair count of 1. **i Note** The *Match Mode* option is currently not available for edit. **! Restriction** Pattern matching is currently not supported. However, when pattern matching is supported, the following restriction exists for this field: Select this field for output only when you create your job with Information Steward Data Cleansing Advisor and you have the Match transform configured for pattern matching. If you use this field under any other condition, the output is blank.
MATCH_SCORE	Contains one of the following types of match score. • The criteria or pattern similarity score when the *MATCH_TYPE* is *R*. • The total weighted score when the *MATCH_TYPE* is *W*. • Blank if the record is a driver record (*MATCH_TYPE* of *D*) or when the records are unique. **i Note** Pattern matching is not currently available.

Output field name (Match)	Description
MATCH_STATUS	Contains the status of the record in relationship to the match group. • *D:* Record is a driver in a match group. • *P:* Record is a passenger in a match group. • *U:* Record is unique. • *B:* Record is both a driver and a passenger. Applicable when you set the *Match Mode* option to *Multi-Driver*. **i Note** Currently, the *Match Mode* option is not available.
MATCH_TYPE	Contains the type of match that brought the record into a match group. • *blank*: Record is unique. • *D*: Driver record. • *R*: Rule (criteria) or pattern. • *W*: Weighted score. The total weighted score met the Weighted match score. **i Note** Pattern matching is not currently available.
REVIEW_RECORD	Contains the sequence number of the record that is a match, but something about the pair of matching records questionable. For records that do not need review, the value is blank. **! Restriction** Pattern matching is currently not supported. However, when pattern matching is supported, the following restriction exists for this field: Select this field for output only when you create your job with Information Steward Data Cleansing Advisor and you have the Match transform configured for pattern matching. If you use this field under any other condition, the output is blank.

Output field name (Match)	Description
SEQUENCE_NUMBER	Contains the number automatically assigned by the transform. The transform stores this number internally as a 64-bit integer. The transform assigns 1 to the first record input record after initialization. The transform continues assigning numbers in sequential order. The sequence number enables the transform to connect two records by having each record reference the other record number. The field data type is Decimal. **! Restriction** Pattern matching is currently not supported. However, when pattern matching is supported, the following restriction exists for this field: Select this field for output only when you create your job with Information Steward Data Cleansing Advisor and you have the Match transform configured for pattern matching. If you use this field under any other condition, the output is blank.

Input Source operation output fields

The following table contains output fields that are available when you use an input source operation in the transform.

Input source operation output fields

Output field name (Match–Input Source)	Description
SOURCE_GROUP_NAME	Contains the name of the Source Group that the current record belongs to. If a record does not belong to any Source Group, then the field is empty.
SOURCE_NAME	Contains the name of the input source to which the current record belongs.
SOURCE_TYPE	Contains the type of source that the current record belongs to. • *N:* Normal source. • *P:* Suppress source. • *S:* Special source.

Source statistics from input sources post-match output fields

The following table contains fields available for group statistics post-match operations where you enable the *Generate source statistics from input sources* option.

The fields are in addition to the fields generated by the pre-match group statistics operation.

Source statistics from input sources post-match output fields

Output field name (Match–Input Source)	Description
GROUP_COUNT	Contains 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	Contains the order in which the input source appears in this match group. The transform assigns the order number sequentially with the first group that appears in the match group receiving a 1. Records that come from the same input source receive the same order number. Unique records have a value of 0.
GROUP_SOURCE_GROUP_ORDER	Contains the order in which the records that have the same *GROUP_SOURCE_GROUP_APPEARANCE* value appear in the match group. The transform assigns the order number sequentially with the first occurrence receiving a 1. The transform assigns a 0 to unique records. The transform also assigns a 0 to records in a match group that do not belong to a source group.
GROUP_SOURCE_ORDER	Contains the order of the records within the match group that have the same *GROUP_SOURCE_APPEARANCE* value. The transform assigns the order number sequentially with the first occurrence receiving a 1. The transform assigns a 0 to unique records.

Output field name (Match–Input Source)	Description

GROUP_SOURCE_TYPE	Contains the type of source for the records in the match group.
	• *M:* Records are from more than one input source, excluding records from special sources.
	• *P:* Records are from a suppress source.
	  • If the master record comes from a suppression source, then all records in the match group have a *P*
	  • If the master record comes from a normal or special source, then the suppression record and all records after it, have a *P*, but the records before the suppression record have an *M* or *S*.
	• *S:* Records are from a single input source.
	• *blank*: Records are unique.

MULTI_SOURCE_COUNT	Contains the number of sources represented in the match group, excluding the following source types: • Special sources • Suppress sources • 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. For unique records: • If a unique record is from a normal source, it has a value of 1. • If a unique record is from a special or suppression source, it has a value of 0.
SOURCE_COUNT	Specifies the number of sources represented in the match group, regardless of the source types. The values of this output field can range from 1 to the number of records in the match group. Unique records have a value of 1.
SOURCE_GROUP_COUNT	Contains the number of source groups represented in the match group. The transform does not count records in the match group that do not belong to a source group. The value of this output field range from 0 to 10,000, which is the maximum number of input sources allowed. Unique records have a value of 0 or 1.

Group statistics from source values post-match output fields

The following table contains fields available for group statistics post-match operations where you enable the *Generate source statistics from source values* option.

The fields are in addition to the fields generated by the group statistics operation.

Group statistics for source values post-match output fields

Output field name (Match–Input Source)	Description
Group_Count	Provides the total number of records in the match group. Unique records have a value of 1.
Group_Order	Contains the order number for the group in which the record belongs. • Master record = 1. • Subordinate records = 2 through the number of records in match group • Unique records = 0 Control the group order by including a Prioritization operation in the Post Match processing group.
Group_Rank	Contains the record type the record has in the group. • *M*: Master record • *S*: Subordinate record • *blank*: Unique record

Group_Type	Specifies whether a record contributed to the source count, and if so, whether there were other sources represented in the match group.
	i Note
	Records from multiple sources are not counted toward a multiple-source match group.
	• *M:* Multiple sources.
	• *S:* Single source.
	• *P:* At least one record from a Suppression source is included in the match group.
	  • If the master record comes from a suppression source, then all records in the match group have a *P*.
	  • If the master record comes from a normal or special source, then the suppression record, and all records after it, has a *P*, but the records before the suppression record have an *M* or *S*.

Output field name (Match–Input Source)	Description
SOURCE_COUNT	Contains 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 have a value of 1.
SOURCE_ID	Contains the logical source identification. In most cases, the logical source identification is the input source value. In other cases, it is the default logical source value.
SOURCE_ID_COUNT	Contains the number of source IDs represented in the match group.
SOURCE_TYPE_ID	Contains the type of logical source. • *N:* Normal source • *P:* Suppress source • *S:* Special source

Source Group operation output fields

The fields in the following table are generated only if you use a source group operation in your Match transform.

Source group operation output fields

Output field name (Match–Source Group)	Description
GROUP_SOURCE_GROUP_APPEARANCE	Contains the order in which the source group appears in this match group. The transform assigns 1 to the first group that appears in the match group. The numbering continues sequentially. Records that come from the same source group receive the same value. Unique records receive a value of 0. Records in a match group that are not assigned to a source group also receive a value of 0.
GROUP_SOURCE_GROUP_ORDER	Contains the order of the records within the match group that have the same *GROUP_SOURCE_GROUP_APPEARANCE* value. The transform assigns 1 to the first occurrence. The numbering continues sequentially. Unique records receive a value of 0. Records in a match group that are not assigned to a source group also receive a value of 0.

Output field name (Match–Source Group)	Description
SOURCE_GROUP_COUNT	Contains the number of source groups represented in the match group. The transform does not count records in the match group that do not belong to a source group. Values range from 0 to the number of records in the match group. The transform assigns either 0 or 1 to unique records.
SOURCE_GROUP_NAME	Contains the name of the source group to which the current record belongs. If a record does not belong to any source group, the field is empty.

Group Statistics operation output fields

The following table contains fields generated only when you use group statistics operations in your Match transform.

Output field name (Match–group statistics)	Description
CONFLICT_GROUP	Indicates whether the record is in a match group that includes a conflict. • *C*: Record is a part of a match group that includes a conflict. • *blank*: Record is not part of a match group that includes a conflict. A conflict occurs when two records in a match group do not match directly. **Example** Records A and B match record C Records A and B do not directly match each other Records A and B are conflict records **! Restriction** Pattern matching is currently not supported. However, when pattern matching is supported, the following restriction exists for this field: Select this field for output only when you create your job with Information Steward Data Cleansing Advisor and you have the Match transform configured for pattern matching. If you use this field under any other condition, the output is blank.

Output field name (Match–group statistics)	Description
GROUP_COUNT	Contains the total number of records in the match group. Unique records have a value of 1.
GROUP_ORDER	Contains the order number for the group in which the record belongs. • *1*: Master record. • *2* through the number of records in match group: Subordinate records. • *0*: Unique record. Control the group order by including Prioritization operation in the Post Match processing group.
GROUP_RANK	Contains the record type the record has in the group. • *M*: Master record • *S*: Subordinate record • *blank*: Unique record

GROUP_TYPE	Specifies whether a record contributed to the source count, and if so, whether there were other sources represented in the match group.
	
	i Note
	Records from multiple sources are not counted toward a multiple-source match group.
	
	• *M:* Multiple sources.
	• *S:* Single source.
	• *P:* At least one record from a Suppression source is included in the match group.
	  • If the master record comes from a suppression source, then all records in the match group have a *P*.
	  • If the master record comes from a normal or special source, then the suppression record, and all records after it, has a *P*, but the records before the suppression record have an *M* or *S*.

Output field name (Match–group statistics)	Description
REVIEW_GROUP	Indicates that the record is a part of a match group for which the transform recommends a review. • *R*: Record is a part of a match group with review records. • *blank*: Record does not belong to a match group with review records. The transform recommends a review when a pair of matching records is identified as questionable. **! Restriction** Pattern matching is currently not supported. However, when pattern matching is supported, the following restriction exists for this field: Select this field for output only when you create the job with Information Steward Data Cleansing Advisor and you have the Match transform configured for pattern matching. If you use this field under any other condition, the output is blank.
SOURCE_COUNT	Contains the number of logical sources in this match group.
SOURCE_ID	Contains the logical source identification. In most cases, the logical source value is the input source value. In other cases, it is the default logical source value.
SOURCE_ID_COUNT	Contains the number of source IDs represented in the match group.
SOURCE_TYPE_ID	Contains the type of logical source. • *N:* Normal source • *P:* Suppress source • *S:* Special source

Unique ID operation output fields

These fields are available only if you use a Unique ID operation in your Match transform.

Output field name (Match–Unique ID)	Description
ID_STATUS	Contains the status of the *Unique_ID* output field. • *blank*: *UNIQUE_ID* output field is the same as the *UNIQUE_ID* input field. • *D*: *UNIQUE_ID* output field is blank and the previous *UNIQUE_ID* is deleted. • *N*: *UNIQUE_ID* output field has a new ID. • *O*: *UNIQUE_ID* output field assigned old or existing *UNIQUE_ID*. **i Note** A status of *0* happens when a record is combined.
RECORD_OPERATION	Contains the operation that the transform should perform on the record based on the input fields, excluding the protected fields. • *blank*: No operation. • *A*: Assign a unique ID to the record. • *C*: Combine the record unique ID. • *D*: Delete the record unique ID. • *S*: Split the record unique ID.
UNIQUE_ID	Contains the unique ID that the transform generates for this record. If the record already has a valid unique ID, then the transform outputs the same unique ID. If the transform does not assign a unique ID, the output field is blank.

Group Prioritization output fields

The following output field is available when you add a Group Prioritization operation to a Match transform.

Output field name (Match–Group Prioritization)	Description
PRIORITY_VALUE	Contains the sum of all priority and blank penalty values that you define in *Record Completeness* tab of the *Group Prioritization* editor. If you do not order records using the *Record Completeness* tab, this field contains *0*.

Output flag selection output fields

The following output field is available when you add an Output flag election operation to a Match transform.

Output field name (Match–Output flag)	Description
SELECT_RECORD	Specifies whether the transform should select the current record based on your selections in the *Output Flag Selection Editor*. • *Y*: Transform should select the record. • N: Transform should not select the record.

3.5.12 USA Regulatory Address Cleanse transform

Use the USA Regulatory Address Cleanse transform to identify, parse, validate, and correct U.S. address data according to the U.S. Post Office standards.

USA Regulatory Address Cleanse information

Characteristic	Description
	USA Regulatory Address Cleanse icon

Characteristic	Description
Use	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). When you use the required options, this transform creates the USPS Form 3553 and enhances your records with many useful codes. When you run the transform in a non certification mode, you can use the suggestion list feature. **i Note** If an input record has characters not included in the Latin1 code page, the USA Regulatory Address Cleanse transform does not process that data. Instead, the transform sends the mapped input record to the corresponding standardized output field, when applicable. The transform does not populate any other output fields 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, place the USA Regulatory Address Cleanse transform before the Data Cleanse transform and any of the Match transforms in the data flow. We recommend using a sample job or data flow that you set up according to best practices for a specific use case.
Content objects	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. Additionally, download any of the USA Regulatory Address Cleanse blueprints from the SAP SCN page.

USA Regulatory Address Cleanse Report and Analysis options [page 687]
Configure the options in the *Report and analysis* option group to generate reports.

USA Regulatory Address Cleanse reference files [page 687]
Set the location for reference files that the USA Regulatory Address Cleanse transform uses to correct and standardize your data.

USA Regulatory Address Cleanse transform performance options [page 692]
Set options in the *Transform Performance* group to enhance performance for DPV, DSF2, LACSLink, NCOALink, RDI, and SuiteLink processing.

USA Regulatory Address Cleanse USPS License Information [page 693]
If you perform NCOALink, SuiteLink, LACSLink, DPV, or DSF2 processing, complete the required USPS license information.

USA Regulatory Address Cleanse transform NCOALink options [page 696]
Complete the options in the NCOALink group when you prepare your data for NCOALink certification.

USA Regulatory Address Cleanse assignment options [page 707]
Set options in the Assignment Options group to enable the USPS processes that the transform performs on your data.

USA Regulatory Address Cleanse standardization options [page 710]
Set options for standardizing your address data in the Standardization group of options.

USA Regulatory Address Cleanse non-certified options [page 720]
The Non-Certified options group contains settings that, when enabled, create a mailing list that is not compliant with CASS rules.

USA Regulatory Address Cleanse USPS CASS Report Options [page 723]
Complete the *CASS Report Options* group with USPS Form 3553 information as required by CASS.

USA Regulatory Address Cleanse suggestion list options [page 724]
The *Suggestion List* group contains options to configure how the software outputs suggestions lists.

USA Regulatory Address Cleanse suggestion list components [page 727]
Complete options in the *Suggestion List Components* to specify the address field components to include in the *Suggestion_List* output field.

USA Regulatory Address Cleanse USPS Z4 Change options [page 729]
Set USPS Z4 Change options to enable Z4Change processing and to specify the last time the Postcode 2 was updated.

USA Regulatory Address Cleanse input fields [page 730]
The USA Regulatory Address Cleanse transform has recognized input fields to use in field mapping.

USA Regulatory Address Cleanse output fields [page 732]
Use the USA Regulatory Address Cleanse transform output fields for generated data and to add additional fields to your output.

Related Information

Download Data Quality blueprints and other content objects [page 340]

3.5.12.1 USA Regulatory Address Cleanse Report and Analysis options

Configure the options in the *Report and analysis* option group to generate reports.

Report and Analysis group option descriptions

Option	Description
Gather Statistics Per Data Source	Specifies whether to generate report data for each data source specified in the *Data_Source_ID* input field. • *YES*: If the *Generate Report Data* option is enabled, generates report statistics for each data source specified in the *Data_Source_ID* input field. • *NO*: If the *Generate Report Data* option is enabled, generates reports for the input database without generating statistics, based on the *Data_Source_ID* input field. **i Note** If you select *YES*, other setup requirements apply.
Generate Report Data	Specifies whether to generate report data for this transform. • *YES*: Generates report data for this transform. • *NO*: Does not generate report data for this transform. **i Note** If you do not need to generate reports, such as for testing, set this option to *NO* to improve performance.

Related Information

Statistics for multiple data sources

3.5.12.2 USA Regulatory Address Cleanse reference files

Set the location for reference files that the USA Regulatory Address Cleanse transform uses to correct and standardize your data.

Because there are so many reference files, we recommend that you use the applicable substitution variable instead of typing the location for each file.

> **⚠ Caution**
>
> The reference file substitution variables represent the path to the reference files. If you change the location in the substitution variable, you change the reference file location for all jobs that use the substitution variable.

Reference Files option group descriptions

Option	Description
Address Directory 1	`zip4us.dir` Specifies the path to the National Directory. The National Directory is organized by ZIP Code. It lists street names, ranges of house numbers, postal, and other codes. Use substitution variable `$$RefFilesAddressCleanse`.
Address Directory 2	`*.dir` Optional. Specifies the path to a directory that you customize for a special purpose. We do not provide a directory for this option. Most users should leave the *Address Directory 2* option blank.
Address Geo 1 Directory *Address Geo 2 Directory* *Address Geo 3 Directory* *Address Geo 4 Directory* *Address Geo 5 Directory* *Address Geo 6 Directory* *Address Geo 7 Directory* *Address Geo 8 Directory* *Address Geo 9 Directory* *Address Geo 10 Directory*	`ageo1.dir` to `ageo10.dir` Specifies the path to the Address-level GeoCensus files. Required only if you use the Address-level GeoCensus option or if you use the Geocoder transform. Use substitution variable `$$RefFilesAddressCleanse`.
Address SHS Directory	`zip4us.shs` Required for standardization. Specifies the path to the SHS directory. The SHS directory enhances normal primary name lookups. Use substitution variable `$$RefFilesAddressCleanse`.

Option	Description
Centroid Geo Directory	`cgeo2.dir` Required only if you use the centroid-level GeoCensus option or if you use the Geocoder transform. Specifies the path to the Centroid Geo directory. Use substitution variable `$$RefFilesAddressCleanse`.
City Directory	`city##.dir` Specifies the path to the City directory. The City directory contains a table of city names, states, and ZIP Codes organized by state and city. Use substitution variable `$$RefFilesAddressCleanse`.
DPV Path	Required for CASS certification. Specifies the path to the DPV (Delivery Point Validation) directory files. The DPV directory contains information that identifies whether a ZIP+4 coded address represents a known address and that the address is deliverable. Use substitution variable `$$RefFilesAddressCleanse`.
DSF2 Augment Path	Specifies the path to the directory that contains the DSF2 (Second Generation Delivery Sequence) files. The DSF2 Augment Path directory contains information about rural routes, high contract routes, and so on. Use substitution variable `$$RefFilesAddressCleanse`.
eLOT Directory	`elot.dir` Specifies the path to the eLOT directory. The eLOT directory contains codes for the delivery point that represents the delivery route walk sequence for all mail carriers. Include only if the *Enable eLot* option in the Assignment Options group is set to *YES*. Use substitution variable `$$RefFilesAddressCleanse`.

Option	Description
EWS Directory	`ewsyymmdd.dir` Specifies the path to the EWS directory (Early Warning System). The transform lists the directory in the *Option Value* column so the transform finds the most current directory. EWS reduces the number of misdirected mail by providing valid delivery points that are created between national directory updates. Use substitution variable `$$RefFilesAddressCleanse`.
LACSLink Path	Specifies the path to the LACSLink directory files. Required for CASS certification. The LACSLink directories update addresses when the physical address changes, not when residents move. Use substitution variable `$$RefFilesAddressCleanse`.
NCOALink Path	Specifies the path to the NCOALink directory files. Required for NCOALink processing and certification. The NCOALink directories contain records of addresses changed through a change of address form filed with the USPS. Use substitution variable `$$RefFilesAddressCleanse`.
Postcode Directory	`zcf10.dir` Specifies the path to the Postcode directory. The Postcode directory contains the same data as the City directory, but is organized by ZIP Code. Use substitution variable `$$RefFilesAddressCleanse`.
Postcode Reverse Directory	`revzip4.dir` Specifies the path to the Postcode Reverse directory. The Reverse directory enables the software to assign more postal codes when the input data includes a unique ZIP Code and valid ZIP+4. Use substitution variable `$$RefFilesAddressCleanse`.

Option	Description
RDI Path	Specifies the path to the RDI directory. The RDI directory indicates whether an address is residential. Use substitution variable `$$RefFilesAddressCleanse`.
Reverse Soundex Address Directory	`zip4us.rev` Specifies the path to the Reverse Soundex Address directory. The Reverse Soundex Address directory enhances primary name lookups. Use substitution variable `$$RefFilesAddressCleanse`.
SuiteLink Path	Required for CASS certification. Specifies the path to the SuiteLink directories. The SuiteLink directories contains specially indexed address information. For example, SuiteLink directories contain secondary numbers and unit designators for locations identified as high-rise business default buildings. Use substitution variable `$$RefFilesAddressCleanse`.
USPS Log Path	Specifies the path to the directory for NCOALink, DPV, and LACSLink log files. This directory must already exist and be writable. The transform determines the file names during processing as the USPS requires. **i Note** Use the same path to all jobs. If you have multiple clients, use the same log file directory for all clients so that the transform combines the log files. Use substitution parameter `$$CertificationLogPath`.
Z4 Change Directory	`z4change.dir` Specifies the path to the Z4 Change directory. The Z4 Change directory lists all the ZIP Codes and ZIP+4 Codes in the country. Use substitution variable `$$RefFilesAddressCleanse`.

SAP provides a directory update letter that contains information about the current directory versions and how to obtain directories. Follow this link for the latest directory update letter: https://uacp2.hana.ondemand.com/viewer/p/ADDRESSING_DIRECTORIES.

3.5.12.3 USA Regulatory Address Cleanse transform performance options

Set options in the *Transform Performance* group to enhance performance for DPV, DSF2, LACSLink, NCOALink, RDI, and SuiteLink processing.

Transform Performance option group descriptions

Option	Description
Cache DPV Directories	Specifies whether the software caches DPV directories into memory. The software caches the directories only once and any other DPV threads that are running in the data flow use the same cache. • *YES*: Enables caching. • *NO*: Disables caching.
Cache DSF2 Augment Directories	Specifies whether the software caches DSF2 Augment directories into memory. • *YES*: Enables caching. • *NO*: Disables caching.
Cache LACSLink Directories	Specifies whether the software caches the LACSLink directories into memory. • *YES*: Enables caching. • *NO*: Disables caching.
Cache RDI Directories	Specifies whether the software caches RDI directories into memory. • *YES*: Enables caching. • *NO*: Disables caching.
Cache SuiteLink Directories	Specifies whether the software caches the SuiteLink directories into memory. • *YES*: Enables caching. • *NO*: Disables caching.

Option	Description
Insufficient Cache Memory Action	If there is insufficient memory to cache the directories that you have set up for caching, specifies the action the software takes. • *CONTINUE*: Attempts to continue initialization without caching. • *ERROR*: Issues an error and terminates the transform.
NCOALink Caching Mode	Specifies the method for caching NCOALink directories. • *AUTO*: The software uses available memory for caching. • *MANUAL*: Enter a value in the *NCOALink Memory Usage* option. **i Note** Select *MANUAL* if you have a limited amount of memory available and you want to allocate a set amount of memory for caching. • *NONE*: Disables caching. *NONE* is the default setting. **i Note** Select *NONE* for smaller jobs. Executing a smaller job could take less time than the time to cache the directories.
NCOALink Memory Usage	Specifies a value to allocate a set amount of memory for NCOALink directory caching. The software uses this value as the maximum amount of memory that it can use for caching the NCOALink directories. Required if you set the *NCOALink Caching Mode* to *MANUAL*. **i Note** This value is not for memory per thread. If the *Degree of Parallelism* value is greater than one, the *NCOALink Memory Usage* value is the total to be allocated for all threads.

3.5.12.4 USA Regulatory Address Cleanse USPS License Information

If you perform NCOALink, SuiteLink, LACSLink, DPV, or DSF2 processing, complete the required USPS license information.

Provide USPS license information for the licensee and the customer.

- Licensee is the company performing the processing.
- Customer is the company for whom the licensee is performing the processing.

If you are performing the processing for yourself, you are the end user.

When available, use the applicable substitution variables for these options.

USPS License Information group option descriptions

Option	Description
Customer Company Name *Customer Company Address* *Customer Company Locality* *Customer Company Region* *Customer Company Postcode1* *Customer Company Postcode2*	Specifies the customer information including name and address. If you are an end user, leave these fields blank. However, complete the fields if you have an Alternate Stop Processing agreement from NCOALink and you have entered a special keycode into Data Services License Manager. The software includes the customer information in the NCOALink Processing Summary report and log files. Use the following substitution variables for these fields: • `$$CompanyName` • `$$CompanyAddress` • `$$CompanyLocality` • `$$CompanyRegion` • `$$CompanyPostcode1` • `$$CompanyPostcode2`
Customer Company Phone	Optional. Use substitution variable `$$CompanyPhone`.
DSF2 Licensee ID	Specifies the DSF2 identification number assigned to the licensee by the USPS. Use substitution variable `$$DSF2LicenseeID`.
IMB Mailer ID	Optional. Specify the unique Intelligent Mail Barcode (IMB) Mailer ID from the USPS, if applicable. Use substitution parameter `$$IMBMailerID`. The IMB Mailer ID is a unique code, 6- or 9-digits, assigned to mailers by the USPS based on their annual mail volumes. The software includes this information in the NCOALink Processing Acknowledgement Form (PAF).

Option	Description
Licensee Name	Required for NCOALink. Specifies the name of the company mentioned in the USPS license agreement. Maximum size is 30 characters. The software includes this information in the PAF log and NCOALink Processing Summary report.
List Owner NAICS Code	Required. Specifies the North American Industry Classification System (NAICS) code, which identifies the business in which the list owner engages. Use substitution parameter `$$CompanyNAICSCode`.
List ID	Required for NCOALink. Specifies the 6-digit Customer or List ID for NCOALink limited and full service providers. If you are an end user, leave this field blank. The List ID is a unique ID assigned by the licensee to identify the customer or list owner. If you are the licensee and you do not have a naming scheme in place, we suggest that you create the 6-digit ID using the following information: • First 3 digits: Customer name or identifier • Last 3 digits: List name identifier
List Processing Frequency	Specifies the number of times per year the list is processed for NCOALink. Enter a 2-digit value from 1 to 52. If the list owner has other lists processed by the NCOALink licensee at different frequencies, enter 99.
List Received Date	Specifies the date when the NCOALink licensee received the list. Use the `yyyy/mm/dd` format. If you are an end user, leave this field blank.
List Return Date	Specifies the date when the list is to be returned to the customer. Use the `yyyy/mm/dd` format. If you are an end user, leave this field blank.
Provider Level	Specifies the provider levels for which the licensee has a registered license keycode. The dropdown list contains the provider levels supported in your registered keycodes. The substitution variable `$$USPSProviderLevel` is the default value.

USPS License Information required options [page 696]

The options in the USPS License Information group are required based on the type of service you provide and are licensed for.

3.5.12.4.1 USPS License Information required options

The options in the USPS License Information group are required based on the type of service you provide and are licensed for.

If you are processing NCOALink, DSF2, DPV, SuiteLink, or LACSLink, the USPS License Information group contains options that you must complete. The following table indicates the USPS services that require a value for each option of the *USPS License Information* group.

> → Tip
>
> For options that have substitution variables, set up the variables in *Tools* > *Substitution Parameter Configurations*.

Option	NCOALink Full Service Provider	NCOALink Limited Service Provider[1]	NCOALink End User	DSF2	DPV	LACSLink
Licensee Name	X	X	X	X		
List Owner NAICS Code	X	X	X	X		
List ID	X	X	X	X		
Customer Company Name	X	X	X	X	X	X
Customer Company Address	X	X	X	X	X	X
Customer Company Locality	X	X	X	X	X	X
Customer Company Region	X	X	X	X	X	X
Customer Company Postcode1	X	X	X	X	X	X
Customer Company Postcode2	X	X	X	X	X	X
Customer Company Phone						
List Processing Frequency	X	X	X			
List Received Date	X	X		X		
List Return Date	X	X		X		
Provider Level	X	X	X			
IMB Mailer ID						
DSF2 Licensee ID				X		

3.5.12.5 USA Regulatory Address Cleanse transform NCOALink options

Complete the options in the NCOALink group when you prepare your data for NCOALink certification.

[1] With NCOALink enabled

NCOALink option group descriptions

Option	Description
Mailing List Name	Specifies the name of the mailing list to be processed. Enter up to 30 characters. If this list is a master house list or your only mailing list, consider entering your company name here. This name appears in the log files.
Platform ID	Specifies the NCOALink licensee identification number. The Platform ID is the identification number assigned to the individual or company who is the NCOALink licensee. The ID is four characters.

3.5.12.5.1 USPS NCOALink processing options

Specify the USPS processes to perform on the data by enabling processes in the Processing Options group.

Processing Options descriptions

Option	Description
Consider Moves Within Months	Specifies whether the transform ignores change of address data older than the specified number of months. **Example** Enter 12 to use change of address data that has a move-effective date within the last 12 months. • If you are an end user or limited service provider, enter a value from 6 through 18. • If you're a full service provider or using ANKLink, enter a value from 6 through 48. • Leave the option blank to enable the transform to use all available data based on your license. Blank is the default setting.
External Processes Updating List	Specifies whether the transform performs additional processes on the list before or after the USA Regulatory Address Cleanse transform.

Option	Description
High Match Rate Expectancy	Specifies whether you expect your data to has a high match rate. Helps the USPS distinguish between files that have a legitimate reason for a high percentage of NCOALink matches and files that do not, which are used to fraudulently create mover lists. • *NONE* or blank: You do not expect a high match rate. *NONE* is the default setting. • *ANKLINK_PROCESSED_LIST*: A legitimate reason for a high match rate. A file that you processed with ANKLink contains records for people who have moved but you don't yet have their new address. **i Note** *ANKLINK_PROCESSED_LIST* is available only to full service providers. • *STAGE_FILE*: A legitimate reason for a high match rate. A Stage file is for USPS testing. If you're performing Stage I or Stage II testing, ensure that the *List Processing Objective* option is set to a Stage option. • *RETURN_MAIL_LIST*: A legitimate reason for a high match rate. A returned mail list file contains records for mail that were returned to sender.

Option	Description
List Processing Mode	Specifies a reason for processing the list. • *CHANGE_OF_ADDRESS*: To update the list with the latest address data. *CHANGE_OF_ADDRESS* is the default setting. • *STATISTICS_ONLY*: To analyze list statistics such as the number of records that have updated addresses or the total number of each type of move. **i Note** When you select *STATISTICS_ONLY*, the transform does not update your records with move-updated address information. • *RETURN_CODES_ONLY*: For informational purposes. The transform posts return codes in the output components *NCOALINK_RETURN_CODE* or *ANKLINK_RETURN_CODE*. Examining the return codes can tell you whether matching records were found in the NCOALink directories and why or why not. **i Note** When you select *RETURN_CODES_ONLY*, the transform does not update your records with move-updated address information.
List Processing Objective	Specifies your reason for using NCOALink on this list: • *EMPLOYEE_TRAINING*: Processing is a part of employee training. • *INTERNAL_DATABASE_TESTING*: Processing a list owned by the licensee for testing a database. • *MARKETING*: Processing to test an external customer list. • *NORMAL*: Processing to update the list before a mailing. *NORMAL* is the default setting. • *STAGE_I_AND_STAGE_II*: Processing to test the matching performance against a USPS test file. The USPS scores the Stage II test file. • *SYSTEM_TESTING*: Processing as part of system testing. For example, test loading USPS file updates. **i Note** When you certify for CASS and DSF2, indicate the reason in the *Assignment Options* > *USPS Certification Testing Mode.*

Option	Description
Processing First Class Mail	Specifies whether you are processing first class mail. Select *YES* or *NO*. *YES* is the default setting.
Processing Periodicals	Specifies whether you are processing periodical mail. Select *YES* or *NO*. *NO* is the default setting.
Processing Standard Mail	Specifies whether you are processing standard mail. Select *YES* or *NO*. *NO* is the default setting.
Processing Package Service Mail	Specifies whether you are processing package service mail. Select *YES* or *NO*. *NO* is the default setting.
Retrieve Move Types	Specifies what type of move information the software generates. • *BUSINESS* • *INDIVIDUAL* • *INDIVIDUAL_AND_BUSINESS* • *INDIVIDUAL_AND_FAMILY* • *INDIVIDUAL_AND_FAMILY_AND_BUSINESSINDIVIDUAL_AND_FAMILY_AND_BUSINESS* is the default setting.

3.5.12.5.2 USPS NCOALink report options

Set the option in the NCOALink Report Options group so the transform includes or excludes return code descriptions in the NCOALink Processing Summary report.

Report option description

Option	Description
Generate Return Code Descriptions	Specifies whether the transform includes return code descriptions along with the return codes in the NCOALink Processing Summary report. • *YES*: Includes return code descriptions with the return codes in the NCOALink Processing Summary report. • *NO*: Does not include return code descriptions in the NCOALink Processing Summary report. *NO* is the default setting. 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. Without this option enabled, the NCOALink Processing Summary report contains only a short summary of the return codes.

3.5.12.5.3 USPS NCOALink output option

Enable this output option so that the transform updates standardized output fields with NCOALink information.

NCOALink Output option description

Option	Description
Apply Move to Standardized Fields	Specifies whether the transform updates applicable standardized output fields with address data from NCOALink processing. **i Note** This option does not affect component type output fields. • *YES*: Updates applicable standardized fields with NCOALink move update address data. *YES* is the default setting. • *NO*: Outputs standardized fields with only the standardized version of the input data.

3.5.12.5.4 USPS NCOALink PAF details

To complete the required NCOALink Processing Acknowledgement Form (PAF), complete options in the PAF Details group.

End users do not have to complete the options in this group.

PAF Details option descriptions

Option	Description
Company Website	Optional. Specifies the company website address for the person signing the PAF.
Customer Alternate Company Name	Specifies an alternate name when the list owner company name is also known by a different name.
Customer Parent Company Name	Specifies the name of the parent company of the list owner company, if applicable.
Date Signed By Licensee	Specifies the date that the NCOALink service provider signs the PAF. Use the following format: yyyy/mm/dd.
Date Signed By Customer	Specifies the date the customer signs the PAF. Use the following format: yyyy/mm/dd.
Email of Person Signing	Optional. Specifies the email address for the person who signs the PAF.
Name Of Person Signing	Specifies the name of the person signing the PAF. Limit of 50 characters.
Title Of Person Signing	Specifies the job title of the person signing the PAF. Limit of 50 characters.
Type	Specifies the reason for this PAF. • *INITIAL*: For initial authorization to process NCOALink for listed customer. • *MODIFIED*: To update information that changed on an existing PAF. • *RENEWAL*: To replace an existing PAF that is expired or is expiring.

Option	Description
Using Alternative PAF	Required. Specifies whether you are using an alternative PAF that is not the official USPS PAF form. • *YES*: Using an alternative PAF. > **i Note** > Requires specific permission from the USPS. • *NO*: Not using the official USPS PAF form. *NO* is the default setting.
Using Cooperative Database	Required for Full and Limited Service Providers. Indicates whether the list is from a cooperative database. • *YES*: List is from a cooperative database. > **i Note** > The USPS requires a PAF for each participant in the cooperative database. > **i Note** > When set to *YES*, the PAF log file contains a "C" to indicate that the processed list is from a cooperative database. *NO*: List is not from a cooperative database. *NO* is the default setting.

3.5.12.5.5 USPS NCOALink service provider options

Complete the information in the Service Provider group to provide required information about the service provider and the mailing list being processed.

End users do not have to complete the options in this group.

When you complete the options in this group, answer as if you have already processed the list.

Service Provider Options descriptions

Option	Description
Additional Notes	Specifies whether the customer submitted a written request for an extension. • *NONE*: The customer did not submit a written request for an extension. *NONE* is the default setting. • *CUSTOMER_REQUESTED_EXTENSION*: The customer submitted a written request for an extension.
Buyer Company Name	Specifies the company or individual who is leasing, renting, or buying the list after you perform NCOALink processing on the list.
Concurrent Processes Performed	Specifies whether you processed this list concurrently with NCOALink processing. Select *YES* or *NO*.
Concurrent Processed Data Modified	Specifies whether concurrent processing you performed during NCOALink processing changed the data. • *NO*: Concurrent processes performed during NCOALink processing did not change the data. *NO* is the default setting. • *FROM_POSTAL_DATA*: Concurrent processes performed during NCOALink processing added postal data to the list. • *FROM_NON_POSTAL_DATA*: Concurrent processes performed during NCOALink processing added non postal data to the list. • *FROM_BOTH*: Concurrent processes performed during NCOALink processing added both postal and non postal data to the list.
In-House List Processing	Required for Full Service Providers. Specifies whether the list is an in-house or internal list. Select *YES* or *NO*. **i Note** When set to *YES*, the transform adds an "I" to the Customer Service log to indicate that the list is an in-house list.
Output Returned	Specifies the type of output the NCOALink job returned to the client. • *STANDARD*: The job returned all required NCOALink output to the client. *STANDARD* is the default setting. • *MODIFIED*: The job returned output that was modified by one or more post processes. • *BOTH*: The job returned output that was modified by one or more post processes. The job also returned a separate file that contained all of the required output.

Option	Description
Post Processes Performed	Specifies whether you processed the list after NCOALink processing. Select *YES* or *NO*.
Post Processed Data Modified	Specifies whether post processes performed after NCOALink processing included postal or non postal changes to the data. • *NO*: Post processing performed after NCOALink processing did not include changes to the data. *NO* is the default setting. • *FROM_POSTAL_DATA*: Post processing performed after NCOALink processing included postal changes to the data. • *FROM_NON_POSTAL_DATA*: Post processing performed after NCOALink processing included non postal changes to the data. • *FROM_BOTH*: Post processing performed after NCOALink processing included both postal and non postal changes to the data.
Postcode For Mail Entry	Specifies the ZIP Code of the Business Mail Entry Unit (BMEU) or post office where you submitted the mail for mailing.
Pre Processes Performed	Specifies whether you performed processes on the list before you performed NCOALink processing. Select *YES* or *NO*.
Pre Processed Data Modified	Specifies whether preprocesses performed before NCOALink processing updated the data with postal or non postal data. • *NO*: Preprocessing performed before NCOALink processing did not include changes to the data. *NO* is the default setting. • *FROM_POSTAL_DATA*: Preprocessing performed before NCOALink processing included postal changes to the data. • *FROM_NON_POSTAL_DATA*: Preprocessing performed before NCOALink processing included non postal changes to the data. • *FROM_BOTH*: Preprocessing performed before NCOALink processing included both postal and non postal changes to the data.

3.5.12.5.6 USPS NCOALink contact detail options

Complete the Contact Detail List options with information about the NCOALink broker, list administrator, or list owner.

TheContact Detail List options are not required for end users. Enter information about the list provider. A list provider can be any of the following:

- Broker: Directs business to the service provider.
- List Administrator: Stores and maintains address lists.
- Owner: Processes lists with no involvement from a broker or list administrator.

Contact Detail List option descriptions

Option	Description
Address	Specifies the address for the list provider.
Contact Level	Specifies the degree of separation the list provider is from you. Enter a value from 1 through 99. **Example** Enter 1 if you received the list from your broker. Enter a 2 if your broker received the list from a list administrator. **Note** The transform does not use this information in any log file.
Contact Company Website	Optional. Specifies the website of the list provider.
Locality	Specifies the locality (city name) of the list provider.
Type	Specifies the type of list provider. • *BROKER*: List provider is a broker, who directs business to the service provider. • *LIST_ADMINISTRATOR*: List provider is a list administrator, who stores and maintains address lists. • *OWNER*: List provider is the owner, who processes the list with no involvement from a broker or list administrator.
License Assigned ID	Specifies an identification that you assign to the list provider. Enter six characters.

Option	Description
NAICS Code	Specifies the numeric North American Industry Classification System code for the list provider. The NAICS code identifies the business in which the list provider engages. For more information about NAICS codes, see http://www.census.gov/epcd/www/naics.html .
Name	Specifies the name of the list provider.
PAF Signing Date	Specifies the date the list provider signed their PAF. Use the following format: yyyy/mm/dd.
Phone	Specifies the phone number of the list provider.
Postcode1	Specifies the Postcode1 (ZIP Code) for the list provider.
Postcode2	Specifies the Postcode2 (ZIP+4) for the list provider.
Region	Specifies the Region (State) for the list provider.

3.5.12.6 USA Regulatory Address Cleanse assignment options

Set options in the Assignment Options group to enable the USPS processes that the transform performs on your data.

Assignment group option descriptions

Option	Description
Enable DPV	Specifies whether to perform DPV processing. • *YES*: Enables DPV processing. • *NO*: Disables DPV processing.
Enable eLOT	Specifies whether to perform eLOT processing. • *YES*: Enables eLOT processing. • *NO*: Disables eLOT processing.

Option	Description
Enable EWS	Specifies whether to perform EWS processing. • *YES*: Enables EWS processing. • *NO*: Disables EWS processing. If the transform cannot make an exact match using the `zip4us.dir` (Address Directory 1) directory, 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.
Enable LACSLink	Specifies whether to perform LACSLink processing. • *YES*: Enables LACSLink processing. • *NO*: Disables LACSLink processing.
Enable NCOALink	Specifies whether to perform NCOALink processing. • *YES*: Enables NCOALink processing. • *NO*: Disables NCOALink processing.
Enable RDI	Specifies whether to perform RDI processing. • *YES*: Enables RDI processing. • *NO*: Disables RDI processing.
Enable Reverse Soundex Search	Specifies whether to use the `zip4us.rev` (Reverse Soundex) directory in an attempt to make address assignments. • *YES*: Enables Reverse Soundex. • *NO*: Disables Reverse Soundex.
Enable SuiteLink	Specifies whether to perform SuiteLink processing. • *YES*: Enables SuiteLink processing. • *NO*: Disables SuiteLink processing.

Option	Description
Geo Mode	Specifies the type of GeoCensus processing to perform. • *ADDRESS*: Processes address-level GeoCensus only. • *BOTH*: Attempts an address-level GeoCensus assignment first, if applicable. If it can't make the GeoCensus assignment, attempts a centroid-level GeoCensus assignment if applicable. • *CENTROID*: Processes centroid-level GeoCensus only. • *NONE*: Disables GeoCensus processing. Choose this option if you have not purchased the GeoCensus option or if you do not want to perform GeoCensus processing. **i Note** Use the Geocoder transform instead of this option. We plan to deprecate the GeoCensus functionality in the USA Regulatory Address Cleanse transform in the future. Upgrade any data flows that currently use the GeoCensus functionality to use the Geocoder transform. For instructions on upgrading from GeoCensus to the Geocoder transform, see the *Upgrade Guide*.
Dual Address	Specifies the action to take when the transform encounters a dual address. • *POSITION*: Selects an address based on the arrangement of the input data. The transform attempts to validate the address that is closest to the lower left corner of the address block. That address might be the postal address (rural route or PO Box) or the street address based on how the data appears in the address. • *POSTAL*: The transform attempts to validate based on the postal address. If that fails, the transform attempts to validate based on the street address. • *STREET*: The transform attempts to validate based on the street address. If that fails, the transform attempts to validate based on the postal address.
Enable DSF2 Augment	Specifies whether to perform Delivery Sequence File Second Generation (DSF2) augment processing. Select *YES* or *NO*.

Option	Description
USPS Certification Testing Mode	Specifies the type of certification to use for processing so the software checks for the appropriate settings and issues warnings and errors when applicable. • *NONE*: Processes without any special settings for certification. *NoneNONE* is the default setting. • *CASS*: Processes with the appropriate settings for CASS self certification. • *DSF2_AUGMENT*: Processes with appropriate settings for DSF2 augment certification. **i Note** When certifying for NCOALink, set the testing mode in the *NCOALink* > *Processing Options* > *List Processing Objective* option.

3.5.12.7 USA Regulatory Address Cleanse standardization options

Set options for standardizing your address data in the Standardization group of options.

Standardization group option descriptions

Option	Description
Standardize Assigned Address	Specifies whether to correct and standardize the assigned address line and lastline data. • *YES*: Corrects and standardizes address line and lastline data. Select *YES* for CASS certification. • *NO*: Does not standardize your address line or lastline data.
Standardize Unassigned Address	Specifies whether to standardize unassigned data. • *YES*: Attempts to parse and standardize any unassigned addresses. • *NO*: Leaves unassigned addresses as entered on input.

Option	Description
Use USPS Primary Name Abbreviation	Specifies whether to abbreviate the primary name in a multiline or standardized last line field so that the entire output data does not exceed 30 characters. The abbreviation applies only when the multiline or standardized last line output data exceeds 30 characters. If the output address line is already 30 characters or less, the transform does not abbreviate the primary name. • *YES*: Enables the transform to use the USPS form of the primary name abbreviation when available in the directory. > **i Note** > Also affects the following multiline and standardized last line field styles: > • *Suffix Style*: Uses Short style > • *Directional Style*: Uses Short style > • *Address Line Alias*: Overrides the *PRESERVE* setting. • *NO*: Disables this option. When this option is set to *YES*, the software outputs an address line with more than 30 characters when there is no abbreviated form of the address in the USPS directory. If you require your data to fit exactly into 30 characters, we recommend setting the appropriate address output field lengths to 30. **Intelligent truncation**: When the number of characters in the output is greater than the length specified for the output field, the software uses intelligent truncation to fit the data in the field without eliminating vital address data. Using intelligent truncation, the software abbreviates data in the following ways: • The software uses the USPS abbreviation for the primary name when enabled. • The software uses the USPS abbreviation for the locality name when enabled. • If the resulting output data does not fit the field, the software truncates the data. Intelligent truncation is automatic and cannot be controlled with a setting.

Option	Description
Use USPS Locality Abbreviation	Specifies whether to use the USPS 13-character city name when available and when the full city name exceeds 13 characters. Applicable only when you use the multiline and standardized last line fields. • *YES*: Uses the USPS 13-character city name when the full city name exceeds 13 characters for multiline and standardized last line fields. When the input city name is a non mailing city name, or it is not valid, the software relies on other settings in the job to determine what to output for city. • *NO*: Disables this option. **Intelligent truncation**: When the number of characters in the output is greater than the length specified for the output field, the software uses intelligent truncation to fit the data in the field without eliminating vital address data. Using intelligent truncation, the software abbreviates data in the following ways: • The software uses the USPS abbreviation for the primary name when *Use USPS Primary Name Abbreviation* is enabled. • The software uses the USPS abbreviation for the locality name when *Use USPS Locality Abbreviation* is enabled. • If the resulting output data does not fit the field, the software truncates the data. Intelligent truncation is automatic and cannot be controlled with a setting.
Capitalization	Specifies the casing of your address data. • *LOWER*: Converts data to all lowercase letters. For example, "Main Street South" becomes "main street south." • *MIXED*: Converts data to initial capital letters. For example, "MAIN STREET SOUTH" becomes "Main Street South." • *UPPER*: Converts data to all capital letters. For example, "Main Street South" becomes "MAIN STREET SOUTH." **i Note** For consistent casing, ensure that this option and the *Capitalization* setting in the Data Cleanse transform are the same.

Option	Description
Directional Style	Specifies whether to abbreviate directional data. • *LONG*: Uses fully spelled directionals such as "North," "South," "East," and "West." • *PRESERVE*: Preserves the style used in the input record. • *SHORT*: Uses abbreviated directionals such as "N," "S," "E," and "W." **i Note** When the *Use USPS Primary Name Abbreviation* option is set to *YES*, the software overrides a setting of *LONG* and *PRESERVE* for the *Directional Style* option and outputs the short directional style.
Primary Type Style	Specifies whether to abbreviate the street (primary) type. • *LONG*: Uses fully spelled primary types such as Street, Avenue, and Road. • *PRESERVE*: Preserves the style used in the input record. • *SHORT*: Uses abbreviated primary types such as St, Ave, and Rd. **i Note** When *Use USPS Primary Name Abbreviation* is set to *YES*, the software overrides a setting of *LONG* and *PRESERVE* for the *Primary Type Style* option and outputs the short primary type style.
Unit Description	Specifies how to standardize the unit description. • *CONVERT*: Uses the unit description found in the postal directory, such as an apartment, suite, room, or floor. • *PRESERVE*: Preserves the unit description from the input record and corrects any spelling errors.

Option	Description
Retain Pound Sign in Unit Description	Specifies whether the software outputs the pound sign (#) to the *Unit_Description* output field when it is present on input. • *YES*: Outputs the pound sign (#) unit designator to the *UNIT_DESCRIPTION* output field. • *NO*: Outputs the pound sign (#) unit designator to the *EXTRANEOUS_SECONDARY_UNIT_NUMBER* or the *EXTRANEOUS_SECONDARY_NON_POSTAL* output fields, or to both fields. The option does not affect the following address situations: • Puerto Rico addresses • Military addresses • Rural Route addresses • Addresses without "#" in the address line • Addresses with remainder words
Append Private Mailbox	Specifies whether the transform appends private mailbox data (PMB) to the address in the same field. • *YES*: Appends the PMB with the address in the same field. • *NO*: Outputs the PMB into the *NON_POSTAL_SECONDARY_ADDRESS*, *NON_POSTAL_UNIT*, and the *NON_POSTAL_UNIT_NUMBER* output fields. A PMB is like a post office box except that it is hosted by a private company.
Preserve Dual Address Order	Specifies whether to preserve or change the format of dual addresses. • *YES*: Preserves the order when an input address contains both a street and mailing address. • *NO*: Changes the order of input dual address lines. When the input address contains both a locality and mailing address, the transform moves the assigned address immediately above the locality and region.

Option	Description
Address Line Alias	Specifies whether to convert an input address line alias on input to the preferred USPS address line on output. • *CONVERT*: Converts an input address line alias on input to the preferred USPS format on output. > **i Note** > To be compliant with CASS, set up your jobs to return the USPS preferred address. When you set *Address Line Alias* to *CONVERT*, the software returns the USPS-preferred address, even when the input record has a base address or an alias address. The software does not produce a USPS 3553 when you select *PRESERVE* for *Address Line Alias* . • *PRESERVE*: Does not convert the input address line alias to the preferred USPS format, and outputs it as it was input. This option is not CASS compliant. The software does not produce a USPS 3553 form when you select this option.
Preserve Place Names	Specifies whether to preserve or change non-mailing city names (place names). • *YES*: Preserves the non-mailing city name. For example, when an input place name is Hollywood, the transform outputs Hollywood. • *NO*: Changes non mailing city names to city names preferred by the USPS. For example, when an input place name is Hollywood, the transform outputs Los Angeles. > **i Note** > When the *Use USPS Locality Abbreviation* option is set to *YES*, *Preserve Place Names* is set to *YES*, and *Assign With Input Locality* is set to *NO*, the software may not preserve some place names over 13 characters and abbreviates them.

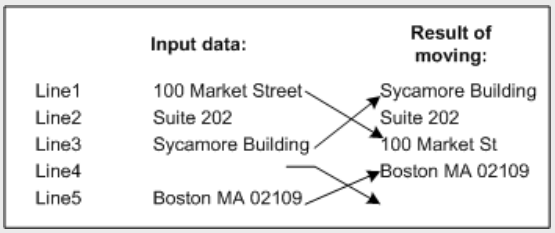
Option	Description
Assign With Input Locality	Specifies whether to use the last-line index when assigning the locality (city) name. • *YES*: Assigns the Locality1 based on the locality name that is input if it is valid for the Postcode 1. Does not change the Locality1 based on last line index. • *NO*: Assigns the Locality1 based on the locality that is input if it is valid for the Postcode 1 and not a place name; otherwise, assigns Locality1 based on the last-line index of the address line. Produces a more geographically true Locality1. If you choose *NO*, the value you choose for the *Preserve Place Names* option does not matter; the software converts place names. **i Note** When the *Use USPS Locality Abbreviation* option is set to *YES*, *Preserve Place Names* is set to *YES*, and *Assign With Input Locality* is set to *NO*, the software may not preserve some place names over 13 characters and abbreviates them.

Option	Description
Include Unused Address Line Data	Specifies whether to output the unused address line data for discrete and multiline fields. • *YES*: Includes the unused address line data, including invalid secondary information, in the address line and in the *ADDRESS_LINE_REMAINDER1* field. • *NO*: Does not output the unused address line data in the address line but outputs the information to certain output fields.

Example

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 2 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
Add Firm Match Secondary	Specifies whether the software adds secondary address information when the address matches an address in the SuiteLink directory that contains secondary address information. • *YES*: Adds secondary address information obtained from SuiteLink directories to the address line. • *NO*: Does not add secondary address information obtained from SuiteLink directories to the address line, but includes SuiteLink-found information reflected in the lastline ZIP+4 Code and in other output fields. **i Note** Select *NO* if you are certifying for CASS and you do not want to update address lines in your data with the SuiteLink secondary information. 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. There are instances when the software ignores the *NO* setting for *Add Firm Match Secondary*. For example, the software ignores the setting when it does not find a secondary address match in the National directories, but finds an exact match to the SuiteLink secondary address. The software does not consider the SuiteLink secondary address as a change to the input secondary address. The software also does not consider the SuiteLink secondary address as a change to the input secondary address when it finds an exact match to the SuiteLink secondary unit and or range.

Option	Description
Move Multiline Data	Specifies whether the transform rearranges multiline data to conform to USPS guidelines, and the method for arranging blank lines. • *NO*: Keeps input multiline data in input order. • *BOTTOM*: Moves the primary address into position above the locality, region, postal code line, or lastline. Moves any blank lines to the top of the address, and shifts the data to the bottom of the block. • *TOP*: Moves the primary address into position above the locality, region, postal code line, or lastline. Moves any blank lines to the bottom of the block, and shifts the data to the top. You do not have to standardize your data to enable this option. **Example** when you select *BOTTOM*:  When you choose *TOP* or *BOTTOM*, ensure that the length of all input fields mapped to Multiline fields are the same length. For example, When you set *MULTILINE1* to 60, set all other Multiline fields to 60.
Combine Multilines	Specifies what to do with related fields input on separate lines. • *YES*: Combines related input fields from separate lines onto the same line. • *NO*: Does not combine input fields on separate lines.

Option	Description
Multiline Update Postcode 1	Specifies whether the transform assigns input postcode1 data with the USPS postcode data and outputs to Multiline. Affects data passed in and retrieved through multiline fields. • *DONT_UPDATE*: Assigns Postcode 1 to input, but does not output to Multiline. Outputs the original Postcode 1 and outputs the assigned Postcode 1 to other output fields. • *ERASE_THEN_UPDATE*: Assigns Postcode 1 and outputs the assigned Postcode 1 to the Multiline output fields. If the transform cannot assign input Postcode 1, does not output the original Postcode 1 to Multiline output fields. • *UPDATE*: Assigns Postcode 1 and outputs the assigned Postcode 1 to the Multiline output fields. Retains the input Postcode 1 when it does not assign Postcode 1.
Multiline Update Postcode 2	Specifies whether the transform assigns input postcode2 data with the USPS postcode data and outputs to Multiline. Affects data passed in and retrieved through multiline fields. • *DONT_UPDATE*: Assigns Postcode 2 to input, but does not output to Multiline. Outputs the original Postcode 2 and outputs the assigned Postcode 2 to other output fields. • *ERASE_THEN_UPDATE*: Assigns Postcode 2 and outputs the assigned Postcode 2 to the Multiline output fields. If the transform cannot assign input Postcode 2, does not output the original Postcode 2 to Multiline output fields. • *UPDATE*: Assigns Postcode 2 and outputs the assigned Postcode 2 to the Multiline output fields. Retains the input Postcode 2 when it does not assign Postcode 2.

3.5.12.8 USA Regulatory Address Cleanse non-certified options

The Non-Certified options group contains settings that, when enabled, create a mailing list that is not compliant with CASS rules.

Non-Certified group option descriptions

Option	Description
Disable Certification	Specifies whether the transform performs address cleansing without the restrictions of CASS certification rules. • *YES*: Disables CASS certification. Select *Yes* when you enable any of the other options in the Non-Certified group. With certification disabled, you can use the national directory for 14 months after the directory creation date. For CASS certified mailings, you can use the national directory only for 3 or 4 months after the creation date. • *NO*: Performs address cleansing underCASS certification rules. Ignores any enabled options in the Non-Certified group.
Assign With Input Postcode	Specifies whether the transform should use the last four digits of the input 9-digit post code to make a finer level of assignment than it could make under CASS rules. The last four digits of the 9-digit post code must be unique to a valid firm or secondary address. **i Note** The last four digits are usually the Postcode2, which is the ZIP+4. • *YES*: Enables this option. • *NO*: Disables this option. The transform ignores the last four digits of the input 9-digit postcode and makes a CASS certified assignment.
Accept Inexact Postcode Move	Specifies whether the transform ignores non matching elements between the input data and the national directory. Transform uses the built-in matching thresholds to determine that the record matches the record in the national directory. Applicable when an input record has an obsolete postcode or a postcode move. • *YES*: Enables this option. • *NO*: Disables this option.

Option	Description
Assign Postcode2 Not DPV Validated	Specifies whether to output the Postcode2 data from the national directory even when you make the following settings: *Enable DPV* in the Assignment Options group set to *NO* *Disable Certification* in this group to *YES* • *YES*: The transform does not validate the address using DPV. • *NO*: The transform leaves the output field *Postcode2* blank when it makes an assignment, and you set *Enable DPV* to *NO*.
Enable Parse Only	Specifies whether the transform only parses your data into their discrete components, and does not validate data against the postal directories. • *YES*: Parses records into their discrete components, but does not perform a lookup in the postal directories. Processing is faster, but the parsed results are not verified against the postal directories. • *NO*: Parses records into their discrete components and performs a lookup in the postal directories. Processing is slower, but the parsed results are verified against the postal directories.

Option	Description
Enable Geo Only	Specifies whether the transform processes with only the address-level GeoCensus directories instead of the centroid level GeoCensus directories. **i Note** Before you set this option, consider using the Geocoder transform instead. We plan to deprecate the GeoCensus functionality in the USA Regulatory Address Cleanse transform in the near future. We recommend that you upgrade any data flows that currently use the GeoCensus functionality in the USA Regulatory Address Cleanse transform to use the Geocoder transform. For instructions to upgrade your jobs to use the Geocoder transform, see the *Upgrade Guide*. • *YES*: Processes address cleansing using the GeoCensus directories only. The transform does not use the postal directories. Ensure that you define the GeoCensus directory location in the *Reference Files* option group. Ensure that you include the appropriate GeoCensus output fields. The transform does not post GeoCensus data to output fields other than GeoCensus output fields. • *NO*: Processes address cleansing using the GeoCensus directories and the postal directories.
Enable Suggestion Lists	Specifies whether the software uses suggestion lists for records that do not assign. For transactional processes only. • *YES*: Generates suggestion lists for addresses that do not assign so that you can choose an address to output. *NO*: Does not generate suggestion lists for unassigned addresses.

3.5.12.9 USA Regulatory Address Cleanse USPS CASS Report Options

Complete the *CASS Report Options* group with USPS Form 3553 information as required by CASS.

CASS Report Options descriptions

Option	Description
List Name	Specifies the name of the mailing list (up to 20 characters).
List Owner	Specifies the name of your company (up to 19 characters).

Option	Description
Mailer Address 1	Specifies the name and address of the person or organization for whom you are preparing the mailing (up to 29 characters per line).
Mailer Address 2	
Mailer Address 3	
Mailer Address 4	
Company Name Certified	Specifies the name of the company that has the end-user CASS certification. If you rely on SAP as your CASS-certified company, leave this option blank. The software inserts "SAP" as the default value. If you have your own CASS end-user certification, enter your company name, up to 40 characters.
Software Version	Specifies the version number of the software that is certified to process CASS-certified mailing lists. If you rely on SAP as your CASS-certified company, leave this option blank. The software inserts the applicable software name and version number as the default value. If you used your own software for CASS certification, enter the applicable software name and version number.
LOT Certification	Specifies whether you or your company have LOT certification. • *YES*: LOT certification is in your or your company name, but not CASS certification. *NO*: CASS certification is in your or your company name, but not LOT certification. ℹ Note Setting *LOT Certification* to *NO* ensures that the LOT certification lines on the Form 3553 are blank.

3.5.12.10 USA Regulatory Address Cleanse suggestion list options

The *Suggestion List* group contains options to configure how the software outputs suggestions lists.

Suggestion List group option descriptions

Option/Option group	Description
Max Number Lastlines	Specifies the maximum number of lastline suggestions that the transform generates. Maximum is 15. **i Note** If you set a low maximum, a viable suggestion could be left out of the suggestion list. **Example** • Use the setting to limit the size of the SOAP documents that the Web service sends. • Use the setting to limit the maximum number of suggestions that your users choose from.
Max Number Addresslines	Specifies the maximum number of address-line suggestions that the transform generates. Maximum is 100. **i Note** If you set a low maximum, a viable suggestion could be left out of the suggestion list. **Example** • Use the setting to limit the size of the SOAP documents that the Web service sends. • Use the setting to limit the maximum number of suggestions that your users choose from.
Lastlines Match Minimum	Specifies the minimum similarity score the transform requires for lastline suggestions. From 0 to 80. A suggestion list candidate must meet the minimum threshold before the software presents it as a suggestion for the input lastline. A lower number may produce more suggestions.
Address Lines Match Minimum	Specifies the minimum similarity score the transform requires for address line suggestions. From 0 to 80. A suggestion list candidate must meet the minimum threshold before the software presents it as a suggestion for the input address line. A lower number may produce more suggestions.

Option/Option group	Description
Combine Overlapping Ranges	Specifies how the transform consolidates individual suggestions that have overlapping ranges. • *COMBINE_IGNORING_GAPS*: Ignores gaps in ranges and overlaps the ranges. Selecting this option causes the transform to be more aggressive when overlapping ranges. • *COMBINE_PRESERVING_GAPS*: Preserves gaps in primary ranges, but still consolidates overlapping ranges. • *NONE*: Does not consolidate overlapping ranges.
Address Range Window	Specifies a range around the primary address in which the transform selects suggestions. Use this option to limit the suggestions to be within a few blocks of your input. **Example** You set this option to 500. Input address is: 1000 Pine St. The software returns suggestions in a ranged from 750 to 1250 Pine Street. Enter "0" so there is no limit to the ranges that the transform returns.
Match Range	Specifies whether the transform disregards an address-line suggestion when it does not match the primary range of the input address. • *YES*: Returns address-line suggestion only when it matches the primary range of the input address. • *NO*: Returns address-line suggestion when it doesn't have the same primary range as the input.
Style	Specifies the style of the output file. • *DELIMITED*: Outputs the suggestion list data in a delimited text format. Specify delimiter to use in the *Delimiter* and *Field Delimiter* options. • *XML*: Outputs the suggestion list data as hierarchical XML. Specify *XML* when you integrate suggestion lists through the Web service. Then use the XML tools that you own to parse the suggestion list data.
Delimiter	Specifies the delimiter to use between each suggestion for a delimited style suggestion list output file. Applicable when you chose *Delimited* for the *Style* option. Choose any character or string to separate each suggestion. Ensure that this value differs from the value you choose for the *Field Delimiter* value.

Option/Option group	Description
Field Delimiter	Specifies the delimiter to use between each suggestion list. Applicable when you chose *Delimited* for the *Style* option. Ensure that this value differs from the value you choose for the *Delimiter* value.

Related Information

Similarity score
Methods to extract Data Quality XML strings

3.5.12.11 USA Regulatory Address Cleanse suggestion list components

Complete options in the *Suggestion List Components* to specify the address field components to include in the *Suggestion_List* output field.

> **i Note**
>
> When you leave a component blank and you set the suggestion list *Style* option to *XML*, the transform does not output a value to the *Suggestion_List* output field.

In the following options, select *Yes* to include the item in the *Suggestion_List* output field.

Suggestion List Components option descriptions

Option/option group	Description
Selection	Specifies whether the transform outputs the selection number for multiple suggestions.
Locality1	Specifies whether the transform outputs the Locality1 preferred by the USPS. Applicable for primary, secondary, and lastline address levels.
Region1	Specifies whether the transform outputs the state, province, territory, or region. Applicable for primary, secondary, and lastline address levels.
Postcode	Specifies whether the transform outputs the five-digit Postcode1, excluding the ZIP4. Applicable for primary, secondary, and lastline address levels.

Option/option group	Description
Primary Number Low	Specifies whether the transform outputs the low portion of the primary number range. Applicable for primary and secondary address levels.
Primary Number High	Specifies whether the transform outputs the high portion of the primary number range. Applicable for primary and secondary address levels.
Primary Prefix1	Specifies whether the transform outputs the abbreviated directional, such as N, S, NW, and so on, that precedes a street name. Applicable for primary and secondary address levels.
Primary Name1	Specifies whether the software outputs the street name description. Applicable for primary and secondary address levels. **i Note** The transform does not output high or low primary numbers to this field.
Primary Type1	Specifies whether the transform outputs the abbreviated street type, such as St, Ave, or Pl. Applicable for primary and secondary address levels.
Primary Postfix1	Specifies whether the transform outputs the abbreviated directional, such as N, S, NW, and so on, that follows a street name. Applicable for primary and secondary address levels.
Primary Name Full1	Specifies whether the transform outputs the primary address line, such as the street address or post office box. The output includes the following information: Primary Prefix1, Primary Name1, Primary Type1, and Primary Postfix1. Applicable for primary and secondary address levels. **i Note** The transform does not output high or low primary numbers to this field.
Postcode2 Odd	Specifies whether the transform outputs the four-digit ZIP4 code, odd numbers only. Applicable for primary and secondary address levels.
Postcode2 Even	Specifies whether the transform outputs the four-digit ZIP4 code, even numbers only. Applicable for primary and secondary address levels.

Option/option group	Description
Primary Side Indicator	Specifies whether the transform outputs “Odd” or “Even” for the primary side indicator. Applicable for primary and secondary address levels.
Firm	Specifies whether the transform outputs the firm name for the secondary address.
Unit Description	Specifies whether the transform outputs the unit description, such as #, Apartment, or Flat. Applicable for secondary address level.
Unit Number Low	Specifies whether the transform outputs the low portion of the unit number range. Applicable for secondary address level.
Unit Number High	Specifies whether the transform outputs the high portion of the unit number range. Applicable for secondary address level.
Secondary Side Indicator	Specifies whether the transform outputs “Odd” or “Even” for the secondary side indicator. Applicable for secondary address level.

3.5.12.12 USA Regulatory Address Cleanse USPS Z4 Change options

Set USPS Z4 Change options to enable Z4Change processing and to specify the last time the Postcode 2 was updated.

Z4 Change Options group descriptions

Option	Description
Enable Z4 Change	Specifies whether the transform performs Z4Change processing. • *YES*: Performs Z4Change processing. • *NO*: Does not perform Z4Change processing.

Option	Description
Last ZIP4 Assign Date	Specifies the month and year that you most recently updated the mailing list with ZIP4 information, either through a full address correction process or a previous Z4Change pass. Enter the date using the format MM/YYYY. The transform verifies that the date is within the 12-month period covered by the Z4Change file. If there is a date problem, the transform issues an execution error message.

3.5.12.13 USA Regulatory Address Cleanse input fields

The USA Regulatory Address Cleanse transform has recognized input fields to use in field mapping.

USA Regulatory Address Cleanse input field descriptions

Field name	Description
ADDRESS_LINE	Contains the delivery address line that includes information like the house number, street name, and unit information.
CHECK_DIGIT	Contains the check digit for an eleven-digit delivery-point bar code. Applicable only when the transform makes a full assignment. The transform provides the check digit for a 5-digit bar code when a 5-digit assignment is possible or when the address is undeliverable. When the address is unassigned, the transform basis the check digit on the unverified input Postcode1 (ZIP Code.)
COUNTRY	Contains the country name. **i Note** The USA Regulatory Address Cleanse transform processes only USA. Therefore, it does not attempt to make an assignment for addresses outside of the USA, USA possessions, territories, and protectorates.
COUNTY_CODE	Contains the three-digit county code. Numbers start at 001 within each state.

Field name	Description
DATA_SOURCE_ID	Contains an identification for the input source or list. Use this field to identify the source of an input set. If there are multiple lists on input, use this field to identify the list that an input record belongs to. The transform generates statistics for each unique value in this field when you include the field in your mapping and enable the *Gather Statistics Per Data Source* option in the *Report and Analysis* group.
DELIVERY_POINT	Contains the two-digit DPBC code.
FAMILY_NAME1	Contains the family name, such as Smith or Jones.
FIRM	Contains the company name.
GIVEN_NAME1	Contains the given name, such as Robert or Catherine.
GIVEN_NAME2	Contains the second given name, also known as middle name.
LASTLINE	Contains the last line delivery information that can include all or some of the following fields: • *LOCALITY1* • *REGION1* • *POSTCODE1* • *POSTCODE2*
LOCALITY1	Contains the city, town, or suburb.
LOCALITY2	Contains the urbanization information for Puerto Rico addresses.
LOT	Contains the Line of Travel information.
LOT_ORDER	Contains the one-character Line of Travel sortation order: • *A*: Ascending • *D*: Descending The transform requires LOT codes for non-automated Carrier Route (CART) presorting for Standard Mail and Enhanced Carrier Route (ECR) Subclass.
MULTILINE1-12	Contains a line of data from the input file. The data type can vary from record to record.
NAME	Contains the name of the person associated with the address.

Field name	Description
POSTCODE_FULL	Contains the complete postal code, either the ZIP10 with a hyphen or the ZIP9 without a hyphen.
POSTCODE1	Contains the five-digit primary ZIP Code. This field does not include the 4-digit ZIP4 Code.
POSTCODE2	Contains the four-digit portion of a ZIP Code, called ZIP4 or ZIP+4. The 4-digit ZIP Code is the code that follows the primary ZIP Code separated with a hyphen. For example, 54601-1234.
POSTNAME	Contains an honorary post name or a maturity post name. An honorary post name that indicates certification, academic degree, or affiliation. For example, CPA. A maturity post name indicates heritage. For example, Jr. and Sr.
PRENAME	Contains a prename, such as Mr., Ms., or Mrs.
REGION1	Contains the name of the state or province for this address.
SORTCODE_ROUTE	Contains the four-digit carrier route number.
STAGE_ADDRESS_FLAG *STAGE_LASTLINE_FLAG* *STAGE_NAME_FLAG* *STAGE_RECORD_KEY*	Contains information from the USPS stage file for NCOALink stage testing.
SUGGESTION_REPLY1-5	Contains an index number that corresponds to a specific last line suggestion or an address line suggestion. Optionally use these fields to input a street primary range or a street secondary range. With a value of 0, the transform ignores the suggestion list.
UNIT_NUMBER	Contains the secondary address information such as the unit description or secondary number.

3.5.12.14 USA Regulatory Address Cleanse output fields

Use the USA Regulatory Address Cleanse transform output fields for generated data and to add additional fields to your output.

USA Regulatory Address Cleanse output field descriptions

Field name	Description
ADDRESS_LINE	Contains the complete standardized primary and secondary address line. Style of suffixes, directional, and unit designators depends on how you define your options. **i Note** If the output values don't fit within the length of the output field, the transform uses intelligent truncation.
ADDRESS_LINE_REMAINDER1	Contains extraneous data that the transform finds in the address line that it cannot identify or that does not belong in a standardized address.

Field name	Description
ADDRESS_TYPE	Contains a two-character record-type indicator for the assigned address. Applicable for *DELIVERY* and *DUAL FIELD_ADDRCLASS*. The first character indicates the type of record in the address directory to which the address matched. The second character may be a D or is blank: • *D*: Default. If further input information was available on input, the transform could make a finer level of address assignment. • *blank*: Input information is sufficient to make the finest level of address assignment. The following list contains the first character with a D, and describes the meaning of default for the type of record: • *FD*: Firm default. The transform did not assign a firm-level Postcode2, but could do so if given more firm information. • *GD*: General delivery default. "General Delivery" is the only primary name listed for the Postcode1 in the postal directory. • *HD*: High-rise default. The transform assigned the Postcode2 for the entire building. If more information was available, the transform could make an assignment at the unit, floor, or wing level. For example, the transform assigns the Postcode2 based on the building because the input suite or apartment number is out of range. • *RD*: Rural route or highway contract default. The transform assigned the Postcode2 for the entire route but could make a finer assignment with the box number. • *SD*: Street default. The transform assigned a Postcode2 for the block because there is not enough information on input, or there is no existing Postcode2 for the building to make a finer level of assignment. • *UD*: Unique default. The transform assigned a unique Postcode1 because the owner of the Postcode1 did not provide Postcode2 assignments or the transform could not find a match in the directories. **i Note** When the transform cannot assign an address, it provides an address-type indication based on the way the input data is parsed. This process is not foolproof. The transform can indicate that it parsed a street, rural route, highway contract, general delivery, or PO Box.

Field name	Description
AGEO_COUNTYCODE	Contains the Federal Information Processing Standard (FIPS) county code. For example, 063.
AGEO_LATITUDE	Contains the latitude, degrees north of the equator, in the following format: 12.123456.
AGEO_LONGITUDE	Contains the longitude, degrees west of the Greenwich Meridian, in the following format: -12.123456.
AGEO_MCDCODE	Contains the U.S. Census Bureau Minor Civil Division (MCD) data. If MCD data is unavailable, contains Census County Division (CCD) data. For example, 40775.
AGEO_PLACECODE	Contains the Federal Information Processing Standard (FIPS) place code. For example, 40775.
AGEO_SECTIONCODE	Contains the U.S. census tract code in the format 1234567890. For example, 0003001059.
AGEO_STATECODE	Contains the Federal Information Processing Standard (FIPS) state code. For example, 55.
ALIAS_TYPE	Contains the alias-type indicator for the assigned address. Applicable for the output field attribute category *FIELD_ADDRCLASS*, type *OFFICIAL*. Alias type describes the input address, not the output address. • *A*: Input address matches an abbreviated street name. • *B*: Input address matches the high-rise alternate default base record. • *C*: Input street name is out of date; to get new street name, convert your record to the preferred alias. • *H*: Input address is an undesirable alternate that is subject to conversion to a USPS preferred street address (high-rise alternate). • *O*: Input address is a street nickname or other alias. • *P*: Input address is a preferred alias. • *Blank*: Input address is not an alias or is unassigned.

Field name	Description
ANKLINK_RETURN_CODE	Contains an ANKLink (attempted not known) return code. Valid values are: • *77*: Transform found an ANKLink match. If NCOA-Link_Return_Code contains *A*, *91*, or *92*, you may be able to obtain a new address from an NCOALink full service provider. • *Blank*: Transform did not perform an NCOALink lookup or did not fine an ANKLink match. ANKLink_Re-turn_Code is always Blank for full service providers.
AUDIT_DROPPED_SECONDARY	Contains the dropped secondary address information. Use this field for audit testing or when the transform makes an ANKLink match.
AUDIT _PRENAME *AUDIT _GIVEN_NAME1* *AUDIT _GIVEN _NAME2* *AUDIT_FAMILY_NAME1* *AUDIT _POSTNAME*	Contains the name data that the transform used to make an NCOALink match. In some cases, the name in these fields is not the same as the input name. For example, when the transform uses a nickname, alternate spelling, or initial. In the case of a firm match, these name fields contain a split version of the firm data. The transform also populates these fields when it makes an ANKLink match. Use these fields for audit testing.
AUDIT _GENDER	Contains the gender information. Use this field for audit test-ing or when the transform makes and ANKLink match.
AUDIT _GENERAL	Contains query data, result data, and hint bytes required by the USPS for Stage I and Stage II tests. Use this field for au-dit purposes only. This field is required for audits. The *AUDIT _GENERAL* field can also contain ANKLink return codes. For more information about this field, see the NCOALink User Technical Reference at https://postalpro.usps.com/NCOALink_User_Tech_Info .
AUDIT _PRIMARY_NAME	Contains the primary name that NCOALink used for match-ing. Required for audits.
AUDIT _RANGE	Contains the range that NCOALink used for matching. Required for audits.

Field name	Description
AUDIT _SECONDARY_RANGE	Contains the secondary range that NCOALink used for matching. Required for audits.
AUDIT _TRUNCATED_GIVEN_NAME1	Contains the truncated given name1. Required for audits.
AUDIT _TRUNCATED_GIVEN_NAME2	Contains the truncated given name2 or middle name that NCOALink used for matching. Required for audits.
AUDIT_UNIT	Contains the unit data that NCOALink used for matching. Required for audits.
CARRIER_ROUTE_SORT_ZONE	Contains the code that indicates the record eligibility for Standard Mail Automation Enhanced Carrier Route: • *A*: Eligible for carrier route rates. Merging is allowed. • *B*: Eligible for carrier route rates. Merging is not allowed. • *C*: Not eligible for carrier route rates. Merging is allowed. • *D*: Not eligible for carrier route rates. Merging is not allowed.
CASS_ASSIGNMENT_TYPE	Indicates the option that the transform used in making the assignment. • *O*: Non-CASS and DPV tie-break options were disabled or were not used to make an assignment. • *1*: Inexact Postcode1 move assignment. • *2*: Input Postcode2 assignment. • *3*: Used DPV tie-breaking to make assignment. • *Blank*: Transform could not assign input address.
CASS_RECORD_TYPE	Contains the record type necessary for posting on the CASS test. This field is populated for assigned records only. The valid record types include the following: • *F*: Firm • *G*: General delivery • *H*: High-rise • *P*: Post office box • *R*: Rural route or highway contract • *S*: Street

Field name	Description
CGEO_BSACODE	Contains the Core-Based Statistical Area (CBSA) code. A CBSA is a census-based area defined by the U. S. Office of Management and Budget. You can collect statistics for less urban areas of the country when you use CBSA. CBSAs cover approximately 90 percent of the entire U.S. population.
CGEO_LATITUDE	Contains latitude, degrees north of the equator, in the format 12.123456.
CGEO_LONGITUDE	Contains longitude, degrees west of the Greenwich Meridian, in the format -12.123456.
CGEO_METROCODE	Contains a Metropolitan Statistical Area (MSA) number, such as 3870. A value of 0000 indicates that address does not lie in any MSA, which is usually a rural area.
CGEO_SECTIONCODE	Contains a U.S. census tract and block group code. For example, 0202001067. The first six digits are the tract number, and the first of the final four digits is the block group code within the tract. The transform uses the Metropolitan Statistical Area (MSA) and the block group codes for matching to demographic-coding databases. To specify a unique census block group within the entire country, combine the *SORTCODE_POSTCODE* and *CGEO_SECTIONCODE* fields.
CHECK_DIGIT	Contains the check digit for delivery-point bar code. If the transform could not make a full postal code (ZIP+4) assignment, contains the check digit for a five-digit bar code.
COUNT	Contains the suggestion count generated as the result of looking up the current record. The count is a non-negative value. If the current record does not end processing with a suggestion list needing resolution, then the value in this field is 0.
COUNTY	Contains the county name.
COUNTY_CODE	Contains the Federal Information Processing Standard (FIPS) three-digit county code. Numbers are unique within states. If you prepare a presorted periodicals mailing, you might use county information.
COUNTY_NAME	Contains the full county name.

Field name	Description
DELIVERY_POINT	Contains the two-digit Delivery Point Bar Code (DPBC).
DELIVERY_TYPE	Contains the type of postal facility: • *A*: Airport Mail Facility (AMF) • *B*: Branch Office • *C*: Community Post Office (CPO) • *D*: Area Distribution Center (ADC) • *E*: Sectional Center Facility (SCF) • *F*: Delivery Distribution • *G*: General Mail Facility (GMF) • *K*: Network Distribution Centers (NDC) • *M*: Money Order Unit • *N*: City/place name • *P*: Post Office (main) • *S*: Station • *U*: Urbanization (Puerto Rico only)
DISTRICT	Contains the district number for the U.S. House of Representatives.
DPV_CMRA	Contains the DPV Commercial Mail Receiving Agency (CMRA) component that the transform generates for this record. • *L*: Address triggered DPV locking. • *N*: Address is not a CMRA. • *Y*: Address is a valid CMRA. • *Blank*: Blank output value indicates that *Enable DPV Validation* option is set to *No*, the software locked DPV processing, or the transform cannot assign the input address.
DPV_DOOR_NOT_ACCESSIBLE	Indicates that the address is a designated Door Not Accessible (DNA) address. • *Y*: Address is a DNA address. • *N*: Address is not a DNA address. • *blank*: *Enable DPV* is set to *NO*, `dpv_door_na.dir` is not present, or there is an assignment error. The USPS uses DNA indicators to mark addresses where the carrier cannot access the mail receptacle or door.

Field name	Description
DPV_FOOTNOTE	Contains the DPV footnotes that the USPS requires for end users for CASS certification. The footnotes contain the following information: • *AA*: Input address matches to the ZIP+4 file. • *A1*: Input address does not match to the ZIP+4 file. • *BB*: All input address field values match to DPV. • *CC*: Input address primary number matches to DPV, but secondary number does not match (the secondary is present but invalid). • *F1*: Input address matches a military address. • *G1*: Input address matches a general delivery address. • *M1*: Input address primary number is missing. • *M3*: Input address primary number is invalid. • *N1*: Input address primary number matches to DPV but the address is missing the secondary number. • *NL*: NCOALink move address cannot be DPV confirmed. NCOALink directory does not exactly match DPV directory data because the NCOALink directories are updated more frequently than the DPV directories. **i Note** The *NL* footnote is applicable for the *MOVE_UPDATED* field class for the output field attribute category *FIELD_CLASS*, type *MOVE_UPDATED*. • *P1*: Input address missing rural route or highway contract box number. • *P3*: Input address has invalid post office, rural route, or highway contract number. • *R1*: Input address matches to CMRA, but the secondary number is not present. • *RR*: Input address matches CMRA. • *U1*: Input address matches a unique address. **i Note** The transform always posts the DPV footnotes in the same order, and this field is not always 12 characters in length. Data Services combines footnotes to fully explain the DPV assignment.

Field name	Description
DPV_NON_DELIVERY_DAYS	Indicates whether there are days when there is no mail delivery. Values include the following: • Y: There are days when there is no mail delivery. • N: There are no days when there is no mail delivery. • *blank*: DPV is not enabled or both of the directories for non-delivery days are missing. The directories for non-delivery days are: • `dpv_non_delivery_days_flag.dir` • `dpv_non_delivery_days.dir`
DPV_NON_DELIVERY_DAY_SUN *DPV_NON_DELIVERY_DAY_MON* *DPV_NON_DELIVERY_DAY_TUE* *DPV_NON_DELIVERY_DAY_WED* *DPV_NON_DELIVERY_DAY_THU* *DPV_NON_DELIVERY_DAY_FRI* *DPV_NON_DELIVERY_DAY_SAT*	Contains a non-delivery day Indicator for the day of the week. Valid values: • *Y*: Indicates a non-delivery day. • *N*: Indicates that mail is delivered on that particular day but not on all days. • *Blank*: Indicates that the DPV option is disabled or both of the directory files for non-delivery days are missing. Directories for non-delivery days are: `dpv_non_delivery_days_flag.dir` and `dpv_non_delivery_days.dir`.
DPV_NO_SECURE_LOCATION	Indicates that the address has a No Secure Location (NSL) indicator. • *Y*: Address is a designated NSL. • *N*: Address is not a designated NSL. • *blank*: *Enable DPV* is set to *NO*, The NSL directory is not present, or there is an assignment error. The USPS uses NSL indicators to mark addresses where there is no security. For example: • There is no one at the address to collect the mail. • The storefront does not have a mail recepticle. • The business is closed on certain days. The directory for NSL is `dpv_no_secure_loc.dir`.

Field name	Description
DPV_NOSTATS	Contains a No Status (No Stat) indicator. The No Stat indicator tells whether the address is a vacant property, it receives mail as a part of a drop, or it does not have an established delivery yet. • *Y*: Address flagged as No Stat in DPV data. • *N*: Address is not a No Stat type address. • *Blank*: Address not looked up. **i Note** The US Addressing report contains DPV NoStats counts in the DPV Summary section.
DPV_STATUS	Contains a DPV status component that the transform generates for this record. • *D*: Primary range is a confirmed delivery point, but secondary range is not available on input. • *L*: Address triggered DPV locking. • *N*: Address is not valid delivery point. • *S*: Dropped secondary address information or dropped trailing alpha from primary address to match primary range to a confirmed delivery point. • *Y*: Address is confirmed delivery point. Primary range and secondary range (if present) are valid. • *Blank*: Either the *Enable DPV Validation* is set to *No*, the software locked DPV processing, or the transform cannot assign the input address.
DPV_THROWBACK_INDICATOR	Indicates that the address is a DPV PO Box Throwback. • *Y*: The address is a DPV PO Box Throwback. • *N*: The address is not a DPV Throwback. • *blank*: *Enable DPV* is set to *NO*, `dpv_po_box_throwback.dir` is not present, or there is an assignment error. The USPS uses the DPV PO Box Throwback indicator with both the DSF2 and DPV programs. The indicator identifies street addresses where the delivery point is a street address but mail delivery is to the customer PO Box address.

Field name	Description
DPV_VACANT	Contains a vacant address indicator. • *Y*: Address is vacant. • *N*: Address is not vacant. • *Blank*: Address not looked up. **i Note** The US Addressing report contains DPV Vacant counts in the DPV Summary section.
DSF2_BUSINESS_INDICATOR	Contains a residential or business indicator. • *Y*: Business address. • *N*: Not a business address. • *Blank*: Address not looked up. Use this information to lower your parcel-shipping costs. For example, some parcel delivery services charge more for delivery to residential addresses.
DSF2_DELIVERY_TYPE	Contains the delivery type: • *1*: Curb-side delivery. • *2*: NDCBU (Neighborhood Delivery Centralized Box Unit) delivery. • *3*: Central delivery. • *4*: Door slot delivery. • *Blank*: Address not looked up.
DSF2_DROP_COUNT	Contains the drop count. If the *DSF2_DROP_INDICATOR* field contains *Y* or the *DPV_CMRA* field contains *Y*, then this field contains a value from 000 to 999. The values 000 to 999 indicate the number of businesses or families served by the delivery point.
DSF2_DROP_INDICATOR	Contains the drop indicator: • *Y*: Delivery point serves multiple businesses or families. For example, delivery point may be a CMRA (Commercial Mail Receiving Agency). • *N*: Delivery address is not a CMRA. • *Blank*: Address not looked up.

Field name	Description
DSF2_EDUCATIONAL_IND	Contains the educational indicator: • *Y*: Address is an educational institution. • *N*: Address is not an educational institution. • *Blank*: Address not looked up.
DSF2_LACS_CONVERSION_INDICATOR	Contains the LACS (Locatable Address Conversion System) indicator: • *Y*: Address is LACS convertible. • *N*: Address is not LACS convertible. • *Blank*: Address not looked up.
DSF2_RECORD_TYPE	Contains the record type indicator: • *B*: Business address • *R*: Residential address • *U*: Unknown. AP.DSF_Del type is blank. `<Blank>`: No information available.
DSF2_SEASONAL_INDICATOR	Contains the seasonal address indicator: • *Y*: Address is seasonally occupied. • *N*: Address is not seasonal. • *Blank*: Address not looked up.
DSF2_THROWBACK_INDICATOR	Contains the throw-back indicator: • *Y*: Customer with street address wants delivery at PO Box instead. • *N*: No throw back necessary. • *Blank*: Address not looked up.
ERROR	Contains the error status code generated by the suggestion list process. • *0*: No suggestion selection errors. • *1*: Necessary selection information blank. For example, the transform generates a lastline suggestion list, but a lastline selection input field is missing data. • *2*: Suggestion selection invalid. For example, user selected 8 but there are only 5 suggestions. • *3*: Suggestion entry in input field invalid. • *4*: Suggestion range in input field invalid. • *5*: Suggestion secondary range in input field is invalid.

Field name	Description
EWS_Match	Contains the results of the EWS (Early Warning System) match. • *T*: True. Address in EWS directory and is EWS match. • *F*: False. Address not in EWS directory. • *Blank*: EWS not enabled.
EXTRA1-10	Contains non-address data above or below the address data in the address block. Available only for input data mapped to Multiline input fields.
EXTRANEOUS_SECONDARY_ADDRESS_DATA	Contains data from the *EXTRANEOUS_SECONDARY_UNIT_NUMBER* output field and the *EXTRANEOUS_SECONDARY_NON_POSTAL* output field respectively. The transform also places additional pound unit designator "#" data in the remainder or extra components, including invalid secondary address data.
EXTRANEOUS_SECONDARY_NON_POSTAL	Contains data that the transform determines is private mail box data. Determined based on position in the address line and other information such as a pound unit designator "#". Extraneous data may include invalid secondary address data.
EXTRANEOUS_SECONDARY_UNIT_NUMBER	Contains data that the transform determines is secondary range data based on position in the address line and other information in the address. Extraneous data may include invalid secondary address data.
FAULT_CODE	Contains a code that indicates why the transform cannot assign address. Field is blank when the transform assigns the address.
FAULT_OR_STATUS_CODE	Contains a fault code that indicates why the transform cannot assign address, or a status code that indicates how the input address differs from the assigned address.
Finance_Area_Postcode	Contains the lowest Postcode1 within a Finance Number. **i Note** Finance Numbers link data to a single post office or postmaster.

Field name	Description
FIRM	Contains a firm name from the postal directory. If firm name is not in the postal directory, contains the firm name from the input record. **i Note** Not applicable for input data in multilines. If the firm name is not in the postal directory, the transform cannot reliably identify the firm name from input data. The postal directory may contain firm names that are shortened or spelled differently. If you do not want to use the postal directory firm names on your mail piece, retrieve the firm name from your input data.
FOREIGN_CODE	Specifies whether the address is foreign or domestic. • *F*: Foreign address • *Blank*: Domestic (U.S.) address
FULL_ADDRESS	Contains a complete address line, including secondary address and dual address line data (street and postal). **i Note** If you set the *Include Unused Address Line Data* option in the *Standardization Options* group to *Yes*, may contain invalid secondary address information. **i Note** If an output value does not fit within the length of the output field, the transform truncates the data using intelligent truncation.
GEO_MATCHCODE	Contains a match code that indicates the precision of the latitude and longitude assignment. For all match codes, descriptions, and the precision levels, see "Geo match codes".
INTERMEDIATE_CODES	Contains codes required by the USPS for NCOALink certification and audit testing.

Field name	Description
LACSCODE	Contains LACS (Locatable Address Conversion System) indicator. • *T*: Address needs 9-1-1 conversion from box to street address and then submitted to a LACS vendor. • *F*: Address does not need conversion. • *Blank*: Address not assigned.
LACSLINK_INDICATOR	Contains the conversion status of addresses processed with LACSLink. • *Y*: Address converted by LACSLink and assigned a LACSLink_Return_Code of A. • *N*: Address looked up but not converted by LACSLink. • *F*: Address is false-positive. • *S*: Address converted by LACSLink but it was necessary to drop the secondary information. • *Blank*: No LACSLink lookup attempted.
LACSLINK_QUERY	Contains the pre LACSLink-converted address, or is blank when there is no attempted LACSLink lookup. Can contain a pound sign for the unit designator when the input address contains both a unit designator and a secondary unit. The software populates this field under the following circumstances: • You enable LACSLink in the transform. • The software attempts a LACSLink lookup. The Pre LACSLink-converted address is in the standard USPS Publication 28 format.
LACSLINK_RETURN_CODE	Contains the match status for LACSLink processing. • *A*: LACSLink record match and the address data fields contain a LACSLink address. • *00*: No LACSLink record match and no converted address. • *09*: LACSLink record match to an old address that is a "high-rise default" address. No new address provided. • *14*: LACSLink record match but could not convert to a deliverable address. • *92*: LACSLink record match only after dropping the secondary number from input address. • *Blank*: No LACSLink lookup attempted.

Field name	Description
LASTLINE	Contains the locality, region, and postal code on one line. **i Note** If an output value does not fit within the length of the output field, the transform truncates the data using intelligent truncation.
LOCALITY1	Contains the locality. • **Canada and USA:** Contains the locality preferred by the postal authority. • **Other countries:** Contains the city, town, locality, or suburb. **i Note** If an output value does not fit within the length of the output field, the transform truncates the data using intelligent truncation.
LOCALITY1_ALTERNATE	Contains the preserved input Locality1 when both of the following situations apply: • The postal authority does not recognize the Locality1 as valid for the input address line. • The transform changed the original Locality1 because of address assignment rules. Or, contains the default Locality1 for the assigned finance area under one of the following situations: • There is no input Locality1. • There is an input Locality1 but it is not valid in the assigned finance area.
LOCALITY1_LLIDX	Contains the Locality1 (city) name that is more geographically precise than the *LOCALITY1_OFFICIAL*. LLIDX (last-line index) is a USPS number that ties a ZIP+4 record to a particular city, state, and ZIP. **i Note** If an output value does not fit within the length of the output field, the transform truncates the data using intelligent truncation.

Field name	Description
LOCALITY1_NAME	Contains the City, town, or suburb name. The transform always preserves Locality1 names that are marked as invalid for mailing by the USPS, regardless of the values set for the *Preserve Place Name* and *Assign With Input Locality* options.
LOCALITY1_OFFICIAL	Contains the standardized locality name or converted locality name. The transform outputs a converted locality name when the input city name is a USPS-invalid city name, regardless of how you set the *Preserve Place Name* option.
LOCALITY1_OFFICIAL_ABBR	Contains the official USPS abbreviation for the city name, when available. The field is blank when the full city name is less than 13 characters, or the full city name is longer than 13 characters but there is no official USPS abbreviation.
LOCALITY2	Contains the additional city, town, locality, or suburb. **USA:** Contains the Puerto Rico urbanization name.
LOCALITY2_OFFICIAL	Contains the Puerto Rico urbanization name.
LOT	Contains the line of travel number.
LOT_ORDER	Contains the order in which the Line of travel is sorted. • *A*: Ascending • *D*: Descending
MATCHED_ADDRESSLINE_INDICATOR	Indicates if an address line matches a ZIP+4 record in the directories. • *T*: Address line matches a ZIP+4 record. • *F*: Address line does not match a ZIP+4 record.
MATCHED_LASTLINE_INDICATOR	Indicates if a last line matches a City—ZCF record in the directories. • *T*: Last line matches a City—ZCF record. • *F*: Last line does not match a City—ZCF record.

Field name	Description
MOVE_EFFECTIVE_DATE	Contains the date from NCOALink or ANKLink that indicates the effective date of a move. Output in the format yyyymm. Required for USPS audit. → Tip To use this date in a function or to post it to an output file, you may have to convert the date format to mm/dd/yyyy.
MOVE_TYPE	Contains an indicator that the move type is for a business, family, or an individual. The transform populates this field when the input matches with information in NCOALink or ANKLink directories. • *B*: Business moved, matched by company name. • *F*: Family move, matched by last name. • *I*: Individual move, matched by first and last name.
MULTILINE1-12	Contains any data type, which can vary from record to record. i Note May contain invalid secondary address information when you set the *Include Unused Address Line Data* option in the *Standardization Options* group to *Yes*. i Note If output is address data, and the output does not fit within the length of the output field, the transform truncates the data using intelligent truncation.
NAME	Contains the name of a person associated with the address.
NCOALINK_HINT_BYTE	Used or audit testing only.

Field name	Description
NCOALINK_RETURN_CODE	Contains the NCOALink return code. Also populated when the input record matches a record in ANKLink. To populate this field, set the *List Processing Mode* to one of the three available options: • *Change of Address* • *Statistics Only* • *Return Codes Only* The transform includes a brief description of the return codes in the NCOALink Processing Summary report. To include more detailed return code descriptions in the report, enable the *Generate Return Code Descriptions* option in the *NCOALink Report Options* group.
STAGE_TEST_RECORD	Contains values for stage testing for NCOALink, CASS, DSF2 Augment, DSF2 Sequence, and DSF2 Invoice self-certifications. The transform populates the values of this field automatically to match the format required for stage testing.
NON_CASS_FIRM	Contains the firm name resulting from processing missing or invalid firm information using the input ZIP+4.
NON_CASS_SECONDARY_ADDRESS	Contains the secondary address match resulting from processing missing or invalid secondary address with the input ZIP+4. **i Note** If output value does not fit within the length of the output field, the transform truncates the data using intelligent truncation.
NON_CASS_UNIT	Contains the unit designator match resulting from processing missing or invalid unit designator with the input ZIP+4.
NON_CASS_UNIT_NUMBER	Contains the unit designator match resulting from processing missing or invalid unit designator information with the input ZIP+4.
NON_POSTAL_SECONDARY_ADDRESS	Contains the complete non-postal secondary address. Non-Postal means that the mail is delivered through a private mailbox (PMB) company rather than the USPS. For example, PMB 10 or # 10.

Field name	Description
NON_POSTAL_UNIT	Contains the non-postal unit designator. Non-Postal means that the mail is delivered through a private mailbox (PMB) company rather than the USPS. For example, output of “PMB” or “#”.
NON_POSTAL_UNIT_NUMBER	Contains the non-postal secondary range, which is the private mailbox (PMB) number without the designator. Non-Postal means that the mail is delivered through a private mailbox company rather than the USPS.
PARSED_FIRM	Contains the firm name when the change of address is based on a firm name. Also applicable for ANKLink matches.
PO_BOX_ONLY_POSTCODE	Indicates whether the ZIP Code is listed in the postal directory as a ZIP Code for post office box delivery only. • *Y*: ZIP Code is for a PO Box only zone. • *N*: ZIP Code is not for a PO Box only zone. • *Blank*: ZIP Code is not assigned.
POSTAL_BOX_NUMBER	Contains the post office box number.
POSTCODE_FULL	Contains the complete ZIP10, which is the ZIP Code and the ZIP+4 separated with a hyphen. For example, 54601-1234.
POSTCODE_FULL_NO_HYPHEN	Contains the complete ZIP9, which is the ZIP Code and ZIP +4 without a hyphen. For example, 546011234.
POSTCODE_TYPE	Contains the assigned ZIP Code type. • *M*: Military • *U*: Unique. Specific to a university, large firm, or other institution. • *Blank*: Ordinary ZIP Code or the ZIP Code was not assigned.
POSTCODE1	Contains the 5-digit ZIP Code. Does not include the 4-digit ZIP+4.
POSTCODE1_CHANGE_IND	Indicates whether postal code realignment affected the address. • *T*: True. The transform corrected the postal code and the locality, if applicable. • *F*: False. The transform did not correct the postal code. • *Blank*: Address not corrected.

Field name	Description
POSTCODE2	Contains the four-digit, ZIP+4 Code. Located after the primary postal code on a mail piece that is either preceded with a hyphen or not. For example, for the full ZIP Code 54601-1234, the value is "1234."
PRE_SUITELINK_DELIVERY_POINT	Contains the 2-digit delivery point barcode assigned before the transform performs SuiteLink processing.
PRE_SUITELINK_POSTCODE1	Contains the ZIP Code assigned before the transform performs SuiteLink processing.
PRE_SUITELINK_POSTCODE2	The ZIP+4 for a high-rise default or street default record assigned before the transform performs SuiteLink processing.
PRE_SUITELINK_UNIT_DESCRIPTION	Contains the unit designator assigned before the transform performs SuiteLink processing. *Blank* when the transform did not assign secondary information.
PRE_SUITELINK_UNIT_NUMBER	Contains the secondary range information assigned before the transform performs SuiteLink processing. *Blank* when the transform did not assign secondary information.
PRIMARY_ADDRESS	Contains the primary address line, such as the street address or post office box. Does not include secondary address information such as apartment. If you enable the *Use USPS Primary Name Abbreviation* option, the transform uses the USPS Primary Name abbreviation first. **i Note** If output does not fit within the length of the output field, the transform truncates the data using intelligent truncation.
PRIMARY_NAME1	Contains the primary street name description. **i Note** If output does not fit within the length of the output field, the transform truncates the data using intelligent truncation.
PRIMARY_NUMBER	Contains the house or building number.
PRIMARY_POSTFIX1	Contains the abbreviated directional that follows the street name. For example, N, S, NW, or SE.

Field name	Description
PRIMARY_POSTFIX1_LONG	Contains the full spelled-out directional that follows the street name. For example, North or South.
PRIMARY_PREFIX1	Contains the abbreviated directional that precedes a street name. For example, N, S, NW, or SE.
PRIMARY_PREFIX1_LONG	Contains the full spelled-out directional that precedes a street name For example, North or South.
PRIMARY_SECONDARY_ADDRESS	Contains the primary address and secondary address on one line. Does not include remainder data. The software outputs this line as if the *Include Unused Address Line Data* option is set to *No*. When the *Include Unused Address Line Data* option is set to *No*, the output does not include invalid secondary address line information. **i Note** If output does not fit within the length of the output field, the transform truncates the data using intelligent truncation.
PRIMARY_TYPE1	Contains the abbreviated street type, such as St, Ave, or Pl.
PRIMARY_TYPE1_LONG	Contains the full spelled-out street type, such as Street or Avenue.
QSS_DEFAULT	Indicates whether the record qualifies as a default match instead of a higher level of assignment. • *T*: True. Record qualifies as a default match instead of a higher level of assignment. • *F*: False. Record does not qualify as a default match instead of a higher level of assignment.
RDI_INDICATOR	Indicates whether the address is residential. • *Y*: Residential address. • *N*: Nonresidential address.
REGION1	Contains the state, province, territory, or region.
RURAL_ROUTE_BOX_NUMBER	Contains the rural route box number.
RURAL_ROUTE_NUMBER	Contains the rural route number.
SECONDARY_ADDRESS	Contains the building name, floor, and room number in one field.

Field name	Description
SORTCODE_POSTCODE	Contains the Federal Information Processing Standards (FIPS) code for state and county. Combines the 2-digit state code with the 3-digit county code. **i Note** U.S. territories, possessions, or protectorates such as Puerto Rico, the U.S. Virgin Islands, or the Pacific Islands do not have FIPS state digits.
SORTCODE_ROUTE	Contains the four-digit carrier route.
STATUS	Specifies the suggestion status generated when the software looks up the record and processes suggestions. • *A*: Suggestion processing ended because an address suggestion list needed resolution. • *L*: Suggestion processing ended because a lastline suggestion list needed resolution. • *N:*: No suggestion list processing and no generated suggestion lists. • *R*: Primary range invalid for the selected address suggestion. • *S*: Secondary range invalid for the selected address suggestion. • *U:* Secondary address invalid for the selected address suggestion.
STATUS_CODE	Contains a code that indicates how the input address differs from the assigned address. *Blank* when the address is unassigned.
SUGGESTION_LIST	Contains a string of suggestion list component values. Include suggestion list components by selecting *Yes* for the options in the *Suggestion List Components* group.

Field name	Description

SUITELINK_RETCODE	Contains the match status code for SuiteLink processing.
	• *A*: SuiteLink record match. Secondary information exists and the transform assigned SuiteLink information to the record.
	• *00*: No SuiteLink record match. Transform attempted SuiteLink lookup but did not find a matching record.
	• *Blank*: No SuiteLink lookup performed because of one of the following situations:
	  • The address is not a high-rise default according to CASS.
	  • The address does not contain a firm.

UNDELIVERABLE_INDICATOR	Indicates whether the record has a deliverable address. • *T*: Address is tagged by the USPS as unsuitable for mail delivery. For example, address is for a cemetery. • *F*: Address was either not matched to a ZIP+4 record or was matched to a ZIP+4 record that indicates the address is suitable for mail delivery.
UNIT_DESCRIPTION	Contains the unit description, such as #, Apartment, or Flat.
UNIT_DESCRIPTION_DIRECTORY	Contains the unit designator from the ZIP+4 directory. The value is blank when the transform does not find a unit designator from the ZIP+4 directory.
UNIT_NUMBER	Contains the unit number, such as 100 in the unit APT 100.

USA Regulatory Address Cleanse Geo_Matchcode codes [page 757]
Geo match codes indicate the precision of the latitude and longitude assignment.

Related Information

USA Regulatory Address Cleanse fault codes [page 817]
USA Regulatory Address Cleanse status codes [page 818]
Output field category columns for Global Address Cleanse and USA Regulator Address Cleanse transforms [page 542]

3.5.12.14.1 USA Regulatory Address Cleanse Geo_Matchcode codes

Geo match codes indicate the precision of the latitude and longitude assignment.

The US Regulatory Address Cleanse transform contains a *Geo_Matchcode* output field that contains the match codes in the following table.

Geo match codes

Value	Description	Centroid	Address	Best	All
0	Address-level match		X	X	X
1	9-digit centroid-level match	X		X	
4	7-digit centroid-level match	X		X	
5	5-digit centroid-level match	X		X	
7	No centroid-level match	X			
8	No address-level match		X		X
9	Both address- and centroid-level tried; no match in either			X	
`<Blank>`	No lastline match	X	X	X	

3.5.13 Address Cleanse reference

The Address Cleanse reference contains additional information about the Global Address Cleanse and the USA Regulatory Address Cleanse transforms, including status and fault codes, USPS certifications, and country information.

USA Regulatory Address Cleanse reference

Reference information for the USA Regulatory Address Cleanse transform includes information about the following topics:

- USPS certifications
- Steps to achieve USPS certifications
- Fault codes
- Status codes
- ShowA and ShowL utilities

Global Address Cleanse reference

Reference information for the Global Address Cleanse transform includes information about the following topics:

- Country coverage
- Country ISO codes and assignment engines
- Countries supported by last line drilldown
- Information codes
- Status codes
- Quality codes

3.5.13.1 USPS certifications

SAP Data Services is certified for USPS address match, move update, and address quality processes that comply with USPS certifications.

Data Services is CASS certified. As long as you use Data Services to prepare mailings, and you enable all of the required options for certification, your mailing list is certified. That certification extends to when you integrate Data Services into your own applications.

Some companies may want to self-certify so that the company name appears in all USPS reports and in the USPS list of approved companies. However, keep in mind that when you self-certify, you may have to pay the USPS for extra charges or fees.

The USPS PostalPro Web site contains all of the certification forms and procedures in the following USPS postal publications:

- The CASS area of the PostalPro Web page is at https://postalpro.usps.com/certifications/cass .
- The NCOALink area of the PostalPro Web page is at https://postalpro.usps.com/mailing-and-shipping-services/NCOALink .
- The DSF2 area of the PostalPro Web page is at https://postalpro.usps.com/address-quality/dsf2 .

We maintain Data Services for the following USPS certifications:

- DPV
- LACSLink
- SuiteLink
- RDI (optional)
- eLOT (optional)
- NCOALink
- DSF2

SAP blueprints for USPS certifications [page 760]
SAP provides blueprints that you use to run the USPS stage tests for CASS, NCOALink, and DSF2 self certifications.

USPS CASS self certification [page 762]
SAP Data Services is already certified for USPS CASS, but we provide information about self certifying for CASS for mailers who want to self-certify.

NCOALink self-certification [page 767]
Become a certified NCOALink mailer by self-certifying using Data Services and the USA Regulatory Address Cleanse transform.

DSF2 Certification [page 776]
Before you use SAP Data Services for DSF2 processing, you obtain DSF2 certification from the USPS.

3.5.13.1.1 SAP blueprints for USPS certifications

SAP provides blueprints that you use to run the USPS stage tests for CASS, NCOALink, and DSF2 self certifications.

The blueprints contain corresponding projects, jobs, data flows, and input and output formats. Additionally, use the blueprints to process a test file provided by the USPS when the USPS requests an audit.

Find the certification blueprints in `<LINK_DIR>\DataQuality\Certifications` where `<LINK_DIR>` is the software common configuration location.

The following table contains the blueprint names and the Data Services objects that you download with the blueprints.

Blueprint	Object	File name
CASS self certification	ATL file	us_cass_self_certification.atl
	Project	DataQualityCertificationCASS
	Job	Job_dqBatchUSAReg_CASSSelfCert
	Data Flow	DF_dqBatchUSAReg_CASSSelfCert
	Transform	USARegulatory_AddressCleanse
	Source format	CASSSelfCert_in
	Target format	CASSSelfCert_out
DSF2 self certification	ATL file	us_dsf2_certification.atl
	Project	DataQualityCertificationDSF2
	Jobs	• Job_DqBatchUSAReg_DSF2Invoice • Job_DqBatchUSAReg_DSF2Sequence • Job_DqBatchUSAReg_DSF2Augment
	Data Flows	• DF_DqBatchUSAReg_DSF2Invoice • DF_DqBatchUSAReg_DSF2Sequence • DF_DqBatchUSAReg_DSF2Augment

Blueprint	Object	File name
	Transforms	• USA_Regulatory_AddressCleanse • Query: GenRowNum • DSF2_Walk_Sequence • Query: OutputOrder
	Source format	DSF2Invoice_in
	Target format	DSF2Invoice_out
NCOALink self certification	ATL	us_ncoalink_stage_certification.atl
	Project	DataQualityCertificationNCOALink
	Jobs	• Job_DqBatchUSAReg_NCOALink-StageI • Job_DqBatchUSAReg_NCOALink-StageII
	Data Flows	• DF_DQBatchUSAReg_NCOALink-StageI • DF_DQBatchUSARegNCOALink-StageII
	Transform	USARegulatoryNCOALink_Addres-sCleanse
	Source formats	• StageI_in • StageII_in
	Target formats	• StageI_out • StageII_out

Parent topic: USPS certifications [page 759]

Related Information

USPS CASS self certification [page 762]
NCOALink self-certification [page 767]
DSF2 Certification [page 776]
Importing SAP blueprints for USPS certification [page 762]

3.5.13.1.1.1 Importing SAP blueprints for USPS certification

Import the special SAP blueprints for USPS self certification processes into your Data Services implementation in your software.

1. In Data Services Designer, right-click in a blank area of the local object library and select *Repository* > *Import From File.*
2. Open the file location `<LINK_DIR>\DataQuality\Certifications` and select the applicable blueprint `.atl` file. Click *Open*.
3. Click *Yes* to the warning about overwriting existing objects.
4. View the objects to be imported in the *Import Plan* dialog box and click *Import*.

   There are options to exclude and include objects. However, import all of the objects for self certification.
5. Click *Import* in the *Passphrase* dialog box. We do not require a password for these blueprints.

   The software downloads the blueprint files. Verify that the files are in your software by opening the local object library and viewing the project listed in the *Project* tab, the data flows listed in the *Data Flows* tab, and so on.

Related Information

SAP blueprints for USPS certifications [page 760]
NCOALink blueprints [page 772]
DSF2 Certification blueprints [page 778]

3.5.13.1.2 USPS CASS self certification

SAP Data Services is already certified for USPS CASS, but we provide information about self certifying for CASS for mailers who want to self-certify.

You may want to become a CASS-certified vendor so that your company name appears in all USPS reports and on the CASS vendor list.

There are two USPS Stage tests for CASS certification: Stage I and Stage II:

- Stage I: Optional. Practice test for which you view the results. Correct any inconsistencies with the USPS-expected results and take the Stage I test until you are satisfied with the results. You do not submit Stage I test results to the USPS.
- Stage II: Required. Does not contain answers like Stage I test. Stage II tests the accuracy of the software. After you take the Stage II test, forward the results to the USPS. You cannot see your results.

For complete information and resources for CASS, visit the PostalPro Web site at https://postalpro.usps.com/certifications/cass .

USPS CASS self certification [page 763]

Even though SAP Data Services is fully certified for USPS CASS, some mailers go through the process of CASS self certtification.

Parent topic: USPS certifications [page 759]

Related Information

SAP blueprints for USPS certifications [page 760]
NCOALink self-certification [page 767]
DSF2 Certification [page 776]

3.5.13.1.2.1 USPS CASS self certification

Even though SAP Data Services is fully certified for USPS CASS, some mailers go through the process of CASS self certtification.

Consider the following information before you decide to self certify for CASS:

- You'll receive the same certification that Data Services already has. Therefore, you do not have to self-certify for CASS if you are using Data Services to produce CASS-certified mailings.
- If you have self certified your data quality application for CASS, your end users, the ones that use your data quality application, don't have to obtain their own CASS certification. When you self-certify your product, users can produce CASS-certified mailing lists with your product.
- CASS certification proves that your application can assign and standardize addresses correctly.
- CASS certification is given to software programs, not the users of the software program.
- The CASS reports pertain to address lists, not mailers.

Parent topic: USPS CASS self certification [page 762]

Related Information

USPS CASS certification static directories [page 764]

Preparing for CASS self-certification [page 765]
CASS Stage testing [page 766]
USPS Form 3553 required options [page 767]

3.5.13.1.2.2 USPS CASS certification static directories

If you are self-certifying for CASS, use static directories instead of the active directories for consistent results.

All USPS processing requires postal directories that change on a periodic basis. However, static directories do not change with the regular directory updates. The following lists some of the benefits of using static directories for USPS certification:

- Static directories do not expire until the end of the current CASS cycle.
- Static directories ensure that your output is consistent between Stage I and Stage II tests.
- When you re certify, use the same static directories as you used for original certification (as long as they haven't expired) for consistent results.

The following directories are available in static format:

- `zip4us.dir`
- `zip4us.shs`
- `zip4us.rev`
- `revzip4.dir`
- `city10.dir`
- `zcf10.dir`
- `dpv*.dir`
- `elot.dir`
- `ew*.dir`
- SuiteLink directories
- LACSLink directories

The USPS provides the test input file and the test DSF2 directories. Obtain all other directories from SAP Support.

Obtain static directories from SAP Support. SAP Support formats the static directories from the USPS so they work with Data Services.

> **i Note**
>
> The USPS provides static directories, however they are not in the format required for Data Services. Do not use the static directories from the USPS.

Store the static directories separately from the production directories. If you store them in the same folder, the static directories overwrite your production directories.

> **→ Tip**
>
> Create a folder named “static” to store your static directories. For example, create the folder named static and save your static directories to `<LINK_DIR>\DataQuality\reference_data\static`, where `<LINK_DIR>` is where the software installation resides.

Also do not run a production job using the static directories. The software issues a verification warning or error under the following circumstances:

- You set up the transform with both static and production directories.
- The release version of the `zip4us.dir` does not match the current CASS cycle software.
- The data versions in the static directories aren't all the same.
- You set up the job for self-certification but you do not set up the static directories.
- You set up the job with static directories, but you do not enable the self-certification options.

Parent topic: USPS CASS self certification [page 762]

Related Information

USPS CASS self certification [page 763]
Preparing for CASS self-certification [page 765]
CASS Stage testing [page 766]
USPS Form 3553 required options [page 767]

3.5.13.1.2.3 Preparing for CASS self-certification

Prepare to self-certify the software for CASS and NCOALink by completing all of the necessary USPS forms, obtaining an authorization code, and obtaining the license key code.

Before you certify your company for CASS, ensure that you completely understand the steps to take for certification from the USPS.

1. Obtain the USPS CASS forms from https://postalpro.usps.com/certifications/cass .

   Forms include:
   - CASS Order Form
   - Terms and Conditions
   - Electronic Product Fulfillment Web Access Request Form
2. Complete and submit the forms to the USPS.

   After the USPS approves your application, they give you an authorization code.
3. Submit the USPS authorization code to SAP Technical Support.
4. Obtain the license key by following instructions provided by SAP Technical Support.

   The license key enables the required functionality in the USPS Regulatory Address Cleanse transform.

Task overview: USPS CASS self certification [page 762]

Related Information

USPS CASS self certification [page 763]
USPS CASS certification static directories [page 764]
CASS Stage testing [page 766]
USPS Form 3553 required options [page 767]
USPS certifications [page 759]
CASS Stage testing [page 766]

3.5.13.1.2.4 CASS Stage testing

The USPS provides an optional practice test named Stage I and the final test named Stage II.

1. Optional. Download the CASS Stage I test from the USPS.

   The USPS does not require that you take the Stage I test. It is a practice test. Use the results of the Stage I test to fine-tune your software so that the results compare favorably with the USPS. For details about obtaining the Stage I test from the USPS, see the USPS *CASS Technical Guide*.
2. When you are satisfied that your Stage I test results compare favorably with the USPS-expected results, request the Stage II test from the USPS.
3. Download and unzip the Stage II test file to an output area.
4. After you run the Stage II file, check that the totals on the USPS Form 3553 and the actual totals from the processed file match.

   Use the supplied CASS self-certification blueprint to set up the Stage II test job.
5. Follow USPS procedures to turn in the Stage II test results.

The USPS takes about two weeks to grade your test.

Task overview: USPS CASS self certification [page 762]

Related Information

USPS CASS self certification [page 763]
USPS CASS certification static directories [page 764]
Preparing for CASS self-certification [page 765]
USPS Form 3553 required options [page 767]

3.5.13.1.2.5 USPS Form 3553 required options

Complete the required options in the CASS Report Options group in the USA Regulatory Address Cleanse transform to be compliant with CASS requirements.

Option	Description
List Name	Retain the default setting of *Stage File*.
List Owner	Retain the default setting of *USPS*.
Mailer Address(1-4)	Specify the name and address of your company, or the company that is becoming CASS certified (up to 29 characters per line).
Company Name Certified	Specify the name of the company that owns Data Services software that processes the mailing list. Reported in the Form 3553.
Software Version	Specify the software name and version number that you are using to receive CASS self certification. **Example** SAP Data Services, 4.x.x

Parent topic: USPS CASS self certification [page 762]

Related Information

USPS CASS self certification [page 763]
USPS CASS certification static directories [page 764]
Preparing for CASS self-certification [page 765]
CASS Stage testing [page 766]

3.5.13.1.3 NCOALink self-certification

Become a certified NCOALink mailer by self-certifying using Data Services and the USA Regulatory Address Cleanse transform.

> **Note**
>
> Self-certifying for NCOALink is optional. When you purchase NCOALink from SAP, the software is already certified for NCOALink. You may want to self-certify so that your company name appears on all reports and in the NCOALink vendor list.

The following steps provide a high-level overview of the tasks required to obtain NCOALink certification.

1. Complete the USPS Application and Self-Certification Statement Approval and submit to the USPS.
2. The USPS sends you an authorization code upon approval.
3. Contact SAP to purchase the necessary NCOALink functionality from SAP. SAP requires your USPS authorization code.
4. Take the required Stage II test.
5. Complete the USPS License Agreement and pay the USPS for NCOALink.

During certification, you process files from the USPS to prove that the software adheres to the requirements of your license agreement. NCOALink certification has two Stage tests:

- Stage I: Optional. Includes answers that allow you to troubleshoot and prepare for the Stage II test.
- Stage II: Required. Does not contain answers. Send results to the USPS for evaluation of the accuracy of your software configuration.

For information about the entire procedure, see the PostalPro Web site at https://postalpro.usps.com/mailing-and-shipping-services/NCOALink . Scroll to the Featured Resources and find the group of documents related to the type of license you are applying for:

- NCOALink End User Documents
- NCOALink Limited Service Provider Documents
- NCOALink Full Service Provider Documents

NCOALink directories and Daily Delete file [page 769]
For NCOALink certification, the USPS provides the NCOALink test data after you receive the USPS authorization code and after you obtain NCOALink software from Data Services.

Extracting NCOALink directories with the utility [page 770]
After you obtain the NCOALink directories from the USPS, use the NCOA utility to extract and uncompress the directories.

NCOALink software product information [page 771]
Complete the NCOALink Compliance Testing Product Information Form with the applicable information about the licensee and SAP software.

NCOALink blueprints [page 772]
SAP Data Services includes a set of objects in the NCOALink blueprint file to use for NCOALink testing.

Configuring the NCOALink blueprints [page 773]
Configure the provided NCOALink blueprint objects before you use the objects to run your Stage I or Stage II tests.

Executing the NCOALink Stage II job [page 775]
Execute the Stage II test and submit the results to the USPS.

Exporting NCOALink log files from the repository [page 776]
For NCOALink certification, you export the NCOALink log files from the repository that result from Stage II testing.

Parent topic: USPS certifications [page 759]

Related Information

SAP blueprints for USPS certifications [page 760]
USPS CASS self certification [page 762]
DSF2 Certification [page 776]

3.5.13.1.3.1 NCOALink directories and Daily Delete file

For NCOALink certification, the USPS provides the NCOALink test data after you receive the USPS authorization code and after you obtain NCOALink software from Data Services.

After you are licensed for NCOALink, obtain the NCOALink directories from the USPS either daily or monthly:

- Monthly for NCOALink end users
- Weekly for NCOALink full and limited service providers

Additionally, the USPS requires that you obtain the Daily Delete file daily, or on the days that you run your NCOALink jobs. Download the daily delete file from the USPS and copy it to the directory where your NCOALink directories are located.

Download the NCOALink directories using the EPF Downloader Manager, or your own tool if applicable. Then use the SAP NCOALink utility to extract and uncompress the directories.

The SAP NCOALink utility is a 64-bit application that you run from a command line. Find the utility in `%LINK_DIR%\bin\ncoa\ncoautil.exe` (Windows) or `$LINK_DIR/bin/ncoa/ncoautil` (UNIX). If the utility is not there, download it from the SAP Support Portal at https://support.sap.com/software/address-directories.html.

For complete details about NCOALink directories and the daily delete file, visit the NCOALink pages in the PostalPro Web site at https://postalpro.usps.com/mailing-and-shipping-services/NCOALink .

Parent topic: NCOALink self-certification [page 767]

Related Information

Extracting NCOALink directories with the utility [page 770]
NCOALink software product information [page 771]
NCOALink blueprints [page 772]
Configuring the NCOALink blueprints [page 773]
Executing the NCOALink Stage II job [page 775]
Exporting NCOALink log files from the repository [page 776]

3.5.13.1.3.2 Extracting NCOALink directories with the utility

After you obtain the NCOALink directories from the USPS, use the NCOA utility to extract and uncompress the directories.

This procedure applies after you have obtained your NCOALink license, for regular NCOALink processing.

Before you extract and uncompress the directories, ensure that your system meets the following minimum requirements:

- At least 60 GB of available disk space
- Sufficient RAM

1. Run the NCOALink Utility from the location where it is installed.

   If your NCOALink Utility was installed with the software, the utility is located in the following folder, where `LINK_DIR` is the software installation directory:

   - `%LINK_DIR%\bin\ncoa\ncoautil.exe` (Windows)
   - `$LINK_DIR/bin/ncoa/ncoautil` (UNIX)
2. Use the `ncoautil` command with the following command-line options:

Option		
Windows	**UNIX**	**Description**
`/p:t`	`-p:t`	Perform transfer to copy files from the source to the destination. Also specify the following locations • Compressed NCOALink data files with `/d` or `-d` • Transfer destination location with `/t` or `-t`
`/p:u`	`-p:u`	Perform unpack to uncompress the files. When you use this command, include the following commands: • `/d` or `-d`: Transfer destination location • `/t` or `-t`: Transfer destination location
`/p:v`	`-p:v`	Perform verification on the fields. When you use this command, include the transfer destination location with `/t` or `-t`.
`/d`	`-d`	Specify location of compressed NCOALink data files.
`/t`	`-t`	Specify transfer destination location.
`/nos`	`-nos`	Do not stop on error (returns failure code as exit status).
`/a`	`-a`	Answer all warning messages with Yes.

Combine `p` options when you want to perform multiple tasks at once.

> **Example**
>
> To transfer, unpack, and verify all in the same process, enter `/p:tuv` or `-p:tuv`.

After performing the `p` option specified, the NCOALink utility closes.

Example

For Windows, the command may look like the following example:

```
ncoautil /p:tuv /d D:\downloads\ncoa /t C:\Program Files (x86)\SAP
BusinessObjects\Data Services\DataQuality\reference_data
```

For UNIX, the command may look like the following example:

```
ncoautil -a -nos -p:tuv -d /local/downloads/ncoa -t /local/dataservices/
DataQuality/reference_data
```

Task overview: NCOALink self-certification [page 767]

Related Information

NCOALink directories and Daily Delete file [page 769]
NCOALink software product information [page 771]
NCOALink blueprints [page 772]
Configuring the NCOALink blueprints [page 773]
Executing the NCOALink Stage II job [page 775]
Exporting NCOALink log files from the repository [page 776]

3.5.13.1.3.3 NCOALink software product information

Complete the NCOALink Compliance Testing Product Information Form with the applicable information about the licensee and SAP software.

Obtain the Compliance Testing Product Information Form on the PostalPro Web site at https://postalpro.usps.com/mailing-and-shipping-services/NCOALink .

Compliance Testing Product Information form	Description
Company Name & License Number	Specify the NCOALink licensee company name. The license number is the authorization code provided in your USPS approval letter.
Company's NCOALink Product Name	Enter Mover ID for NCOALink.
Platform or Operating System	Specify the NCOALink licensee information.
NCOALink Software Vendor	Enter SAP Americas, Inc.
NCOALink Software Product Name	Enter Mover ID.
NCOALink Software Product Version	Enter ACE.

Compliance Testing Product Information form	Description
Address Matching ZIP+4 Product Name Address Matching ZIP+4 Product Version	Contact SAP Support for this information.
Address Matching ZIP+4 System	Enter Closed.
Is Software Hardware Dependent?	Enter No.
DPV Product Name	Enter ACE.
DPV Product Version	Contact SAP Support.
LACSLink Product Name	Enter ACE.
LACSLink Product Version	Contact SAP Support.
NCOALink Software options: Integrated or Standalone checkboxes	Select Integrated.
ANKLink Enhancement checkbox (applicable for Limited Service Providers and End Users)	Select if you purchased the ANKLink option from SAP.
HASH—FLAT—BOTH checkboxes	Specifies the file format in which to receive the NCOALink information. SAP supports both formats.
NCOALink Level Option checkboxes	Specifies the NCOALink level for which you are testing.

Parent topic: NCOALink self-certification [page 767]

Related Information

NCOALink directories and Daily Delete file [page 769]
Extracting NCOALink directories with the utility [page 770]
NCOALink blueprints [page 772]
Configuring the NCOALink blueprints [page 773]
Executing the NCOALink Stage II job [page 775]
Exporting NCOALink log files from the repository [page 776]

3.5.13.1.3.4 NCOALink blueprints

SAP Data Services includes a set of objects in the NCOALink blueprint file to use for NCOALink testing.

The blueprints contain the corresponding projects, jobs, data flows, input, and output formats. Additionally, you may use the blueprints to process a test file for a USPS audit.

Import NCOALink blueprints from `<LINK_DIR>\DataQuality\Certifications` where `<LINK_DIR>` is the software common configuration directory.

The following table contains the file names for the Stage I NCOALink blueprints:

Stage I blueprint object file names

Object	Name
ATL file	us_ncoalink_stage_certification.atl
Project	DataQualityCertificationNCOALink
Job	Job_DqBatchUSAReg_NCOALinkStageI
data flow	DF_DqBatchUSAReg_NCOALinkStageI
Input file format	DqUsaNCOALinkStageI _in
Output file format	DqUsaNCOALinkStageI _out

The following table contains the file names for the Stage II NCOALink blueprints:

Stage II blueprint object file names

Object	Name
ATL file	us_ncoalink_stage_certification.atl
Project	DataQualityCertificationNCOALink
Job	Job_DqBatchUSAReg_NCOALinkStageII
data flow	DF_DqBatchUSAReg_NCOALinkStageII
Input file format	DqUsaNCOALinkStageII _in
Output file format	DqUsaNCOALinkStageII _out

Parent topic: NCOALink self-certification [page 767]

Related Information

NCOALink directories and Daily Delete file [page 769]
Extracting NCOALink directories with the utility [page 770]
NCOALink software product information [page 771]
Configuring the NCOALink blueprints [page 773]
Executing the NCOALink Stage II job [page 775]
Exporting NCOALink log files from the repository [page 776]

3.5.13.1.3.5 Configuring the NCOALink blueprints

Configure the provided NCOALink blueprint objects before you use the objects to run your Stage I or Stage II tests.

Import the NCOALink blueprints.

Use the *Substitution Parameter Editor* to set values for the related NCOALink substitution parameters. Using substitution parameters is optional except for setting the $$USPSProviderLevel substitution parameter value for *Provider Level*.

1. In Designer, open the *Projects* tab in the Object Library and double-click the NCOALink project named `DataQualityCertificationsNCOALink`.

   The project appears in the Project Area.
2. Expand the `Job_DqBatchUSAReg_NCOALinkStageI` job node.
3. Double-click the `DF_DqBatchUSAReg_NCOALinkStageI` data flow to open it in the workspace.
4. In the workspace, double-click the DqUsaNCOALinkStageI_in input source to open the *Source File Editor*.
5. Make the following changes in the *Data Files(s)* property group in the bottom pane of the editor:
   a. For *Root Directory*, browse to the directory that contains the Stage I input file supplied by the USPS.
   b. For *File name(s)* , enter the name of the Stage I input file.
   c. Close the *Source File Editor*.
6. Double-click the `DqUsaNCOALinkStageI_out` output file to open the *Target File Editor*.
7. Make the following changes In the *Data Files(s)* property group in the bottom pane of the editor:
   a. For *Root Directory*, browse to the directory that contains the output file.
   b. (Optional.) For *File name(s)*, change the output file name to conform to your file naming convention.
   c. Close the *Target File Editor*.
8. Double-click the `USARegulatoryNCOALink_AddressCleanse` transform to open the *Transform Editor* and open the *Options* tab.
9. Complete the options in the *Reference Files* group. For convenience, use the $$RefFilesAddressCleanse substitution parameters if you prepared them earlier.
10. Complete the options in the *USPS License Information* group:
    a. Enter a list ID in the *List ID* option.

       Make the List ID memorable so you can identify it in reports.
    b. Enter the current date in the *List Received Date* and *List Return Date* options.
    c. Select the substitution parameter $$USSProviderLevelprovider level for *Provider Level*.
    d. If you are a full service provider or limited service provider, complete the options in the *PAF Details* group and the *Service Provider Options* group under *NCOALink*

Task overview: NCOALink self-certification [page 767]

Related Information

NCOALink directories and Daily Delete file [page 769]
Extracting NCOALink directories with the utility [page 770]
NCOALink software product information [page 771]
NCOALink blueprints [page 772]
Executing the NCOALink Stage II job [page 775]
Exporting NCOALink log files from the repository [page 776]
USPS certifications [page 759]

SAP blueprints for USPS certifications [page 760]
Overview of substitution parameters

3.5.13.1.3.6 Executing the NCOALink Stage II job

Execute the Stage II test and submit the results to the USPS.

Before you run the NCOALink Stage II certification job, make sure that you have installed the required directories to the locations that you specified in your Stage II blueprint objects:

- DPV
- LACSLink
- US National Directory files
- NCOALink directories including the Daily Delete file.

1. Download the current version of the USPS daily delete file from https://epf.usps.gov/ .
2. Download the Stage II file from and uncompress it to the location you specified when setting up the Stage II blueprint objects.

   Ensure the input file name in the source transform matches the name of the Stage II file from the USPS.
3. Execute the Stage II job. Follow the specific instructions in the *NCOALink Certification/Audit Instructions* document that the USPS provides.
4. Export the required log files using the Data Services Management Console.
5. Compress the following results and name the compressed file using the same name as the downloaded Stage II file from the USPS:
   - Stage II output file
   - NCOALink Processing Summary Report
   - CASS Form 3553
   - All log files generated in the `$$CertificationLog` path
     - Customer Service Log
     - PAF (Service Providers only)
     - Broker/Agent/List Administrator log (Service Providers only)
6. Send the results to the USPS for verification.

Task overview: NCOALink self-certification [page 767]

Related Information

NCOALink directories and Daily Delete file [page 769]
Extracting NCOALink directories with the utility [page 770]
NCOALink software product information [page 771]
NCOALink blueprints [page 772]
Configuring the NCOALink blueprints [page 773]

Exporting NCOALink log files from the repository [page 776]

3.5.13.1.3.7 Exporting NCOALink log files from the repository

For NCOALink certification, you export the NCOALink log files from the repository that result from Stage II testing.

Export the NCOALink log files from the repository using the Data Services Management Console. Compress the log files with the other required files to send to the USPS for approval.

1. Log in to the Management Console and open the Administrator.
2. Expand the *Management* node at left and then expand *Certification Logs*.
3. Select the applicable repository.
4. Open the *NCOALink* tab.
5. Set the date range to export.
6. Select the *NCOA Licensee ID* from the dropdown list if required.
7. Select the export location.
8. Select *Export*.

The software exports the following log file in the required USPS format to the location that you specified:

- Customer Service Log (CSL)
- PAF for limited and full service providers
- Broker/Agent/List Administrator log for limited and full service providers

Task overview: NCOALink self-certification [page 767]

Related Information

NCOALink directories and Daily Delete file [page 769]
Extracting NCOALink directories with the utility [page 770]
NCOALink software product information [page 771]
NCOALink blueprints [page 772]
Configuring the NCOALink blueprints [page 773]
Executing the NCOALink Stage II job [page 775]

3.5.13.1.4 DSF2 Certification

Before you use SAP Data Services for DSF2 processing, you obtain DSF2 certification from the USPS.

Consult the *DSF2 Certification Package* document for complete certification processes. For complete details about DSF2 and certification, visit the PostalPro Web site at https://postalpro.usps.com/address-quality/dsf2 . SAP requires that you purchase the DSF2 product before you perform the steps for DSF2 certification.

The DSF2 certification consists of the following steps. Wait for written approval for each step before you proceed to the next step.

1. Complete and submit all required documentation including the Application and Self-Certification Statement.
2. Complete detailed documentation including Technical Proposal and Security Documentation.
3. The USPS sends you the Stage I test file along with an Interface Developer Guide.
4. When you are ready for the certification test, send a written request to the USPS and take the test.
5. The USPS sends you an approval letter and license agreement, which you sign and return.

   Complete the DSF2 Equipment Information form [page 777]
   You supply the USPS with information about the software you will use for DSF2 certification and processing on the Equipment Information form.

   DSF2 Certification blueprints [page 778]
   SAP Data Services includes a set of objects in the DSF2 blueprint file to use for DSF2 certification.

   Configuring the DSF2 blueprints [page 779]
   Configure the provided DSF2 blueprint objects before you use the objects to run your certification tests.

Parent topic: USPS certifications [page 759]

Related Information

SAP blueprints for USPS certifications [page 760]
USPS CASS self certification [page 762]
NCOALink self-certification [page 767]

3.5.13.1.4.1 Complete the DSF2 Equipment Information form

You supply the USPS with information about the software you will use for DSF2 certification and processing on the Equipment Information form.

Use the information in the following table to complete the form for the DSF2 certification.

Equipment Information form	Description
Interface Software Vendor	SAP Americas, Inc.
Interface Software Product Name	ACE
Interface Software Product Version	Contact SAP Technical Support.
Address Matching ZIP+4 Product Name	ACE
Address Matching ZIP+4 Product Version	Contact SAP Technical Support.
Address Matching ZIP+4 System	Closed System

Equipment Information form	Description
Interface Hardware Vendor/Model/Type	N/A The software is not hardware dependent
Interface Hardware Operating System	N/A The software is not hardware dependent
Interface Hardware Serial Number	N/A The software is not hardware dependent

You complete the Equipment Information form in step 2 of the DSF2 process. For details, go to https://postalpro.usps.com/address-quality/dsf2 .

Parent topic: DSF2 Certification [page 776]

Related Information

DSF2 Certification blueprints [page 778]
Configuring the DSF2 blueprints [page 779]

3.5.13.1.4.2 DSF2 Certification blueprints

SAP Data Services includes a set of objects in the DSF2 blueprint file to use for DSF2 certification.

The blueprints contain the corresponding projects, jobs, data flows, input, and output formats. Additionally, use the blueprints to process a test file provided by the USPS during an audit.

Import the DSF2 certification blueprints from `<DS_COMMON_DIR>\DataQuality\Certifications`, where `<DS_COMMON_DIR>` is the software common configuration location.

There are three types of DSF2 certifications so we provide three groups of blueprint objects: Augment, Invoice, and Sequence.

USPS DSF2 Augment certification:

DSF2 Augment certification blueprint objects

Object	Name
ATL file	us_dsf2_certification.atl
Project	DataQualityCertificationDSF2
Job	Job_DqBatchUSAReg_DSF2Augment
data flow	DF_DqBatchUSAReg_DSF2Augment
Input file format	DqUsaDSF2Augment_in
Output file format	DqUsaDSF2Augment_out

USPS DSF2 Invoice certification:

DSF2 Invoice certification blueprint objects

Object	Name
ATL file	us_dsf2_certification.atl
Project	DataQualityCertificationDSF2
Job	Job_DqBatchUSAReg_DSF2Invoice
data flow	DF_DqBatchUSAReg_DSF2Invoice
Input file format	DqUsaDSF2Invoice_in
Output file format	DqUsaDSF2Invoice_out

USPS DSF2 Sequence certification:

DSF2 Sequence certification blueprint objects

Object	Name
ATL file	us_dsf2_certification.atl
Project	DataQualityCertificationDSF2
Job	Job_DqBatchUSAReg_DSF2Sequence
data flow	DF_DqBatchUSAReg_DSF2Sequence
Input file format	DqUsaDSF2Sequence_in
Output file format	DqUsaDSF2Sequence_out

Parent topic: DSF2 Certification [page 776]

Related Information

Complete the DSF2 Equipment Information form [page 777]
Configuring the DSF2 blueprints [page 779]

3.5.13.1.4.3 Configuring the DSF2 blueprints

Configure the provided DSF2 blueprint objects before you use the objects to run your certification tests.

Before performing the steps below, import the DSF2 blueprints.

Use the *Substitution Parameter Editor* to set values for the related DSF2 substitution parameters.

> **i Note**
>
> DSF2 Augment only. Enter the static directories location for the $$RefFilesUSPSStatic substitution variable.

1. In Designer, open the *Projects* tab in the Object Library and double-click the DSF2 project named `DataQualityCertificationDSF2`.

The project appears in the Project Area.

2. Expand the desired certification job to reveal the data flow.
3. Double-click the related data flow to open it in the workspace.
4. In the workspace, double-click the applicable input file format (`*.in`) to open the *Source File Editor*.
5. Make the following changes in the *Data Files(s)* property group:
   a. For *Root Directory*, browse to the directory that contains the DSF2 input file.
   b. For *File name(s)*, enter the name of the DSF2 input file.
   c. Close the *Source File Editor*.
6. Double-click the applicable output file format (`*.out`) to open the *Target File Editor*.
7. Make the following changes in the *Data Files(s)* property group:
   a. For *Root Directory*, browse to the directory that contains the DSF2 output file.
   b. Optional. For *File name(s)* , change the output file name to conform to your file naming convention.
   c. Close the *Target File Editor*.
8. Double-click the `USARegulatory_AddressCleanse` transform to open the *Transform Editor* and open the *Options* tab.

   > **i Note**
   >
   > For DSF2 Sequence and Invoice certifications, you will open the DSF2_Walk_Sequencer transform.

9. Complete the options in the Reference Files group. For convenience, use the substitution parameters that you prepared earlier.

   For DSF2 Augment certification, use the $$RefFilesUSPSStatic substitution variable to save time.
10. In the CASS Report Options, update each option that is listed as "CHANGE_THIS" if applicable.

Task overview: DSF2 Certification [page 776]

Related Information

Complete the DSF2 Equipment Information form [page 777]
DSF2 Certification blueprints [page 778]
SAP blueprints for USPS certifications [page 760]
Importing SAP blueprints for USPS certification [page 762]
Overview of substitution parameters

3.5.13.2 Country coverage

This table contains information about address cleansing and geocode coverage per country.

> **i Note**
>
> Reference to "country" or "countries" in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

The “Country Reference Data” column identifies which country-specific reference data covers each country for address cleansing. For each country that is covered by a country-specific reference data the chart identifies which languages and scripts are supported in the reference data, and to what depth address data can be validated.

The validation levels from poorest to finest are as follows.

- City – Validation is to the city (locality) and postcode.
- Primary – Validation is to the street, and sometimes to the house number.
- Secondary – Validation is to the building or secondary data such as floor, apartment, or suite.

In the following table, countries where the Validation Level column is blank use the All-World reference data. This data supports the local language in Latin script and validates to the city level.

Validation to a particular level means that, when the address data sent in the request matches data in the reference data, then SAP Data Services can correct errors and add missing components. While the remaining address data is not validated, Data Services can still standardize and format it using the country norms and the address cleansing rules.

The “Geo” column identifies which countries are supported for returning geo-location coordinates.

> **i Note**
>
> Geo and Reverse Geo aren’t supported for US Territories.
>
> Also note that greater geocoding coverage occurs when using the DQMm transform. See the SAP Data Quality Management, microservices Country Coverage topic for more detailed information.

Country/ Region Code	Country/Region Name	Country/ Region Refer- ence Data	Languages	Scripts	Validation Level	Geo
AD	Andorra	Spain	Spanish	Latin	Primary	
AE	United Arab Emirates					
AF	Afghanistan					
AG	Antigua and Barbuda					
AI	Anguilla					
AL	Albania					
AM	Armenia					
AO	Angola					
AQ	Antarctica				N/A for All World directo- ries Country Identification Engine recog- nizes country	
AR	Argentina					
AS	American Samoa	United States	English	Latin	Primary	
AT	Austria	Austria	German	Latin	Secondary	Yes

Country/ Region Code	Country/Region Name	Country/ Region Reference Data	Languages	Scripts	Validation Level	Geo
AU	Australia	Australia	English	Latin	Primary	Yes
AW	Aruba					
AX	Åland Islands	Finland	Swedish and Finnish	Latin	City	
AZ	Azerbaijan					
BA	Bosnia and Herzegovina					
BB	Barbados					
BD	Bangladesh					
BE	Belgium	Belgium	Dutch, French, German	Latin	Primary	Yes
BF	Burkina Faso					
BG	Bulgaria	Bulgaria	Bulgarian	Cyrillic, Latin	Primary	
BH	Bahrain					
BI	Burundi					
BJ	Benin					
BL	Saint Barthélemy	France	French	Latin	Secondary	
BM	Bermuda					
BN	Brunei Darussalam					
BO	Plurinational State of Bolivia					
BQ	Bonaire, Sint Eustatius, and Saba					
BR	Brazil	Brazil	Portuguese	Latin	Primary	Yes
BS	Bahamas (the)					
BT	Bhutan					
BV	Bouvet Island				N/A for All World directories Country Identification Engine recognizes country	
BW	Botswana					
BY	Belarus					
BZ	Belize					
CA	Canada	Canada	English, French	Latin	Secondary	Yes
CC	Cocos (Keeling) Islands	Australia	English	Latin	Primary	

Country/ Region Code	Country/Region Name	Country/ Region Refer- ence Data	Languages	Scripts	Validation Level	Geo
CD	Congo (Democratic Republic of the)					
CF	Central African Republic (the)					
CG	Republic of Congo					
CH	Switzerland	Switzerland	German, French, Italian	Latin	Secondary	Yes
CI	Côte d'Ivoire					
CK	Cook Islands (the)					
CL	Chile					
CM	Cameroon					
CN	People's Republic of China	China	Chinese	Simplified Chinese, Latin	Primary	
CO	Colombia					
CR	Costa Rica					
CU	Cuba					
CV	Cabo Verde					
CW	Curaçao					
CX	Christmas Island	Australia	English	Latin	Primary	
CY	Cyprus					
CZ	Czechia	Czechia	Czech	Latin	Secondary Locality ali- ases for Eng- lish, Italian, and German	Yes
DE	Germany	Germany	German	Latin	Primary	Yes
DJ	Djibouti					
DK	Denmark	Denmark	Danish	Latin	Primary	Yes
DM	Dominica					
DO	Dominican Republic (the)					
DZ	Algeria					
EC	Ecuador					
EE	Estonia	Estonia	Estonian	Latin	Secondary	Yes
EG	Egypt					

Country/ Region Code	Country/Region Name	Country/ Region Reference Data	Languages	Scripts	Validation Level	Geo
EH	Western Sahara				N/A for All World directories Country Identification Engine recognizes country	
ER	Eritrea					
ES	Spain	Spain	Spanish (including Catalan, Galician, and Basque)	Latin	Primary	Yes
ET	Ethiopia					
FI	Finland	Finland	Finnish Swedish	Latin	Primary	Yes
FJ	Fiji					
FK	Falkland Islands (the) (Malvinas)					
FM	Micronesia (Federated States of)	United States	English	Latin	Primary	
FO	Faroe Islands (the)	Denmark	Danish	Latin	Primary	
FR	France	France	French	Latin	Secondary	Yes
GA	Gabon					
GB	United Kingdom of Great Britain and Northern Ireland	United Kingdom	English	Latin	Secondary	Yes
GD	Grenada					
GE	Georgia					
GF	French Guiana	France	French	Latin	Secondary	
GG	Guernsey	United Kingdom	English	Latin	Secondary	
GH	Ghana					
GI	Gibraltar					
GL	Greenland	Denmark	Danish, Kalaallisut	Latin	City	
GM	Gambia (the)					
GN	Guinea					
GP	Guadeloupe	France	French	Latin	Secondary	

Country/ Region Code	Country/Region Name	Country/ Region Reference Data	Languages	Scripts	Validation Level	Geo
GQ	Equatorial Guinea					
GR	Greece				Primary	
GS	South Georgia and the South Sandwich Islands				N/A for All World directories Country Identification Engine recognizes country	
GT	Guatemala					
GU	Guam	United States	English	Latin	Secondary	
GW	Guinea-Bissau					
GY	Guyana					
HK	Hong Kong	China	Chinese			
HM	Heard Island and McDonald Islands				N/A for All World directories Country Identification Engine recognizes country	
HN	Honduras					
HR	Croatia					
HT	Haiti					
HU	Hungary	Hungary	Hungarian	Latin	Primary	
ID	Indonesia					
IE	Ireland	Ireland	English, Irish	Latin	Secondary	
IL	Israel					
IM	Isle of Man	United Kingdom	English	Latin	Secondary	
IN	India	India	English	Latin	Primary	
IO	British Indian Ocean Territory (the)				N/A for All World directories Country Identification Engine recognizes country	
IQ	Iraq					

Country/ Region Code	Country/Region Name	Country/ Region Refer-ence Data	Languages	Scripts	Validation Level	Geo
IR	Iran (Islamic Republic of)					
IS	Iceland					
IT	Italy	Italy	Italian	Latin	Primary	Yes
JE	Jersey	United King-dom	English	Latin	Secondary	
JM	Jamaica					
JO	Jordan					
JP	Japan	Japan	Japanese	Kanji, Hira-gana, Kata-kana	Primary	
KE	Kenya					
KG	Kyrgyzstan					
KH	Cambodia					
KI	Kiribati					
KM	Comoros (the)					
KN	Saint Kitts and Nevis					
KP	Korea (Democratic People's Republic of)					
KR	Korea (Republic of)	Korea	Korean	Hangul, Latin	Primary	Yes
KW	Kuwait					
KY	Cayman Islands (the)					
KZ	Kazakhstan					
LA	Lao People's Democratic Re-public (the)					
LB	Lebanon					
LC	Saint Lucia					
LI	Liechtenstein	Switzerland	German	Latin	Primary	Yes
LK	Sri Lanka					
LR	Liberia					
LS	Lesotho					
LT	Lithuania	Lithuania	Lithuanian	Latin	Primary	Yes
LU	Luxembourg	Luxembourg	French, German, Lëtzebuergesch	Latin	Primary	Yes
LV	Latvia	Latvia	Latvian	Latin	Secondary	
LY	Libya					

Country/ Region Code	Country/Region Name	Country/ Region Refer- ence Data	Languages	Scripts	Validation Level	Geo
MA	Morocco					
MC	Monaco	France	French	Latin	Primary (di- rectory con- tains some or- ganization in- formation)	
MD	Moldova (the Republic of)					
ME	Montenegro					
MF	Saint Martin (French part)	France	French	Latin	Secondary	Yes
MG	Madagascar					
MH	Marshall Islands (the)	United States	English	Latin	Primary	
MK	North Macedonia					
ML	Mali					
MM	Myanmar					
MN	Mongolia					
MO	Macao (also known as Macau)	China	Chinese	Traditional Chinese, Latin	Primary	Yes
MP	Northern Mariana Islands (the)	United States	English	Latin	Primary	
MQ	Martinique	France	French	Latin	Secondary	
MR	Mauritania					
MS	Montserrat					
MT	Malta					
MU	Mauritius					
MV	Maldives					
MW	Malawi					
MX	Mexico	Mexico	Spanish	Latin	Primary	
MY	Malaysia					
MZ	Mozambique					
NA	Namibia					
NC	New Caledonia	France	French	Latin	City	
NE	Niger (the)					
NF	Norfolk Island	Australia	English	Latin	Primary	
NG	Nigeria					
NI	Nicaragua					

Country/ Region Code	Country/Region Name	Country/ Region Refer- ence Data	Languages	Scripts	Validation Level	Geo
NL	Netherlands	Netherlands	Dutch	Latin	Primary	Yes
NO	Norway	Norway	Norwegian	Latin	Primary	Yes
NP	Nepal					
NR	Nauru					
NU	Niue				N/A for All World Directo- ries Country Iden- tification En- gine recog- nizes country	
OM	Oman					
PA	Panama					
PE	Peru					
PF	French Polynesia	France	French	Latin	City	
PG	Papua New Guinea					
PH	Philippines (the)					
PK	Pakistan					
PL	Poland	Poland	Polish	Latin	Primary	Yes
PM	Saint Pierre and Miquelon	France	French	Latin	City	
PN	Pitcairn					
PR	Puerto Rico	United States	Spanish	Latin	Secondary	
PS	Palestine (State of)					
PT	Portugal	Portugal	Portuguese	Latin	Secondary	Yes
PW	Palau	United States	English	Latin	Primary	
PY	Paraguay					
QA	Qatar					
RE	Réunion	France	French	Latin	Secondary	
RO	Romania					
RS	Serbia					
RU	Russian Federation (the)	Russia	Russian	Cyrillic, Latin	Secondary	
RW	Rwanda					
SA	Saudi Arabia					
SB	Solomon Islands					
SC	Seychelles					

Country/ Region Code	Country/Region Name	Country/ Region Refer- ence Data	Languages	Scripts	Validation Level	Geo
SD	Sudan (the)					
SE	Sweden	Sweden	Swedish	Latin	Primary	Yes
SG	Singapore	Singapore	English	Latin	Secondary	Yes
SH	Saint Helena, Ascension, and Tristan da Cunha					
SI	Slovenia					
SJ	Svalbard and Jan Mayen	Norway	Norwegian	Latin	Primary	
SK	Slovakia	Slovakia	Slovakian	Latin	Primary	
SL	Sierra Leone					
SM	San Marino	Italy	Italian	Latin	Primary	
SN	Senegal					
SO	Somalia					
SR	Suriname					
SS	South Sudan					
ST	Sao Tome and Principe					
SV	El Salvador					
SX	Sint Maarten (Dutch part)					
SY	Syrian Arab Republic					
SZ	Eswatini Formerly Swaziland					
TC	Turks and Caicos Islands (the)					
TD	Chad					
TF	French Southern Territories (the)					
TG	Togo					
TH	Thailand					
TJ	Tajikistan					
TK	Tokelau				N/A for All World Directo- ries Country Iden- tification En- gine recog- nizes country	

Country/ Region Code	Country/Region Name	Country/ Region Refer- ence Data	Languages	Scripts	Validation Level	Geo
TL	Timor-Leste				N/A for All World Directo- ries Country Iden- tification En- gine recog- nizes country	
TM	Turkmenistan					
TN	Tunisia					
TO	Tonga					
TR	Turkey	Turkey	Turkish	Latin	Primary	Yes
TT	Trinidad and Tobago					
TV	Tuvalu					
TW	Taiwan	China	Chinese	Traditional Chinese, Latin	Primary Supported with the All World direc- tory	Yes
TZ	Tanzania, United Republic of					
UA	Ukraine					
UG	Uganda					
UM	United States Minor Outlying Islands (the)				N/A for All World Directo- ries Country Iden- tification En- gine recog- nizes country	
US	United States of America (the)	United States	English	Latin	Secondary	Yes
UY	Uruguay					
UZ	Uzbekistan					
VA	Holy See (the)	Italy	Italian	Latin	Primary	
VC	Saint Vincent and the Grena- dines					
VE	Venezuela (Bolivarian Repub- lic of)					
VG	Virgin Islands (British)					
VI	Virgin Islands (U.S.)	United States	English	Latin	Secondary	Yes

Country/ Region Code	Country/Region Name	Country/ Region Refer- ence Data	Languages	Scripts	Validation Level	Geo
VN	Viet Nam					
VU	Vanuatu					
WF	Wallis and Futuna	France	French	Latin	City	
WS	Samoa					
YE	Yemen					
YT	Mayotte	France	French	Latin	Secondary	
ZA	South Africa					
ZM	Zambia					
ZW	Zimbabwe					

3.5.13.3 Country ISO codes and assignment engines

Use country ISO code and assignment engine availability, plus more detailed information, to understand processing results.

> **i Note**
>
> Reference to “country” or “countries” in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

The following 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 Key

Engine	Assignment Level
Canada = C	Country = C
Global Address = G	Locality = L
USA = U	Primary Name = Pn
	Premise = Pr
	Secondary = S

Country/ Region name	2-char ISO code	3-char ISO code	3-digit ISO code	European Post- code prefix	Engine	Assignment level
Afghanistan	AF	AFG	004		G	C, L
Åland Islands	AX	ALA	248	AX	G	C, L

Country/ Region name	2-char ISO code	3-char ISO code	3-digit ISO code	European Post-code prefix	Engine	Assignment level
Albania	AL	ALB	008		G	C, L
Algeria	DZ	DZA	012		G	C, L
American Samoa	AS	ASG	016		U	C, L, Pn, Pr, S
					G	C, L
Andorra	AD	AND	020	AND	G	C, L
Angola	AO	AGO	024		G	C, L
Anguilla	AI	AIA	660		G	C, L
Antarctica	AQ	ATA	010		G	C
Antigua and Barbuda	AG	ATG	028		G	C, L
Argentina	AR	ARG	032		G	C, L
Armenia	AM	ARM	051		G	C, L
Aruba	AW	ABW	533		G	C, L
Australia	AU	AUS	036		G	C, L, Pn, Pr, S
Austria	AT	AUT	040	A	G	C, L, Pn, Pr, S
Azerbaijan	AZ	AZE	031		G	C, L
Bahamas	BS	BHS	044		G	C, L
Bahrain	BH	BHR	048		G	C, L
Bangladesh	BD	BGD	050		G	C, L
Barbados	BB	BRB	052		G	C, L
Belarus	BY	BLR	112		G	C, L
Belgium	BE	BEL	056	B	G	C, L, Pn, Pr
Belize	BZ	BLZ	084		G	C, L
Benin	BJ	BEN	204		G	C, L
Bermuda	BM	BMU	060		G	C, L
Bhutan	BT	BTN	064		G	C, L
Bolivia, Plurinational State of	BO	BOL	068		G	C, L
Bonaire, Sint Eustatius, and Saba	BQ	BES	535		G	C, L
Bosnia and Herzegovina	BA	BIH	070		G	C, L
Botswana	BW	BWA	072		G	C, L
Bouvet Island	BV	BVT	074		G	C
Brazil	BR	BRA	076		G	C, L, Pn, Pr

Country/ Region name	2-char ISO code	3-char ISO code	3-digit ISO code	European Post-code prefix	Engine	Assignment level
British Indian Ocean Territory	IO	IOT	086		G	C
British Virgin Islands	VG	VGB	092		G	C, L
Brunei Darus-salam	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	C, L, Pn, Pr, S
					G	C,L
Cape Verde	CV	CPV	132		G	C
Cayman Is-lands	KY	CYM	136		G	C
Central African Republic	CF	CAF	140		G	C, L
Chad	TD	TCD	148		G	C, L
Chile	CL	CHL	152		G	C, L
China	CN	CHN	156		G	C, L, Pn, Pr
Christmas Is-land (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, Demo-cratic Republic of	CD	COD	180		G	C, L
Congo, Repub-lic of	CG	COG	178		G	C, L
Cook Islands	CK	COK	184		G	C, L
Costa Rica	CR	CRI	188		G	C, L

Country/ Region name	2-char ISO code	3-char ISO code	3-digit ISO code	European Post-code prefix	Engine	Assignment level
Côte d'Ivoire (Ivory Coast)	CI	CIV	384		G	C, L
Croatia (Hrvatska)	HR	HRV	191	HR	G	C, L
Cuba	CU	CUB	192		G	C, L
Curaçao	CW	CUW	531		G	C, L
Cyprus	CY	CYP	196	CY	G	C, L
Czechia	CZ	CZE	203	CZ	G	C, L, Pn, Pr
Democratic People's Republic of Korea	KP	PRK	408		G	C, L
Denmark	DK	DNK	208	DK	G	C, L, Pn, Pr
Djibouti	DJ	DJI	262		G	C, L
Dominica	DM	DMA	212		G	C, L
Dominican Republic	DO	DOM	214		G	C, L
Ecuador	EC	ECU	218		G	C, L
Egypt	EG	EGY	818		G	C, L
El Salvador	SV	SLV	222		G	C, L
Equatorial Guinea	GQ	GNQ	226		G	C, L
Eritrea	ER	ERI	232		G	C, L
Estonia	EE	EST	233	EE	G	C, L, Pn, Pr
Ethiopia	ET	ETH	231		G	C, L
Falkland Islands	FK	FLK	238		G	C, L
Faroe Islands (Included in the Denmark data package)	FO	FRO	234	FO	G	C, L, Pn, Pr
Federated States of Micronesia	FM	FSM	583		U G	C, L, Pn, Pr, S C, L
Fiji	FJ	FJI	242		G	C, L
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/ Region name	2-char ISO code	3-char ISO code	3-digit ISO code	European Post-code 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
Guadeloupe (Included in the France data package)	GP	GLP	312		G	C, L, Pn, Pr
Guam	GU	GUM	316		U	C, L, Pn, Pr, S
					G	C, L
Guatemala	GT	GTM	320		G	C, L
Guernsey (Included in the United Kingdom data package)	GG	GGY	831	G	G	C, L, Pn, Pr, S
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

Country/ Region name	2-char ISO code	3-char ISO code	3-digit ISO code	European Post-code prefix	Engine	Assignment level
Holy See (Vatican City State) (Included in the Italy data package)	VA	VAT	336		G	C, L, Pn, Pr
Honduras	HN	HND	340		G	C, L
Hong Kong	HK	HKG	344		G	C, L
Hungary	HU	HUN	348	H	G	C, L, Pn, Pr
Iceland	IS	ISL	352	IS	G	C, L
India	IN	IND	356		G	C, L, Pn, Pr
Indonesia	ID	IDN	360		G	C, L
Iraq	IQ	IRQ	368		G	C, L
Ireland, Republic of	IE	IRL	372	IRL	G	C, L
Islamic Republic of Iran	IR	IRN	364		G	C, L
Isle of Man (Included in the United Kingdom data package)	IM	IMN	833		G	C, L, Pn, Pr, S
Israel	IL	ISR	376		G	C, L
Italy	IT	ITA	380	I	G	C, L, Pn, Pr
Jamaica	JM	JAM	388		G	C, L
Japan	JP	JPN	392		G	C, L, Pn, Pr, S
Jersey (Included in the United Kingdom data package)	JE	JEY	832		G	C, L, Pn, Pr, S
Jordan	JO	JOR	400		G	C, L
Kazakhstan	KZ	KAZ	398		G	C, L
Kenya	KE	KEN	404		G	C, L
Kiribati	KI	KIR	296		G	C, L
Kuwait	KW	KWT	414		G	C, L
Kyrgyzstan	KG	KGZ	417		G	C, L
Lao People's Democratic Republic	LA	LAO	418		G	C, L
Latvia	LV	LVA	428	LV	G	C, L, Pn, Pr

Country/ Region name	2-char ISO code	3-char ISO code	3-digit ISO code	European Post-code prefix	Engine	Assignment level
Lebanon	LB	LBN	422		G	C, L
Lesotho	LS	LSO	426		G	C, L
Liberia	LR	LBR	430		G	C, L
Libya	LY	LBY	434		G	C, L
Liechtenstein (Included in 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 (Now known as North Macedonia)	MK	MKD	807	MK	G	C, L
Madagascar	MG	MDG	450		G	C, L
Malawi	MW	MWI	454		G	C, L
Malaysia	MY	MYS	458	M	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	C, L, Pn, Pr, S
					G	C, L
Martinique (Included in the France data package)	MQ	MTQ	474		G	C, L, Pn, Pr
Mauritania	MR	MRT	478		G	C, L
Mauritius	MU	MUS	480		G	C, L
Mayotte (Included in the France data package)	YT	MYT	175		G	C, L, Pn, Pr
Mexico	MX	MEX	484		G	C, L
Moldova	MD	MDA	498	MD	G	C, L
Monaco (Included in the France data package)	MC	MCO	492	F	G	C, L, Pn, Pr

Country/ Region name	2-char ISO code	3-char ISO code	3-digit ISO code	European Post-code prefix	Engine	Assignment level
Mongolia	MN	MNG	496		G	C, L
Montenegro	ME	MNE	499		G	C, L
Montserrat	MS	MSR	500		G	C, L
Morocco	MA	MAR	504		G	C, L
Mozambique	MZ	MOZ	508		G	C, L
Myanmar	MM	MMR	104		G	C, L
Namibia	NA	NAM	516		G	C, L
Nauru	NR	NRU	520		G	C, L
Nepal	NP	NPL	524		G	C, L
Netherlands	NL	NLD	528	NL	G	C, L, Pn, Pr
New Caledonia (Included in the France data package)	NC	NCL	540		G	C, L, Pn, Pr
New Zealand	NZ	NZL	554		G	C, L, Pn, Pr, S
Nicaragua	NI	NIC	558		G	C, L
Niger	NE	NER	562		G	C, L
Nigeria	NG	NGA	566		G	C, L
Niue	NU	NIU	570		G	C, L
Norfolk Island (Included in the Australia data package)	NF	NFK	574		G	C, L
Northern Mariana Islands	MP	MNP	580		U G	C, L, Pn, Pr, S C, L
Norway	NO	NOR	578	N	G	C, L, Pn, Pr
Occupied Palestinian Territory	PS	PSE	275		G	C
Oman	OM	OMN	512		G	C, L
Pakistan	PK	PAK	586		G	C, L
Palau	PW	PLW	585		U G	C, L, Pn, Pr, S C, L
Panama	PA	PAN	591		G	C, L
Papua New Guinea	PG	PNG	598		G	C, L
Paraguay	PY	PRY	600		G	C, L

Country/ Region name	2-char ISO code	3-char ISO code	3-digit ISO code	European Post-code prefix	Engine	Assignment level
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
Puerto Rico	PR	PRI	630		U	C, L, Pn, Pr, S
					G	C, L
Qatar	QA	QAT	634		G	C, L
Republic of Korea	KR	KOR	410		G	C, L
Réunion (Included in the France data package)	RE	REU	638		G	C, L, Pn, Pr
Romania	RO	ROU	642	RO	G	C, L
Russian Federation	RU	RUS	643	RUS	G	C, L
Rwanda	RW	RWA	646		G	C, L
Saint Barthélemy (Included in the France data package)	BL	BLM	652		G	C, L
Saint Helena, Ascension, and Tristan da Cunha	SH	SHN	654		G	C, L
Saint Kitts and Nevis	KN	KNA	659		G	C, L
Saint Lucia	LC	LCA	662		G	C, L
Saint Martin (French part) (Included in the France data package)	MF	MAF	663		G	C, L
Saint Pierre and Miquelon (Included in the France data package)	PM	SPM	666		G	C, L, Pn, Pr
Saint Vincent & Grenadines	VC	VCT	670		G	C, L

Country/ Region name	2-char ISO code	3-char ISO code	3-digit ISO code	European Post-code prefix	Engine	Assignment level
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 (Dutch Part)	SX	SXM	534		G	C, L
Slovakia	SK	SVK	703		G	C, L, Pn, Pr
Slovenia	SI	SVN	705		G	C, L
Solomon Islands	SB	SLB	090		G	C, L
Somalia	SO	SOM	706		G	C, L
South Africa	ZA	ZAF	710		G	C, L
South Georgia and the South Sandwich Islands	GS	SGS	239		G	C, L
South Sudan	SS	SDN	728		G	C, L
Spain	ES	ESP	724	E	G	C, L, Pn, Pr
Sri Lanka	LK	LKA	144		G	C, L
Sudan	SD	SDN	736		G	C, L
Suriname	SR	SUR	740		G	C, L
Svalbard and Jan Mayen (Included in the Norway data package)	SJ	SJM	744		G	C
Swaziland	SZ	SWZ	748		G	C, L
Sweden	SE	SWE	752	S	G	C, L, Pn, Pr
Switzerland	CH	CHE	756	CH	G	C, L, Pn, Pr

Country/ Region name	2-char ISO code	3-char ISO code	3-digit ISO code	European Post-code prefix	Engine	Assignment level
Syrian Arab Republic	SY	SYR	760		G	C, L
Taiwan	TW	TWN	158		G	C, L
Tajikistan	TJ	TJK	762		G	C, L
Thailand	TH	THA	764		G	C, L
Timor-Leste	TL	TLS	626		G	C
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
U.S. Virgin Islands	VI	VIR	850		U G	C, L, Pn, Pr, S
Uganda	UG	UGA	800		G	C, L
Ukraine	UA	UKR	804	UK	G	C, L
United Arab Emirates	AE	ARE	784		G	C, L
United Kingdom	GB	GBR	826	GB	G	C, L, Pn, Pr, S
United Republic of Tanzania	TZ	TZA	834		G	C, L
United States	US	USA	840		U G	C, L, Pn, Pr, S C, L
United States Minor Outlying Islands	UM	UMI	581		U G	C, L, Pn, Pr, S
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

Country/ Region name	2-char ISO code	3-char ISO code	3-digit ISO code	European Post-code prefix	Engine	Assignment level
Wallis and Futuna	WF	WLF	876		G	C, L, Pn, Pr
Western Sahara	EH	ESH	732		G	C, L
Yemen	YE	YEM	887		G	C, L
Zambia	ZM	ZMB	894		G	C, L
Zimbabwe	ZW	ZWE	716		G	C, L

3.5.13.4 Countries supported by lastline drilldown

When you set *Enable Lastline Drilldown* to *YES*, indicate either a country name or the two-character ISO country code.

> **i Note**
>
> Reference to "country" or "countries" in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

Use either the English country name or the two-character ISO country code as input. For some countries, enter the country name in its native script. When the input country name is in its native script, the software also provides the drilldown list in the native script.

The Global engine supports the *Enable Lastline Drilldown* option. Lastline drilldown isn't available for the Canada and USA engines.

Country name	Two-character ISO code	Country name in native script
Afghanistan	AF	
Åland Islands	AX	
Albania	AL	
Algeria	DZ	
American Samoa	AS	
Andorra	AD	
Angola	AO	
Antigua and Barbuda	AG	
Argentina	AR	
Armenia	AM	
Australia	AU	
Austria	AT	
Azerbaijan	AZ	

Country name	Two-character ISO code	Country name in native script
Bahamas	BS	
Bahrain	BH	
Bangladesh	BD	
Barbados	BB	
Belarus	BY	
Belgium	BE	
Belize	BZ	
Benin	BJ	
Bermuda	BM	
Bhutan	BT	
Bolivarian Republic of Venezuela	VE	
Bonaire, Sint Eustatius, and Saba	BQ	
Botswana	BW	
Brazil	BR	
British Virgin Islands	VG	
Brunei Darussalam	BN	
Bulgaria	BG	
Burkina Faso	BF	
Burundi	BI	
CABO VERDE	CV	
Cambodia	KH	
Cameroon	CM	
Canada	CA	
Cayman Islands	KY	
Central African Republic	CF	
Chad	TD	
Chile	CL	
China	CN	中国
Christmas Island	CX	
Cocos (Keeling) Islands	CC	
Colombia	CO	
Comoros	KM	
Congo	CG	
Costa Rica	CR	

Country name	Two-character ISO code	Country name in native script
Côte d'Ivoire	CI	
Croatia	HR	
Cuba	CU	
Curaçao	CW	
Cyprus	CY	
Czechia	CZ	
Democratic People's Republic of Korea	KP	
Denmark	DK	
Djibouti	DJ	
Dominica	DM	
Dominican Republic	DO	
Ecuador	EC	
Egypt	EG	
El Salvador	SV	
Equatorial Guinea	GQ	
Eritrea	ER	
Estonia	EE	
Ethiopia	ET	
Faroe Islands	FO	
Federated States of Micronesia	FM	
Fiji	FJ	
Finland	FI	
France	FR	
French Guiana	GF	
French Polynesia	PF	
Gabon	GA	
Gambia	GM	
Georgia	GE	
Germany	DE	
Ghana	GH	
Greece	GR	ΕΛΛΆΔΑ
Greenland	GL	
Grenada	GD	
Guadeloupe	GP	

Country name	Two-character ISO code	Country name in native script
Guam	GU	
Guatemala	GT	
Guernsey	GG	
Guinea	GN	
Guinea-Bissau	GW	
Guyana	GY	
Haiti	HT	
Holy See (Vatican City State)	VA	
Honduras	HN	
Hungary	HU	
Iceland	IS	
India	IN	
Indonesia	ID	
Iraq	IQ	
Ireland	IE	
Islamic Republic of Iran	IR	
Isle of Man	IM	
Italy	IT	
Jamaica	JM	
Japan	JP	日本
Jersey	JE	
Jordan	JO	
Kazakhstan	KZ	
Kenya	KE	
Kiribati	KI	
Kyrgyzstan	KG	
Lao People's Democratic Republic	LA	
Latvia	LV	
Lebanon	LB	
Lesotho	LS	
Liberia	LR	
Libya	LY	
Liechtenstein	LI	
Lithuania	LT	

Country name	Two-character ISO code	Country name in native script
Luxembourg	LU	
Madagascar	MG	
Malawi	MW	
Malaysia	MY	
Maldives	MV	
Mali	ML	
Malta	MT	
Marshall Islands	MH	
Martinique	MQ	
Mauritania	MR	
Mauritius	MU	
Mayotte	YT	
Mexico	MX	
Monaco	MC	
Mongolia	MN	
Montenegro	ME	
Montserrat	MS	
Morocco	MA	
Mozambique	MZ	
Myanmar	MM	
Namibia	NA	
Nauru	NR	
Nepal	NP	
Netherlands	NL	
New Caledonia	NC	
Nicaragua	NI	
Niger	NE	
Nigeria	NG	
Norfolk Island	NF	
Northern Mariana Islands	MP	
Norway	NO	
Oman	OM	
Pakistan	PK	
Palau	PW	

Country name	Two-character ISO code	Country name in native script
Panama	PA	
Papua New Guinea	PG	
Paraguay	PY	
Peru	PE	
Philippines	PH	
Plurinational State of Bolivia	BO	
Poland	PL	
Portugal	PT	
Puerto Rico	PR	
Qatar	QA	
Republic of Korea	KR	대한민국
Republic of Moldova	MD	
Réunion	RE	
Romania	RO	
Russian Federation	RU	РОССИЯ
Rwanda	RW	
Saint Barthélemy	BL	
Saint Helena, Ascension, and Tristan da Cunha	SH	
Saint Kitts and Nevis	KN	
Saint Lucia	LC	
Saint Martin (French part)	MF	
Saint Pierre and Miquelon	PM	
Saint Vincent and the Grenadines	VC	
Samoa	WS	
San Marino	SM	
Sao Tome and Principe	ST	
Saudi Arabia	SA	
Senegal	SN	
Seychelles	SC	
Sierra Leone	SL	
Sint Maarten (Dutch Part)	SX	
Slovakia	SK	
Slovenia	SI	
Solomon Islands	SB	

Country name	Two-character ISO code	Country name in native script
Somalia	SO	
South Africa	ZA	
South Sudan	SS	
Spain	ES	
Sri Lanka	LK	
Sudan	SD	
Suriname	SR	
Svalbard and Jan Mayen	SJ	
Swaziland	SZ	
Sweden	SE	
Switzerland	CH	
Syrian Arab Republic	SY	
Taiwan	TW	台湾
Tajikistan	TJ	
Thailand	TH	
The Democratic Republic of the Congo	CD	
Togo	TG	
Tonga	TO	
Trinidad and Tobago	TT	
Tunisia	TN	
Turkey	TR	
Turkmenistan	TM	
Turks and Caicos Islands	TC	
Tuvalu	TV	
U.S. Virgin Islands	VI	
Uganda	UG	
Ukraine	UA	
United Arab Emirates	AE	
United Kingdom	GB	
United Republic of Tanzania	TZ	
United States	US	
Uruguay	UY	
Uzbekistan	UZ	
Vanuatu	VU	

Country name	Two-character ISO code	Country name in native script
Viet Nam	VN	
Wallis and Futuna	WF	
Yemen	YE	
Zambia	ZM	
Zimbabwe	ZW	

Related Information

Global Address Cleanse Suggestion List Options [page 526]

3.5.13.5 Global Address Cleanse information codes

Global Address Cleanse information codes explain why an address is unassigned.

Information codes are output in the Global Address Cleanse Info_Code output field. Information codes have four characters. The following table explains the seven classification levels for Global Address Cleanse information codes.

Seven classification levels

Classification level	Represents
1000	Input record discrepancies
2000	Inconsistent last line information
3000	Inconsistent address information
4000	Inconsistent secondary address information
5000	All other types of information
6000	Unclassified error
7000	Aborted process caused by a user-defined timeout

Each information code is available based on the engine or engines that you enable. The following table shows the code used for each engine.

Key for engines

Engine	Code
Canada	C
Global Address	G
USA	U
All engines	C, G, and U

Engine	Code
Transform Level (information code does not come from a specific engine	T

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/regions.	T
1030	No country/region found by Country ID or no country/region 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/region 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
2070	Bad or no postcode for locality; multiple postcodes are available.	G
3000	Locality, region, and postcode are valid. Unable to identify the primary address line.	All engines
3010	Locality, region, and postcode are valid. Unable to match primary name to directory.	All engines
3020	Possible primary name matches are too close to choose one.	All engines
3030	Primary range is missing on input or not in the directory.	All engines

Information code	Description	Engine(s)
3050	An invalid or missing primary type is preventing a possible address match.	All engines
3060	A missing primary type and prefix/postfix (directional) are 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 address is not in the directory.	G
3220	Possible building names are too close to choose one.	G
3250	The building name is missing on input or not in the directory, or the range is missing on input or invalid with the input building.	G
3300	The postcode only lookup returned multiple primary names.	G
4000	The secondary information is missing on input or not in the directory.	All engines
4010	Possible secondary address line matches are too close to choose one.	All engines
4500	The organization is missing on input or not in the directory.	G
4510	The organization 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/region.	C, U

Information code	Description	Engine(s)
5020	The entire input address was blank.	T
5030	The country postal authority will not permit assignment due to violation of an assignment rule.	G
5040	Assignment not made after lastline processing due to a violation of lastline assignment rules.	G
5050	The obsolete address can be mapped to multiple addresses.	G
6000	Unclassified error.	All engines
7000	The user-defined processing timeout threshold, set in the *Processing Timeout* option, has been reached, preventing further corrective action on this address.	G

3.5.13.6 Global Address Cleanse status codes

Global Address Cleanse status codes represent the corrections made to the address during processing.

> **i Note**
>
> Reference to "country" or "countries" in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

Status codes appear in the *Status_Code* output field. They're five or six characters based on the engine used for processing.

Each position in the status code represents a specific category for which the code applies.

Position categories

Position	Category
First character	S for Status
Second character	Last line corrections
Third character	Address line corrections
Fourth character	Secondary address line correction
Fifth character	Other primary or secondary address component changes (components that aren't considered basic address components)
Sixth character	Additional information about a record that isn't related to an address change

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.
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 or 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 or post directional.
2	Corrected primary type.
3	Corrected pre or post directional and primary type.
4	Corrected primary name.
5	Corrected pre or post directional and primary name.
6	Corrected primary type and primary name.

Value	Description
7	Corrected pre or post directional, primary type, and primary name.
8	Corrected primary range.
9	Corrected pre or post directional and primary range.
A	Corrected primary type and primary range.
B	Corrected pre or post directional, primary type, and primary range.
C	Corrected primary name and primary range.
D	Corrected pre or post directional, primary name, and primary range.
E	Corrected primary type, primary name, and primary range.
F	Corrected pre or 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 components (unit description, floor description, stairwell description, or wing description).
2	Corrected one or more secondary address components (unit number, floor number, stairwell name, or wing name).
3	Corrected one or more secondary address components (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 components (unit description, floor description, stairwell description, or wing description), and building name.
6	Corrected the one or more secondary address components (unit number, floor number, stairwell name, or wing name), and building name.

Value	Description
7	Corrected one or more secondary address components (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 components (unit description, floor description, stairwell description, or wing description), and firm.
A	Corrected one or more secondary address components (unit number, floor number, stairwell name, or wing name), and firm.
B	Corrected one or more secondary address components (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 components (unit description, floor description, stairwell description, or wing description), building name, and firm.
E	Corrected one or more secondary address components (unit number, floor number, stairwell name, or wing name), building name, and firm.
F	Corrected one or more secondary address components (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 aren't considered basic address components such as other primary address and secondary address information.

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.
4	Changed the point of reference.
5	Changed the Other Primary Address components and the point of reference.
6	Changed the Other Secondary Address components and the point of reference.
7	Changed the Other Primary Address, Other Secondary Address components, and the point of reference.

Sixth character

The value of the sixth character indicates additional information about a record that isn't related to a change in the address.

Value	Description
A	Alias record used for assignment. Global Address engine.
B	Base record assignment. Global Address engine (New Zealand). A Bordering Locality. Global Address engine (Australia).
C	An Alias and a Bordering locality. Global Address engine (Australia).
D	Deleted record. Global Address engine (Austria and Germany).
E	An excluded postal code was found. Canada engine.
I	Record ignored. Global Address engine (New Zealand).
L	Large Volume Receiver (LVR) and Large Organization Postal Code. Global Address engine (Brazil and Japan)
P	A software generated default locality level postcode was assigned. Global Address engine (China).
Q	An alias and a software generated default locality level postcode was assigned. Global Address engine (China).
U	Unique address. Global Address engine (New Zealand).

3.5.13.7 Global Address Cleanse quality codes

Global Address Cleanse quality codes relay additional information about address quality.

> **i Note**
>
> Reference to “country” or “countries” in this topic refers to a country, territory, or geographical area, like Antarctica, based on the context.

Quality codes appear in the *Quality_Code* output field. 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 isn’t an information code

Use the following table to determine the code assigned to the *Quality_Code* output field.

Quality code	Description
Q1	Perfect address on input. All address components were validated without corrections.
Q2	Corrected address. All address components were validated after corrections were made.
Q3	Not all components of the address could be fully validated. There was insufficient information to make a final correction. However, the assessment of the record leads to the assumption that there’s 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’s 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’s a "small" likelihood that this address is deliverable.
Q6	Not all components of the address could be fully validated. There was insufficient information to make a final correction. However, the assessment of the record leads to the assumption that it is "highly unlikely" that this address is deliverable.

3.5.13.8 USA Regulatory Address Cleanse fault codes

Fault codes describe why the USA Regulatory Address Cleanse transform could not assign an address.

The transform outputs fault codes to the *Fault_Code* output field.

Fault Code	Description
E101	Last line is bad or missing.
E212	No locality and bad postal code.
E213	Bad locality, valid region, and no postal code.
E214	Bad locality and bad postal code.
E216	Bad postal code and can't determine which locality match to select.
E302	No primary address line parsed.
E412	Primary name not found in directory.
E413	Possible primary name matches are too close to choose one.
E420	Primary range is missing.
E421	Primary range is invalid for the street/route/building.
E422	Primary prefix needed, input is wrong or missing.
E423	Primary type needed, input is wrong or missing.
E425	Primary type and directional needed, input is wrong or missing.
E427	Primary postfix needed, input is wrong or missing.
E428	Bad postal code, can't select an address match.
E429	Bad locality, can't select an address match.
E430	Possible address-line matches too close to choose one.
E431	Locality2 needed, input is wrong or missing.
E439	Exact match made in EWS directory.
E500	Other error.
E501	Foreign address.
E502	Input record entirely blank.
E503	Postal code not in area covered by partial USPS directory.
E504	Overlapping ranges in USPS directory.
E505	Address does not exist in the USPS directories. Undeliverable address.
E600	Marked by USPS as unsuitable for delivery of mail.
E601	The primary address number did not DPV confirm, and the Postcode2 was removed.

3.5.13.9 USA Regulatory Address Cleanse status codes

The USA Regulatory Address Cleanse transform assigns status codes to assigned addresses to indicate how the input address differs from the assigned address.

The transform outputs status codes to the *Status_Code* output field.

Digit	Description
1st	*A*: The transform truncated the *Address_Line* to make it fit your field. *B*: The transform truncated both the *Address_Line* and the *Locality1_Name*. *C*: The transform truncated the *Locality1_Name* to make it fit your field. *S*: No truncation occurred.
2nd	*0*: Regarding the *Locality1_Name*, *Region1*, *Postcode1*, and *Postcode2*, there is no significant difference between the input data and the data the transform assigned. *1*: The transform assigned a different *Postcode1*. *2*: The transform assigned a different *Locality1_Name*. *3*: The transform assigned a different *Locality1_Name* and *Postcode1*. *4*: The transform assigned a different *Region1*. *5*: The transform assigned a different *Region1* and *Postcode1*. *6*: The transform assigned a different *Locality1_Name* and *Region1*. *7*: The transform assigned a different *Locality1_Name*, *Region1*, and *Postcode1*. *8*: The transform assigned a different *Postcode2*. *9*: The transform assigned a different *Postcode1* and *Postcode2*. *A*: The transform assigned a different *Locality1_Name* and *Postcode2*. *B*: The transform assigned a different *Locality1_Name*, *Postcode1*, and *Postcode2*. *C*: The transform assigned a different *Region1* and *Postcode2*. *D*: The transform assigned a different *Region1*, *Postcode1*, and *Postcode2*. *E*: The transform assigned a different *Locality1_Name*, *Region1*, and *Postcode2*. *F*: The transform assigned a different *Locality1_Name*, *Region1*, *Postcode1*, and *Postcode2*.

Digit	Description
3rd	*0*: Regarding the *Primary_Name*, *Primary_Prefix* or *Primary_Postfix*, and *Primary_Type*, there is no significant difference between the input and what the transform assigned. *1*: The transform assigned a different *Primary_Type*. *2*: The transform assigned a different *Primary_Prefix*. *3*: The transform assigned a different *Primary_Type* and *Primary_Prefix*. *4*: The transform assigned a different *Primary_Postfix*. *5*: The transform assigned a different *Primary_Type* and *Primary_Postfix*. *6*: The transform assigned a different *Primary_Prefix* and *Primary_Postfix*. *7*: The transform assigned a different *Primary_Prefix*, *Primary_Type*, and *Primary_Postfix*. *8*: The transform assigned a different *Primary_Name*. *9*: The transform assigned a different *Primary_Name* and *Primary_Type*. *A*: The transform assigned a different *Primary_Prefix* and *Primary_Name*. *B*: The transform assigned a different *Primary_Prefix*, *Primary_Name*, and *Primary_Type*. *C*: The transform assigned a different *Primary_Name* and *Primary_Postfix*. *D*: The transform assigned a different *Primary_Name*, *Primary_Type*, and *Primary_Postfix*. *E*: The transform assigned a different *Primary_Prefix*, *Primary_Name*, and *Primary_Postfix*. *F*: The transform assigned a different *Primary_Prefix*, *Primary_Name*, *Primary_Postfix*, and *Primary_Type*.

Digit	Description
4th	*0*: Regarding the *County_Number*, *Sortcode_Route*, *Delivery_Point*, and *Unit_Description*, there is no significant difference between the input data and the data the transform assigned. *1*: The transform assigned a different *Unit_Description*. *2*: The transform assigned a different *Delivery_Point*. *3*: The transform assigned a different *Delivery_Point* and *Unit_Description*. *4*: The transform assigned a different *Sortcode_Route*. *5*: The transform assigned a different *Sortcode_Route* and *Unit_Description*. *6*: The transform assigned a different *Sortcode_Route* and *Delivery_Point*. *7*: The transform assigned a different *Sortcode_Route*, *Delivery_Point*, and *Unit_Description*. *8*: The transform assigned a different *County_Number*. *9*: The transform assigned a different *County_Number* and *Unit_Description*. *A*: The transform assigned a different *County_Number* and *Delivery_Point*. *B*: The transform assigned a different *County_Number*, *Delivery_Point*, and *Unit_Description*. *C*: The transform assigned a different *County_Number* and *Sortcode_Route*. *D*: The transform assigned a different *County_Number*, *Sortcode_Route*, and *Unit_Description*. *E*: The transform assigned a different *County_Number*, *Sortcode_Route*, and *Delivery_Point*. *F*: The transform assigned a different *County_Number*, *Sortcode_Route*, *Delivery_Point*, and *Unit_Description*.

Digit	Description
5th	*0*: Regarding the *LOT*, *LOT_Order*, and*Locality2_Official*, there is no significant difference between the input data and the data the transform assigned. *1*: The transform assigned a different *LOT*. *2*: The transform assigned a different *LOT_Order*. *3*: The transform assigned a different *LOT* and *LOT_Order*. *4*: The transform assigned a different *Locality2_Official*. *5*: The transform assigned a different *Locality2_Official* and *LOT*. *6*: The transform assigned a different *Locality2_Official* and *LOT_Order*. *7*: The transform assigned a different *Locality2_Official*, *LOT*, and *LOT_Order*.
6th	Always outputs a zero (*0*).

3.5.13.10 About ShowA and ShowL for USA addresses

Use the ShowA and ShowL utilities to question the postal directories about address assignment.

Find answers to questions such as the following:

- Why did the transform standardized the address in an unexpected way?
- Why didn't the transform assign the address?
- Why did the transform error code indicate a flaw in the directory?

Use ShowA to display or output information from the Address_1_Directory. Use ShowL to query the City_Directory and the Post_Code_Directory.

Run the ShowA and ShowL utilities from a DOS command line using specific command-line options. These options are listed when you enter the following command:

Windows USA ShowA: `showa /op`

UNIX USA ShowA: `showa -op`

> **i Note**
>
> For addresses in Canada, use `cashowa`.

The Show configuration files

Each Show utility has a configuration file. The configuration files contain parameters for controlling how the utility works. The following tables contain the configuration file names and locations for each utility.

For 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

For Canada addresses

Utility	Executable	File name	Location
ShowA	cashowa.exe	cashowa.cfg	LINK_DIR \dataquality\gac
ShowL	cashowl.exe	cashowl.cfg	LINK_DIR \dataquality\gac

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.

> **i Note**
>
> Run the ShowA and ShowL utilities in the same directory as the ShowA and ShowL configuration files. The utilities do not run when they are located outside of the default location.

3.5.13.11 ShowA and ShowL query modes

Enter queries in the Show utilities using one of two modes: Prompts and commands.

Prompts

If you type the ShowA or ShowL command without any commands, the utility prompts you to enter your query data. The utility takes all other information and options from the configuration file.

After your first query, the utility prompts you to enter the next query. When you are finished entering queries, exit the utility by typing “Quit”.

Commands

When you use commands in the ShowA and ShowL utilities, the utility uses information from the configuration file. For example, the configuration file contains information about the auxiliary files, the output file, display options, and search options. To override any value taken from the configuration file, use command-line options selectively. Depend on the configuration file for any other overrides. The only values that you cannot specify through a command line is the optional output fields.

> **i Note**
>
> If the utility detects any command, it determines that you are operating in command-line mode. Therefore, the utility does not prompt you to enter query data as it does when you use the prompts mode.

3.5.13.12 USA ShowA command-line options

Use commands to control how the ShowA utility behaves.

Use the following command to view a summary of command-line options in the utility:

Windows: `showa /op`

UNIX: `showa -op`

The following table contains the commands and descriptions for both Windows and UNIX.

UNIX	Windows	Description
-a	/a	Appends information to the output file (when it 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 commands.
-p	/p	Pauses screen display every 22 lines.
-2:dpbc	/2:dpbc	Enter the DPBC code for `dpbc`.
-4:zip4	/4:zip4	Enter the postcode2 for `zip4`.
-ad:file	/ad:file	Enter the Address-line dictionary `addrln.dct)` and path name for `file`.
-c:cart	/c:cart	Enter the carrier route number for `cart`.
-f:file	/f:file	Enter the file path and name of the output file for `file`. The output file stores the information from the query instead of displaying it on screen.
-nd:file	/nd:file	Enter the National ZIP+4 directory path and name, `zip4us.dir` for `file`.
-pre:dir	/pre:dir	Enter the primary prefix, such as N, NE, E, SE, S, SW, W, NW, for `dir`.
-pos:dir	/pos:dir	Enter the primary postfix, such as N, NE, E, SE, S, SW, W, NW, for `dir`.

UNIX	Windows	Description
-s:street	/s:street	Enter the street primary name using quotes if there are multiple words, for `street`.
-sfx:suffix	/sfx:suffix	Enter the primary type, such as Ave, Blvd, St, Rd, for `suffix`.
-sh:range	/sh:range	Enter the street primary range high for `range`.
-sl:range	/sl:range	Enter the street primary range low or exact for `range`.
-t:type	/t:type	Enter the file type dBASE3, ASCII, or DELIMITED, for `type`.
-u:urb idx	/u:urb idx	Enter the urbanization Index for `urb idx`.
-z:lo-hi	/z:lo-hi	Enter the low and high range for Post-code1 for `lo-hi`.
-z:zip	/z:zip	Enter the Postcode1 for `zip`.

Related Information

ShowA and ShowL query modes [page 823]

3.5.13.13 USA ShowL command line options

Use commands to control how the ShowL utility behaves.

Use the following command to view a summary of command line options in the utility:

Windows: `showl /op`

UNIX: `showl -op`

The following table contains the commands and descriptions for both Windows and UNIX.

UNIX	Windows	Description
-a	/a	Appends query information to an output file when it exists.
-d	/d	Displays query data on screen.
-op	/op	Displays the list of commands.
-p	/p	Pauses screen display every 22 lines.
-ab:query	/ab:query	Enter the abbreviated Locality1 for `query`.

UNIX	Windows	Description
-cd:file	/cd:file	Enter the City directory path and name (**city04.dir**) for `file`.
-cn:city	/cn:city	Enter Locality1 name, using quotes if there are multiple words, for `city`.
-dr:dir	/dr:dir	Enter the directory to search, City or ZCF, for `dir`.
-f:file	/f:file	Enter the output file path and name for `file`.
-ml:query	/ml:query	Enter the military Postcode1 for `query`.
-mz:query	/mz:query	Enter the multi-zone Locality1 for `query`.
-pn:query	/pn:query	Enter a place name for `query`.
-st:state	/st:state	Enter Region1 for `state`. For U.S., use USPS abbreviations or full state names.
-t:type	/t:type	Enter the output file type, dBASE3, ASCII, or DELIMITED, for `type`.
-un:query	/un:query	Enter the unique Postcode1 for `query`.
-z:zip	/z:zip	Enter the Postcode1 for `zip`.
-zd:file	/zd:file	Enter the ZCF directory path and name (**zcf04.dir**) for `file`.
-z:lo-hi	/z:lo-hi	Enter the Postcode1 range for `lo-hi`.

Related Information

ShowA and ShowL query modes [page 823]

3.5.14 Data Cleanse reference

The Data Cleanse reference contains additional information about the Data Cleanse transform processes.

The Data Cleanse transform generates data quality statistics tables in the repository. These statistics provide insight into how the software cleansed and assigned your data by generating aggregated summary statistics and specific record-level statistics.

Data Cleanse status codes [page 827]
The Data Cleanse transform generates status codes that indicate how it standardized your data, or how a specific standard could be used to standardize your data.

Data Cleanse person parser status codes [page 829]
Person parser status codes indicate how the Data Cleanse transform standardized person information in your records.

Data Cleanse firm parser status codes [page 830]
Firm parser status codes indicate how the transform parsed firm information in your records.

Data Cleanse Date, Phone, and SSN parser status codes [page 831]
Date, Phone, and Social Security number parser status codes indicate how the transform standardized this information in your records.

3.5.14.1 Data Cleanse status codes

The Data Cleanse transform generates status codes that indicate how it standardized your data, or how a specific standard could be used to standardize your data.

To output the Data Cleanse status codes, include the *Status_Code* output field in your job setup. The transform outputs status codes using the following format: `<parser name>_<output field name>`_STD or `<parser name>_<output field name>`_ADD. The value in the `<parser name>` variable can include one of the following parsers:

- custom
- date
- firm
- person
- phone
- SSN

The transform generates only the status codes for records that have standardized output.

Status code exceptions

The transform does not generate a status code for minor changes. The following table contains exceptions that explain why the transform does not generate status codes.

Status code exception descriptions

Exception	Description	Example
Trailing period	Transform does not generate a status code when the transform adds or removes a period after a piece of data.	**Example** The transform does not generate a status code for an input of J Smith. The transform is set to add a period after single letters so it changes “J” to “J.” **Example** The transform generates a status code: UK has a standard of "U.K." The transform generates a status code because there are more punctuation changes than a single trailing period.
Case	Transform does not generate a status code when the only difference between the standard and the original data is how the data is cased.	**Example** The last name Mcdonald has a standard of "McDonald" but, the transform outputs it as "MCDONALD" because you have the *Capitalization* option set to *Upper*. No status code.
Remove punctuation	Transform does not generate a status code when the only difference between the standard and the original data is that the punctuation is removed. Standardization option *Remove Punctuation* set to *Yes*.	**Example** UK has a standard of "U.K." However, the *Remove Punctuation* option is set to *Yes*. No status code.
Diacritical characters	The transform does not generate a status code when the only difference between the standard and the original data is the diacritical characters are removed. Standardization option *Remove Diacritical CharactersYes*.	**Example** Hernandez has a standard of "Hernández". However, the *Remove Diacritical Characters* option is set to *Yes*. No status code. **Example** Hernandez has a standard of "Hernández". However, the *Remove Diacritical Characters* option is set to *No*. Transform generates a status code to show that a standard was used.

Parent topic: Data Cleanse reference [page 826]

Related Information

Data Cleanse person parser status codes [page 829]
Data Cleanse firm parser status codes [page 830]

3.5.14.2 Data Cleanse person parser status codes

Person parser status codes indicate how the Data Cleanse transform standardized person information in your records.

Person parser status codes have the following formats:

- PERSON#_`<output_field>`_STD
- PERSON#_`<output_field>`_ADD

The transform replaces the pound sign (#) with the person parse.

Person parser status code descriptions

Status code	Description
PERSON#_NAME_DESIGNATOR_STD	Output *NAME_DESIGNATOR* using the Standardized field class.
PERSON#_PRENAME_ADD	Assigned *PRENAME*
PERSON#_PRENAME _STD	Output *PRENAME* using the *STANDARDIZED FIELD_CATEGORY*.
PERSON#_GIVEN_NAME1_STD	Output *GIVEN_NAME1* using the *STANDARDIZED FIELD_CATEGORY*.
PERSON#_GIVEN_NAME2_STD	Output *GIVEN_NAME2* using the *STANDARDIZED FIELD_CATEGORY*.
PERSON#_FAMILY_NAME1_STD	Output *FAMILY_NAME1* using the *STANDARDIZED FIELD_CATEGORY*.
PERSON#_FAMILY_NAME2_STD	Output *FAMILY_NAME2* using the *STANDARDIZED FIELD_CATEGORY*.
PERSON#_HONORARY_POSTNAME_STD	Output *HONORARY_POSTNAME* using the *STANDARDIZED FIELD_CATEGORY*.
PERSON#_MATURITY_POSTNAME_STD	Output *MATURITY_POSTNAME* using the *STANDARDIZED FIELD_CATEGORY*.
PERSON#_TITLE_STD	Output *TITLE* using the *STANDARDIZED FIELD_CATEGORY*.
PERSON#_NAME_SPECIAL_STD	Output *NAME_SPECIAL* using the *STANDARDIZED FIELD_CATEGORY*.

The transform may assign multiple status codes for a record. The transform outputs multiple status codes separated by commas.

Example

Status code output string:

```
PERSON1_PRENAME_STD,PERSON1_GIVEN_NAME1_STD,PERSON1_FAMILY_NAME1_STD,PERSON2_GIVE
N_NAME2_STD,PERSON6_MATURITY_POSTNAME_STD
```

The status code string provides the following information:

- The Person1 parse used standards for *PRENAME*, *GIVEN_NAME1*, and *FAMILY_NAME1* output fields.
- The Person2 parse used standards for the *GIVEN_NAME2* output field.
- The Person6 parse used standards for the *MATURITY_POSTNAME* output field.

Parent topic: Data Cleanse reference [page 826]

Related Information

Data Cleanse status codes [page 827]
Data Cleanse firm parser status codes [page 830]
Data Cleanse Date, Phone, and SSN parser status codes [page 831]
Data Cleanse custom parser status codes [page 832]
Data Cleanse information codes [page 833]
Data Cleanse information code prefixes [page 833]
Data Cleanse information code descriptions [page 834]
Data Cleanse diacritical character conversions [page 839]

3.5.14.3 Data Cleanse firm parser status codes

Firm parser status codes indicate how the transform parsed firm information in your records.

Firm parser status codes have the format FIRM#_`<output_field>`_STD. The transform replaces the pound sign (#) with the firm number parse.

Firm parser status code descriptions

Status code	Description
Firm#_Firm_Std	Output *Firm* using the Standardized field class.
Firm#_Firm_Location_Std	Output *Firm_Location* using the Standardized field class.

The transform may assign multiple status codes per record. The transform separates multiple status codes with commas.

Example

The status code:

`FIRM1_FIRM_STD,FIRM2_FIRM_LOCATION_STD,FIRM3_FIRM_STD,FIRM3_FIRM_LOCATION_STD`
provides the following information:

- The Firm1 parse used a *Firm* standard.
- The Firm2 parse used a *Firm_Location* standard.
- The Firm3 parse used both a *Firm* standard and a *Firm_Location* standard.

Parent topic: Data Cleanse reference [page 826]

Related Information

Data Cleanse status codes [page 827]
Data Cleanse person parser status codes [page 829]
Data Cleanse Date, Phone, and SSN parser status codes [page 831]
Data Cleanse custom parser status codes [page 832]
Data Cleanse information codes [page 833]
Data Cleanse information code prefixes [page 833]
Data Cleanse information code descriptions [page 834]
Data Cleanse diacritical character conversions [page 839]

3.5.14.4 Data Cleanse Date, Phone, and SSN parser status codes

Date, Phone, and Social Security number parser status codes indicate how the transform standardized this information in your records.

Date, Phone, and Social Security number (SSN) parser status codes have the following status code format: `<parser_name>`#_`<output_field>`_STD. The transform replaces the pound sign (#) with the parser number.

Date, Phone and Social Security number parser status codes

Status code	Description
Date#_Date_Std	Output *Date* using the Standardized field class.
Phone#_Phone_Std	Output *Phone_Number* using the Standardized field class.
SSN#_SSN_Std	Output *SSN* using the Standardized field class.

The transform may assign multiple status codes per record. The transform separates multiple status codes with commas.

> **Example**
>
> The Date status code, `DATE1_DATE_STD`, `DATE6_DATE_STD`, shows that both *Date1* and *Date6* were standardized.
>
> `SSN1_SSN_STD`, `SSN5_SSN_STD` shows that both *SSN1* and *SSN5* were standardized.
>
> `PHONE1_PHONE_STD`, `PHONE3_PHONE_STD` shows that both *Phone1*Phone1 and *Phone3*Phone3 were standardized.

Parent topic: Data Cleanse reference [page 826]

Related Information

Data Cleanse status codes [page 827]
Data Cleanse person parser status codes [page 829]
Data Cleanse firm parser status codes [page 830]
Data Cleanse custom parser status codes [page 832]
Data Cleanse information codes [page 833]
Data Cleanse information code prefixes [page 833]
Data Cleanse information code descriptions [page 834]
Data Cleanse diacritical character conversions [page 839]

3.5.14.5 Data Cleanse custom parser status codes

Custom status codes indicate how the Data Cleanse transform parsed items that do not fall into the other categories.

Custom parser status codes have the following format: `<parser_name>`#_`<output_field>`_STD. The transform replaces the pound sign (#) with the parser number.

Custom parser status code description

Status code	Description
parser name#_output field_Std	Output the custom `<output_field>` using the Stadard field class.

The transform may assign multiple status codes per record separated by commas.

> **Example**
>
> The Automobile status code, `AUTOMOBILE1_COLOR_STD,AUTOMOBILE1_SIZE_STD,AUTOMOBILE2_COLOR_STD`, provides the following information:
>
> - Automobile1 used a *Color* and *Size*size standard.
> - Automobile2 used a *Color* standard.

Parent topic: Data Cleanse reference [page 826]

Related Information

Data Cleanse status codes [page 827]
Data Cleanse person parser status codes [page 829]
Data Cleanse firm parser status codes [page 830]
Data Cleanse Date, Phone, and SSN parser status codes [page 831]

Data Cleanse information codes [page 833]
Data Cleanse information code prefixes [page 833]
Data Cleanse information code descriptions [page 834]
Data Cleanse diacritical character conversions [page 839]

3.5.14.6 Data Cleanse information codes

The Data Cleanse transform issues information codes to inform you about data that may be suspect and require a manual review.

The transform outputs information codes to the *Info_Code* output field. The transform populates the *Info_Code* field with one or more codes separated by commas. Each code consists of a prefix that identifies the parser type, and the numeric code.

Parent topic: Data Cleanse reference [page 826]

Related Information

Data Cleanse status codes [page 827]
Data Cleanse person parser status codes [page 829]
Data Cleanse firm parser status codes [page 830]
Data Cleanse Date, Phone, and SSN parser status codes [page 831]
Data Cleanse custom parser status codes [page 832]
Data Cleanse information code prefixes [page 833]
Data Cleanse information code descriptions [page 834]
Data Cleanse diacritical character conversions [page 839]

3.5.14.7 Data Cleanse information code prefixes

Each information code consists of a prefix followed by the numeric code. The prefix identifies the parser type and the code indicates the reason why the record may be suspect and needs manual review.

The following table contains code prefixes that are two characters.

Two-character parser type prefixes

Parser type prefix	Description
D1## - D6##	*Date1*through *Date6* parse level information.
F1## - F6##	*Firm1* through *Firm6* parse level information.

Parser type prefix	Description
P1## - P6##	*Person1* through *Person6* parse level information.
T1## - T6##	*Phone1* through *Phone6* parse level information.

The following table contains code prefixes that are one character.

Single letter parser prefix

Parser type prefix	Description
I###	Input field level information.
R###	Record level information.

Parent topic: Data Cleanse reference [page 826]

Related Information

Data Cleanse status codes [page 827]
Data Cleanse person parser status codes [page 829]
Data Cleanse firm parser status codes [page 830]
Data Cleanse Date, Phone, and SSN parser status codes [page 831]
Data Cleanse custom parser status codes [page 832]
Data Cleanse information codes [page 833]
Data Cleanse information code descriptions [page 834]
Data Cleanse diacritical character conversions [page 839]

3.5.14.8 Data Cleanse information code descriptions

Each information code indicates a reason why a record is suspect and requires a manual review.

The following table shows more details about Data Cleanse information codes.

Information code	Description
Date parser information codes	
D#01	*Date#* was not in the expected format.
D#02	*Date#* was converted from 2-digits to 4-digits. The transform applied the century threshold option.
D#03	*Date#* was in the Day_Month_Year format.

Information code	Description
D#04	*Date#* was in the Month_Day_Year format.
D#05	*Date#* was in the Year_Month_Day format.
D#06	*Date#* was in the Year_Day_Month format.
D#07	*Date#* was in an ambiguous format. There is more than one possible format for the date. For example, 12/09/10 is valid for all of the formats, whereas 03/16/94 is valid only for the Month_Day_Year format.
Firm parser information codes	
F#01	The *Firm#* parse contained some data that was not found in the cleansing package.
F#02	The *Firm#* parse had a close person parse. This is only applicable for *Person_Firm* multiline parse when using the *Person_Firm* multiline parser or *Name_Firm_Line* when the data came from the input field *Name_Firm_Line*.
F#03	The *Firm#* parse was a presumptive firm parse based on reasonable evidence.
Person parser information codes	
P#01	The *Person#* parse contained some data that was not found in the cleansing package. This information code does not report on title information, which is different from information code P#51.
P#02	The *Person#* parse had a close *Firm#* parse. This is only applicable for *Person_Firm* multiline parse when using the *Person_Firm* multiline parser or *Name_Firm_Line* when the data came from the input field *Name_Firm_Line*.
P#03	The *Person#* parse was a presumptive name parse based on reasonable evidence.
P#04	The *Person#* parse has no given name, or has a questionable given name.
P#05	The *Person#* parse has no family name, or has a questionable family name.
P#51	The *Person#* parse contained a title token that was not found in the cleansing package. This is different from information code P#01.
P#52	The *Title#* parse has a close firm parse.
P#53	The *Title#* parse was presumptive.
Phone parser information codes	
T#01	The *Phone#* parse did not have a North American area code.
T#02	The transform parsed phone data using a different ISO2 country code than the country listed in the *Option_Country* field.

Information code	Description
T#03	The transform parsed phone data by prepending an ISO2 country code to the incoming phone data.
Input field parser information codes	
I111-I116	All of the input data in *Name_Line#* went to one or more *Extra* output fields; nothing was parsed for this input field.
I121-I126	There was no input data in Name_Line#.
I131-I136	Parsed some of the input data in *Name_Line#*, the rest of the data is in the *Extra* output field.
I151-I156	All of the input data in *Title_Line#* went to one or more *Extra* output fields. Nothing was parsed for this input field.
I161-I166	There was no input data in *Title_Line#*.
I171-I176	Parsed some of the input data in *Title_Line#*, the rest of the data is in the *Extra* output field.
I211-I212	All of the input data in *Person#* discrete went to one or more *Extra* output fields. Nothing was parsed for this input field.
I221-I222	There was no input data in *Person#* discrete.
I231-I232	Parsed some input data from *Person#* discrete. Remaining data is in the *Extra* output field.
I311-I316	All of the input data in *Name_or_Firm_Line#* went to one or more *Extra* output fields; nothing was parsed for this input field.
I321-I326	There was no input data in *Name_or_Firm_Line#*.
I331-I336	Parsed some of the input data in *Name_or_Firm_Line#*, the rest of the data is in the *Extra* output field.
I351-I352	All of the input data in *Firm_Line#* went to one or more *Extra* output fields; nothing was parsed for this input field.
I361-I366	There was no input data in *Firm_Line#*.
I371-I372	Parsed some of the input data in *Firm_Line#*, the rest of the data is in the *Extra* output field.
I421-I422	There was no input data in *Firm#* discrete.
I511-I516	All of the input data in *Date#* went to one or more *Extra* output fields; nothing was parsed for this input field.
I521-I526	There was no input data in *Date#*.
I531-I536	Parsed some of the input data in *Date#*, the rest of the data is in the *Extra* output field.
I711-I716	All of the input data in *Email#* went to one or more *Extra* output fields; nothing was parsed for this input field.
I721-I726	There was no input data in *Email#*.
I731-I736	Parsed some of the input data in *Email#*, the rest of the data is in the *Extra* output field.

Information code	Description
I751-I756	All of the input data in *Phone#* went to one or more *Extra* output fields; nothing was parsed for this input field.
I761-I766	There was no input data in *Phone#*.
I771-I776	Parsed some of the input data in *Phone#*, the rest of the data is in the *Extra* output field.
I811-I816	All of the input data in *SSN#* went to one or more *Extra* output fields; nothing was parsed for this input field.
I821-I826	There was no input data in *SSN#*.
I831-I836	Parsed some of the input data in *SSN#*, the rest of the data is in the *Extra* output field.
I851-I856	All of the input data in *UDPM#* went to one or more *Extra* output fields; nothing was parsed for this input field.
I861-I866	There was no input data in *UDPM#*.
I871-I876	Parsed some of the input data in *UDPM#*, the rest of the data is in the *Extra* output field.
I901-I912	All of the input data in *Date#* went to one or more *Extra* output fields; nothing was parsed for this input field.
I931-I942	There was no input data in *Multiline#*.
I951-I962	Parsed some of the input data in *Multiline#*, the rest of the data is in the *Extra* output field.
Record level parse information codes	
R001	All input field data went to one or more *Extra* output fields; nothing was parsed for the record.
R002	Parsed some of the input fields. One or more other input fields went to the *Extra* output field.
R003	Parsed part of the input fields. Some of the record data went to the *Extra* output field.
R004	All of the record input fields were parsed into appropriate output fields. None of the input data went to the *Extra* output field.
R400	The data in *Option_Content_Domain_Sequence* overrode the content domain sequence transform option.
R405	The data in *Option_Content_Domain_Sequence* was not recognized as a content domain sequence. The data in *Option_Country* or the data in the content domain sequence transform option was used to determine the content domain sequence.
R410	The data in *Option_Output_Format* overrode the output format data cleanse transform option.

Information code	Description
R415	The data in *Option_Output_Format* was not recognized as a valid output format. The data in *Option_Country* or the output format transform option was used to determine the output format.
R420	The data in *Option_Country* was recognized as an ISO2 country code and overrode the content domain sequence transform option.
R421	The data in *Option_Country* was recognized as an ISO2 country code and overrode the output format transform option.
R425	The data in *Option_Country* was not recognized as an ISO2 country code. One or more situations occurred where *Option_Country* data was used to override the content domain sequence, or output format transform options, or both, but failed: • Attempted to use *Option_Country* to override the content domain sequence transform option. This occurs when the *Option_Content_Domain_Sequence* data is invalid (R405) or the data is not supplied. No status code is generated when this occurs. The data was parsed using the content domain sequence transform option. • Attempted to use *Option_Country* to override the output transform option. This occurs when the *Option_Output_Format* data is invalid (R415) or is not supplied. No status code is generated when this occurs. The data was parsed using the output format transform option.
R428	The cleansing package does not recognize the ISO2 country code provided in the *Option_Country* input field.

Parent topic: Data Cleanse reference [page 826]

Related Information

Data Cleanse status codes [page 827]
Data Cleanse person parser status codes [page 829]
Data Cleanse firm parser status codes [page 830]
Data Cleanse Date, Phone, and SSN parser status codes [page 831]
Data Cleanse custom parser status codes [page 832]
Data Cleanse information codes [page 833]
Data Cleanse information code prefixes [page 833]
Data Cleanse diacritical character conversions [page 839]

3.5.14.9 Data Cleanse diacritical character conversions

In the Data Cleanse transform, there is an option where you can remove the diacritical characters from the data on output.

The following table shows the diacritical characters and the ASCII equivalent letter that replaces the diacritical character.

> **i Note**
>
> The diacritical character is listed first, separated by a comma and then the ASCII replacement letter.

Diacritical characters conversion chart

À,A	Ě,E	Ļ,L	Ś,S	Ź,Z	ē,e	ľ,l	ŝ,s
Á,A	Ė,E	Ľ,L	Ŝ,S	Ż,Z	ĕ,e	ŀ,l	ş,s
Â,A	Ę,E	Ŀ,L	Ş,S	à,a	ė,e	ŧ,l	ţ,t
Ã,A	Ĝ,G	Ł,L	Ţ,T	á,a	ę,e	ñ,n	ť,t
Ä,AE	Ğ,G	Ñ,N	Ť,T	â,a	ĝ,g	ń,n	ŧ,t
Å,A	Ġ,G	Ń,N	Ŧ,T	ã,a	ğ,g	ņ,n	ù,u
Ā,A	Ģ,G	Ņ,N	Ù,U	ä,ae	ġ,g	ň,n	ú,u
Ă,A	Ĥ,H	Ň,N	Ú,U	å,a	ģ,g	ò,o	û,u
Ą,A	Ħ,H	Ò,O	Û,U	ā,a	ĥ,h	ó,o	ü,ue
Ç,C	Ì,I	Ó,O	Ü,UE	ă,a	ħ,h	ô,o	ũ,u
Ć,C	Í,I	Ô,O	Ũ,U	ą,a	ì,i	õ,o	ū,u
Ĉ,C	Î,I	Õ,O	Ū,U	ç,c	í,i	ö,oe	ŭ,u
Ċ,C	Ï,I	Ö,OE	Ŭ,U	ć,c	î,i	ø,o	ů,u
Č,C	Ĩ,I	Ø,O	Ů,U	ĉ,c	ï,i	ō,o	ű,u
Ð,D	Ī,I	Ō,O	Ű,U	ċ,c	ĩ,i	ŏ,o	ŵ,w
Ď,D	Ĭ,I	Ŏ,O	Ų,U	č,c	ī,i	ő,o	ý,y
È,E	Į,I	Ő,O	Ŵ,W	ð,d	ĭ,i	ŕ,r	ÿ,y
É,E	İ,I	Ŕ,R	Ý,Y	è,e	į,i	ŗ,r	ŷ,y
Ê,E	Ķ,K	Ŗ,R	Ÿ,Y	é,e	ķ,k	ř,r	ž,z
Ë,E	ĸ,K	Ř,R	Ŷ,Y	ê,e	Í,I	š,s	ź,z
Ē,E	Ĺ,L	Š,S	Ž,Z	ë,e	ı,I	ś,s	ż,z

Example

This example shows the input data and how the diacritical characters are replaced on output.

Diacritical character removal example

Input	Output
María Hernández	Maria Hernandez
Geschäftsführer	Geschaeftsfuehrer
Telecomunicações São Paulo	Telecomunicacoes Sao Paulo

Parent topic: Data Cleanse reference [page 826]

Related Information

Data Cleanse status codes [page 827]
Data Cleanse person parser status codes [page 829]
Data Cleanse firm parser status codes [page 830]
Data Cleanse Date, Phone, and SSN parser status codes [page 831]
Data Cleanse custom parser status codes [page 832]
Data Cleanse information codes [page 833]
Data Cleanse information code prefixes [page 833]
Data Cleanse information code descriptions [page 834]
Data Cleanse Other Standardization options [page 398]

3.6 Platform transforms

Transforms that are needed for general data movement operations.

The platform transforms enable you to do the following:

- Generate, map, and merge rows from two or more sources.
- Create SQL query operations such as expressions, lookups, joins, and filters.
- Perform conditional splitting.
- Mask personal data to keep sensitive data relevant, anonymous, and secure.

Case transform [page 842]
: The Case transform enables you to process select rows in different ways by using branch logic instead of a conditional data flow.

Data Mask transform [page 847]
: Use the Data Mask transform to protect personally identifiable information in your data.

DQM Microservices [page 939]
: Use the DQM Microservices transform to configure and execute SAP Data Quality Management, microservices for location data services within SAP Data Services.

Map_Operation [page 952]

The Map_Operation transform enables you to modify input data using mapping expressions and operation codes.

Merge transform [page 959]
: Use the Merge transform to combine multiple input sources to produce a single output data set that has the same schema as the input data sets.

Query transform [page 960]
: The Query transform retrieves a data set that satisfies conditions that you specify. This transform is similar to a `SQL SELECT` statement.

Row_Generation transform [page 984]
: Use the Row_Generation transform to produce a data set that contains one column.

SQL transform [page 985]
: Use the SQL transform to perform the indicated SQL query operation on input data.

User Defined transform [page 989]
: Use the User Defined transform to define Python expressions to control how the transform processes data.

Validation transform [page 998]
: Use the Validation transform to qualify a data set based on rules for input schema columns.

XML_Map transform [page 1005]
: Use the XML_Map transform to produce a single target data set from one or more hierarchical source data sets.

Parent topic: Transforms [page 240]

Related Information

Transform reference [page 240]
Dynamic transform settings [page 244]
Embedded help for transform editors [page 249]
Data Integrator transforms [page 250]
Data Quality transforms [page 338]
Text Data Processing transforms [page 1025]

3.6.1 Case transform

The Case transform enables you to process select rows in different ways by using branch logic instead of a conditional data flow.

Case transform information

Characteristic	Description
	Case transform icon
Use	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. Use the Case transform to implement IF-THEN-ELSE logic instead of using a conditional flow. Keep in mind the following information when choosing between using a Case transform or a conditional: • Conditionals operate at the work flow level • Case transforms operate within data flows
Content objects	Sample transform.

Case transform editor [page 842]
When you include the Case transform in a data flow, use the options in the editor to create cases.

Case transform options [page 843]
Use the options in the *Case* tab to create cases, designate a default case, and use various function tools to build expressions.

Using the tools in the Case tab [page 845]
The Case tab in the Case transform editor provides several tools to help you create cases.

Case transform data inputs and outputs [page 846]
The Case transform takes input data and divides rows based on cases you create.

3.6.1.1 Case transform editor

When you include the Case transform in a data flow, use the options in the editor to create cases.

The Case transform editor consists of a Schema In pane in the upper portion, and the Case tab in the lower pane.

The Schema In pane contains the output fields from the upstream transform or the source fields.

The Case tab includes tools to define cases. A Case contains expressions that separate data from the source and routes the data to a specific target. The Case tab contains the following tools:

- Expression table that lists labels and their CASE expressions
- Embedded Smart Editor and a full-size Smart Editor for creating CASE expressions
- Function Wizard for creating functions

Parent topic: Case transform [page 842]

Related Information

Case transform options [page 843]
Using the tools in the Case tab [page 845]
Case transform data inputs and outputs [page 846]

3.6.1.2 Case transform options

Use the options in the *Case* tab to create cases, designate a default case, and use various function tools to build expressions.

The Case tab has three areas:

- Checkbox options: Upper portion of the *Case* tab that contains checkbox options and the *Add* and *Delete* buttons.
- Expression pane: Reflects added rows, built expressions, and expression labels.
- Function pane: Displays the selected expression, accepts dragged input fields to add to expressions, and contains the *Functions* and *Ellipses* button to access function building tools.

Case transform option descriptions

Option	Description
Checkbox options	
Produce default output with label	Specifies to create a default CASE using the name in the text box. Checked: The transform uses the default CASE when all other CASE expressions evaluate to False. Not checked: Option name changes to *Produce default output when all expressions are false*.
Produce default output when all expressions are false	Keep unchecked so that the transform does not use the default CASE when all other CASE expressions evaluate to False. If you check this option, it changes to *Produce default output with label*.

Option	Description
Text box	Specifies the name of the default case. Initially the value is *Default*. Enter a different name if applicable. The name appears in a list when you build your data flow and connect the Case transform to a target. For the default target object, you select the name of your default case.
Row can be TRUE for one case only	Specifies that the transform passes the row to the first case that evaluates to True. If not checked, the transform passes the row to all cases that evaluate to True. Select to enable *Preserve expression order*.
Preserve expression order	Specifies to process expressions in the order presented. Check this option if expression order is important to you. When this option is not checked, Data Services process the expressions in a less CPU intensive order, which enhances performance. ⚠ Caution When this option is not checked, the expression reordering can change results. There is no way to guarantee which expression evaluates to True first. Available only when you enable *Row can be TRUE for one case only*.
Add	Click to add a case row.
Delete	Click to delete a selected case row.
Expression table	
Label column	Specifies the name of the connection description that indicates where data goes when corresponding Case condition is true. When you draw a connection line from the Case transform to the target in the data flow, the transform presents a list of label names to select.
Function pane	
Functions	Click to open the Function Wizard.

Option	Description
Ellipses button	Click to open the full-size Smart Editor.

Parent topic: Case transform [page 842]

Related Information

Case transform editor [page 842]
Using the tools in the Case tab [page 845]
Case transform data inputs and outputs [page 846]

3.6.1.3 Using the tools in the Case tab

The Case tab in the Case transform editor provides several tools to help you create cases.

1. Select the *Produce default output when all expressions are false* checkbox to create a default case.
2. Enter a new name if applicable.

   Data Services sends rows to the default case when all other case conditions return false.
3. In the Case tab, click *Add*.

   The transform adds a new line to the expression table, and automatically populates the *Label* column.
4. Select a label and enter a new name for the label.

   The label name is important because you select it when you designate a target for the Case. For example, if the Case extracts records from the source that represent best records, rename the label to best_records.
5. Use various tools to create an expression.

Tool	Description
Functions button	Opens the Function Wizard to create expressions.
Ellipses button	Opens the full-features Smart Editor for larger expressions.
Drag and drop	Select a field from the Schema In pane and drag it to the function editor area.
Delete	Deletes selected expression

Task overview: Case transform [page 842]

Related Information

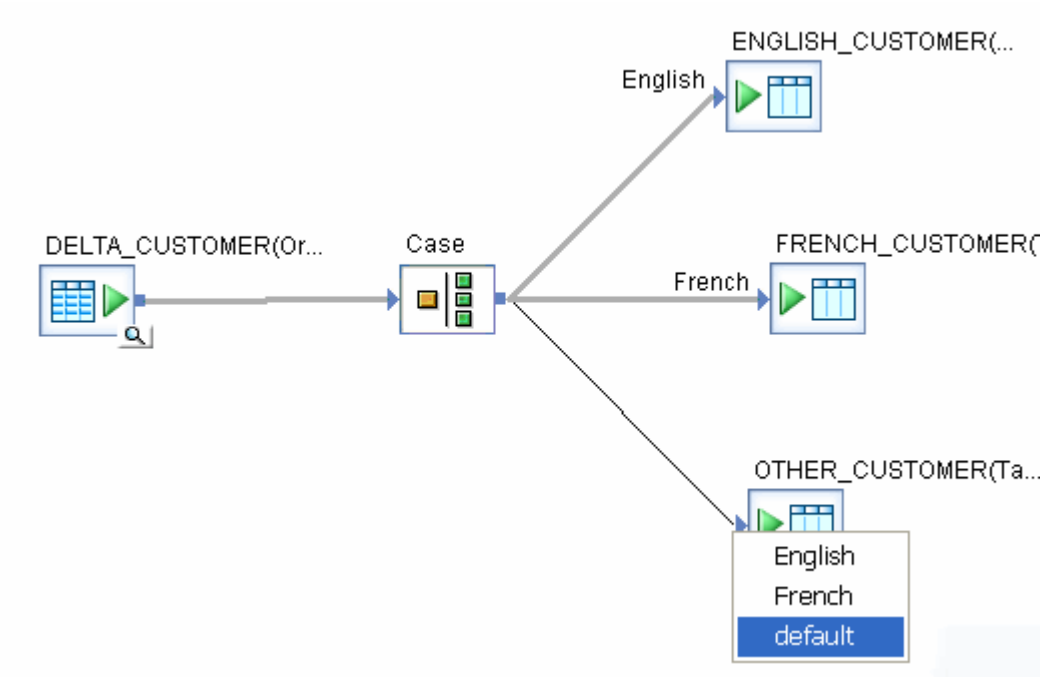
Case transform editor [page 842]
Case transform options [page 843]
Case transform data inputs and outputs [page 846]

3.6.1.4 Case transform data inputs and outputs

The Case transform takes input data and divides rows based on cases you create.

Data Services allows only one data source for the Case transform. However, the Case transform is designed to output to multiple targets.

In the following example, there are three target objects. When the user configured the Case transform, she created two cases and designated the labels English and French. The transform sends the rows that do not pass the expression for English or French to the default target.



When you draw the connection line from the Case transform to the target, Data Services asks you to select the applicable case. Data Services requires that you select each label. For example, if you have six cases in the Case transform, you add six targets to the data flow plus a target for the default case.

Parent topic: Case transform [page 842]

Related Information

Case transform editor [page 842]
Case transform options [page 843]
Using the tools in the Case tab [page 845]

3.6.2 Data Mask transform

Use the Data Mask transform to protect personally identifiable information in your data.

Data Mask transform information

Characteristic	Description
	Data Mask transform icon

Characteristic	Description
Use	Personally identifiable information (PII) includes data such as credit card numbers, salary information, birth dates, personal identification numbers, or bank account numbers. You may want to use data masking to support security and privacy policies, and to protect your customer or employee information from possible theft or exploitation. **Example** Mask employee salary information when you send your employee database to developers and testers in your company who do not have the proper security clearance to view such information, but who need realistic data to correctly develop and test the application. Place the Data Mask transform toward the end of your data flow to ensure that all fields that are to be masked have undergone processing by upstream transforms. If you place the Data Mask transform before other transforms in your data flow, downstream transforms may not be processing actual data but masked data, and in some cases, the transform won't be able to process the fields at all if the Data Mask transform replaced input data with blanks or a mask-out character such as "#". **Caution** The data masking changes are irreversible. Therefore, ensure that you preserve the original databases before masking data. SAP Data Services uses the following techniques for data masking: • Mask out • Number variance • Date variance • Pattern variance • Number generalization • Date generalization
Content objects	Configured transform

For descriptions of Filter Output Fields group, see Data Mask transform Filter Output Fields [page 852].

Data Mask transform input fields [page 849]
Data mask works with input fields that contain personal data.

Data Mask transform output fields [page 851]

The software lists the output fields that you see in the *Output* tab of the Data Mask transform based on the mapped input fields and the options that you set for each applicable Data Mask group.

Data Mask transform Filter Output Fields [page 852]
The *Filter Output Field* option determines what output fields appear in the Data Mask transform *Output* tab.

Data Mask transform Seed option [page 852]
To maintain referential integrity for multiple job runs, use the *Seed* option in the Data Mask transform *Processing Options* group .

Data Mask transform date parsing [page 853]
When input dates are ambiguous for Date Variance and Date Generalization operations, the Data Mask transform uses date parsing settings.

Data Mask transform Mask Out Group [page 856]
Use the Mask Out Group options to hide portions of fields for privacy and protection.

Data Mask transform Seed value [page 865]
The value that you enter for the *Seed* option in the *Processing Options* group ensures that SAP Data Services randomizes a field in the same way for each job run.

Data Mask transform Number Variance Group [page 865]
Set options in *Number Variance* under the *Number Variance Group* to set specific variance types and set minimum and maximum values for each applicable input field.

Data Mask transform Date Variance group [page 879]
Use the options in the *Date Variance Group* to output randomized dates based on a date range, or a fixed number of days, months, or years.

Data Mask transform Pattern Variance Group [page 894]
Use the option groups under the *Pattern Variance Group* to mask input data based on a specified pattern that you define.

Data Mask transform number generalization [page 913]
Use the *Number Generalization Group* to specify options that anonymize data based on ranges and replacement values, and default replacement values.

Data Mask transform Date Generalization [page 918]
To mask output data with a common label or date range, use the Data Mask Date Generalization feature.

3.6.2.1 Data Mask transform input fields

Data mask works with input fields that contain personal data.

Data Mask input fields

Input field	Description
Date[1-6]	For example, 08/16/2013 or August 16 2013. Available for mask out, date variance, and date generalization.

Input field	Description
Email[1-6]	E-mail address. Available for mask out and pattern variance.
Numeric_Data[1-6]	Numeric and character data. Supported numeric data types include decimal, double, integer, and real. Available for mask out, number variance, and number generalization.
Phone[1-6]	Phone number in any format such as a phone number that includes country code, area code, and formatting characters such as parentheses, dashes, and spaces. Available for mask out and pattern variance.
SSN[1-6]	U.S. Social Security number. Available for mask out and pattern variance.
General_Data[1-12]	Data that does not apply to any of the other Data Mask transform input fields. Available for mask out and pattern variance.

Select the *Best practice*, *In use*, or *All* for the *Filter* at the top of the *Input* tab to view the corresponding input fields.

Each of the masking groups in the Data Mask transform has an option named *Mapped Input Field*. To populate the drop list for this option, map your input fields to the appropriate transform input fields in the *Input* tab (*Transform Input Field Name* column). The following table lists the appropriate transform input fields for each data mask option group.

Option group	Map to transform input field name
Mask Out	• Date[1-6] • Email[1-6] • Numeric_Data[1-6] • Phone[1-6] • SSN[1-6] • GeneralData[1-12]
Number Variance	Numeric_Data[1-6]
Date Variance	Date[1-6]
Pattern Variance	• Email[1-6] • Phone[1-6] • SSN[1-6] • General_Data[1-12]
Number Generalization	Numeric_Data[1-6]
Date Generalization	Date[1-6]

> **i Note**
>
> To maintain formatting characters, such as commas, periods, slashes, and spaces, in the output field for the mask out process, set the input data type to Varchar. Additionally, set the *Maintain Formatting* option in the *Mask Out* group in the transform to *Yes*.

Related Information

Mask out maintain format characters [page 858]

3.6.2.2 Data Mask transform output fields

The software lists the output fields that you see in the *Output* tab of the Data Mask transform based on the mapped input fields and the options that you set for each applicable Data Mask group.

Open the *Output* tab and check the box next to all of the fields that you included in the Data Mask transform's *Mask Out*, *Number Variance*, *Date Variance*, *Pattern Variance*, *Number Generalization*, and *Date Generalization* groups.

> **i Note**
>
> If you do not check all of the boxes in the *Output* tab that are included in the *Data Mask* transform, the software issues a job execution warning when you execute the job.

If the fields that you plan to use for data mask are not listed in the *Output* tab, make sure that you have mapped them correctly in the *Input* tab, and that you have set up the corresponding *Mask Out*, *Number Variance*, *Date Variance*, *Pattern Variance*, *Number Generalization*, or *Date Generalization* groups in the Data Mask transform.

Select the *Best practice*, *In use*, or *All* for the *Filter* at the top of the *Output* tab to view the corresponding output fields.

You need to consider the output field size when you prepare the Data Mask transform for any type of masking. The output field length may not accommodate the length of the masked value. If the masked output value has more characters than the number of characters set for the output field, the software truncates the output data.

3.6.2.3 Data Mask transform Filter Output Fields

The *Filter Output Field* option determines what output fields appear in the Data Mask transform *Output* tab.

Option description

Option	Description
Filter Output Field	Specifies which output fields are displayed in the *Output* tab. *Show_Relevant_Fields:* The default setting. The fields available in the *Output* tab are the fields relevant to the mapped input. *Show_All_Fields:* All Data Mask transform output fields are available.

3.6.2.4 Data Mask transform Seed option

To maintain referential integrity for multiple job runs, use the *Seed* option in the Data Mask transform *Processing Options* group .

The following table describes the *Seed* option in the *Processing Options* group of the Data Mask transform.

Option	Description
Seed	(Optional) An alpha and/or a numeric string. Set this option once when you want to maintain referential integrity each time you run the job. One seed value maintains referential integrity for the following variance types set up in the *Data Mask* transform: *Number Variance*, *Date Variance*, and *Pattern Variance*. Additionally, to retain referential integrity for subsequent jobs using this job setup, you have to use the same data and you cannot make changes to the *Data Mask* transform, specifically to the applicable variance types.

Related Information

Data Mask transform Seed value [page 865]

3.6.2.5 Data Mask transform date parsing

When input dates are ambiguous for Date Variance and Date Generalization operations, the Data Mask transform uses date parsing settings.

The *Date Parsing Options* group is optional but it cannot be blank. You cannot repeat the *Date Parsing Options* group.

The settings that you make in the *Date Parsing Options* affect Date Variance and Date Generalization operations. The software uses the settings in the *Date Parsing Options* group only when an input date is ambiguous. An input date is ambiguous for various reasons. For example:

- When the input day of the month in numerals is 12 or less.
- When the input month name is the same in full or short form.
- When the input month is the same word in more than one language.
- When the input year is two digits and it is unclear what century to use.

Keep in mind the following facts when you complete the date parsing options:

- Keep the default settings or change the settings based on your input data.
- Ensure that your settings are valid for your input data. If your settings are not valid, the software may output NULL.
- Ensure that input dates are within the acceptable range and that they are valid. If the software does not recognize input dates because of these factors, it outputs NULL.

Data Mask transform Date Parsing options [page 854]
To specify how SAP Data Services parses input dates when the dates are ambiguous, use settings in the *Date Parsing Options* group.

Data Mask transform date parsing examples [page 855]
Use examples to help you understand how Date Parsing settings affect ambiguous input dates.

3.6.2.5.1 Data Mask transform Date Parsing options

To specify how SAP Data Services parses input dates when the dates are ambiguous, use settings in the *Date Parsing Options* group.

Date Parsing Options descriptions

Option	Description
Default Date Format	Specifies the order in which month, day, and year elements appear in the input string. The software uses this value only when the day, month, or year in the input string is ambiguous. Choose one of these formats: • *Day_Month_Year* • *Month_Day_Year* Default setting • *Year_Day_Month* • *Year_Month_Day*
Default Month Format	Specifies the month format when it is text and not numeric. The software uses this setting to output the month when it cannot determine the output month format based on the input alone. • *Full*: Output the spelled-out version of the month name. • *Short*: Output the abbreviated form of the month name, when an abbreviated form exists. Default setting.
Default Language	Specifies the language that the software should use when parsing ambiguous input dates that use text for the month. The software uses the set language for output. The default language is *English*. **i Note** The software does not verify that the user-defined *Default Language* corresponds to the language of the input month.
Default Century Threshold	Indicates whether a two-digit date is considered part of the 20th or 21st century. Enter a value from 0 to 99. The default value is 25. Specify a two-digit integer that represents the first year that a parsed two-digit year is considered part of the 21st century (20xx). All two-digit years greater than the specified integer are considered part of the 20th century (19xx).

3.6.2.5.2 Data Mask transform date parsing examples

Use examples to help you understand how Date Parsing settings affect ambiguous input dates.

Example

An input month of "Juni" may be ambiguous for language and for month format.

- The month "Juni" is ambiguous for language because it is the same spelling for both German and Norwegian.
- The month "Juni" is ambiguous for month format because it is both the full and short month format in German and Norwegian.

> **Note**
>
> "Juni" is used in other languages too, we only use the German and Norwegian version for example purposes.

Example

The software parses the following input date differently based on the *Default Date Format* setting.

- Input date: 10/01/2017

  Default Date Format = *DAY_MONTH_YEAR*
  Software parses the date as January 10, 2017
- Input date: 10/01/2017

  Default Date Format = *MONTH_DAY_YEAR*
  Software parses the date as October 1, 2017

Example

In English, the software randomizes an input date of May 5, 2014 to a randomized output date of March 22, 2013. However, because May is ambiguous in determining if the output is full or short, the software relies on the *Default Month Format* setting to determine the output format for month.

- When the *Default Month Format* = *FULL*, the software outputs "March" for the month.
- When the *Default Month Format* = *SHORT*, the software outputs "Mar" for the month.

Example

If you enter 11 for the *Default Century Threshold* option, the software processes all two-digit years of 11 or lower as part of the 21st century:

- 08 is considered 2008
- 11 is considered 2011
- 12 is considered 1912

Related Information

Data Mask transform Date Parsing options [page 854]

3.6.2.6 Data Mask transform Mask Out Group

Use the Mask Out Group options to hide portions of fields for privacy and protection.

Duplicate the Mask Out options as many times as needed.

Before you set mask out options, perform the following tasks:

- Select the field to mask
- Determine the portion or portions of the field to remain unmasked. Start from the beginning of the field to the end.
- Determine the number of characters to remain unmasked.
- Determine the replacement character for the masked portion of the field.

For the mask out process, the transform outputs all of the masked output fields as Varchar regardless of the input field type. Additionally, the transform converts any non Varchar input field to Varchar before it processes the field for masking.

> **i Note**
>
> To maintain formatting characters such as commas, periods, slashes, and spaces in the masked out output field:
>
> - Set the input data type to Varchar.
> - Set the *Maintain Formatting* option in the *Mask Out* group to *Yes*.

Data Mask transform Mask Out Group options [page 857]
Set *Mask Out Group* options to mask portions of output fields.

Mask out maintain format characters [page 858]
To control how the Data Mask transform outputs formatting characters, such as spaces, punctuation, and dashes, in the fields that you select for mask out, complete options in the *Mask Out* group.

Mask out examples [page 861]
Use examples to understand how settings in the *Mask Out Group* affect your data.

3.6.2.6.1 Data Mask transform Mask Out Group options

Set *Mask Out Group* options to mask portions of output fields.

Mask Out Group option descriptions

Option	Description
Mapped Input Field	Select the name of the mapped input field that you want to mask from the dropdown list. **i Note** The dropdown list includes only the fields that are mapped to one of the fields listed under the *Transform Input Field Name* column in the *Input* tab.
Starting Position	Indicates the portion of the field that remains unmasked. *Start*: Keeps the specified number of characters unmasked from the start of the string. *End*: Keeps the specified number of characters unmasked from the end of the string.
Unmasked Length	The number of characters to keep unmasked from the start or end of the field based on the *Starting Position* setting.
Masking Character	The character that replaces the original characters in the string. For example, “#” or “*”. **i Note** If you leave this option blank, the transform outputs only the portion of the field that remains unmasked. For example, with *Starting Position* set to END, *Unmasked Length* set to 4, and no setting for *Masking Character*, the number “123456789” is output as “6789”. However, when you set the *Masking Character* option to the “#” character, the output is “#####6789”.

Option	Description
Maintain Formatting	Indicates whether to keep formatting in the output field. The setting affects `<Email[1-6]>` output fields differently than other fields. **For fields other than `<Email[1-6]>`:** • *Yes*: Keeps any spaces, punctuation, and field formatting, such as dashes, slashes, or periods, that are present in the input field. For example, XXX-XXX-1234. • *No*: Treats spaces between characters, punctuation, and field formatting, such as dashes, slashes, or periods, like all other characters in the string. Outputs spaces using the masking character that you designate. **Example** For example, the dashes are masked with "X" (shown bolded for this example): XXX**X**XXX**X**1234. To maintain formatting characters such as commas, periods, slashes, and spaces in the masked out output field: • Set the input data type to *Varchar*. • Set the option, *Maintain Formatting*, to *YES*. **For `<Email[1-6]>` fields:** Specifies whether the transform maintains the formatting of the domain portion of an e-mail address. The domain portion of an e-mail address begins at the "@" character and includes formatting characters to the end of the address. • *Yes*: Maintains the formatting in the domain portion of an e-mail address (from the "@" character to the end of the e-mail address). • *No*: Masks all formatting characters in the domain portion of the e-mail address, including the name portion of the e-mail string. The name portion of the e-mail string is the portion before the @ character.

3.6.2.6.2 Mask out maintain format characters

To control how the Data Mask transform outputs formatting characters, such as spaces, punctuation, and dashes, in the fields that you select for mask out, complete options in the *Mask Out* group.

To keep the format characters from the input data in the masked output field, ensure that you make the following settings:

- Set the input data type to varchar.
- Set the *Maintain Formatting* option in the *Mask Out* group of the transform to *Yes*.
- Set all other *Mask Out* group options as applicable.

Note

Maintain formatting doesn't apply to e-mail fields.

Example

The examples in the following table show how SAP Data Services outputs the format characters from the input data based on the following *Mask Out* settings:

- *Starting Position* = End
- *Unmasked Length* = 4
- *Masking Character* = X
- *Maintain Formatting* = Yes

Data types and output results for Maintain Formatting

Input	Data type	Output	Comment
2014.12.24	Date	XXXX1224	Software doesn't output the periods, even though the *Maintain Formatting* option is set to Yes, because the input data type is Date and not Varchar.
2014.12.24	Varchar	XXXX.12.24	Software maintains the periods on output because the input data type is Varchar.
2014.12.24 21:30:00	Datetime	XXXXXXXXXX3000	Software doesn't output the periods, space, and colons, even though the *Maintain Formatting* option is set to Yes, because the input data type is Datetime and not Varchar.
2014.12.24 21:30:00	Varchar	XXXX.XX.XX XX:30:00	Software maintains the periods, space, and colons on output because the data type is Varchar.

Input	Data type	Output	Comment
1,234.56	Double	XX34.56	Software doesn't output the thousands separator, even though the *Maintain Formatting* option is set to Yes, because the data type is Double instead of Varchar. **i Note** This example also applies to other numeric data types.
1,234.56	Varchar	X,X34.56	Software maintains the thousands separator and the period on output because the data type is Varchar.
1.234,56	Double	XX34.56	Software doesn't maintain the thousands separator, even though the *Maintain Formatting* option is set to Yes, because the data type is Double. Also, the software changes the input comma to a period on output. **i Note** This example also applies to other numeric data types.
1.234,56	Varchar	X.X34,56	Software maintains the period and comma on output because the data type is Varchar.

Related Information

File format [page 71]

3.6.2.6.3 Mask out examples

Use examples to understand how settings in the *Mask Out Group* affect your data.

The following tables illustrate how the *Mask Out Group* transform options process various types of information such as U. S. Social Security Numbers, phone numbers, dates, and general data.

> **i Note**
>
> To maintain formatting characters (such as commas, periods, slashes, and spaces) in the masked out output field, set the *Data Type* for the applicable input field to varchar in the *File Format Editor*. Additionally, set the *Maintain Formatting* option in the *Mask Out* options to Yes.

For all of the examples below, the input field data type is set to varchar.

U. S. Social Security Numbers

Option settings	Original data	Output: Masking character = #	Output: Masking character = None
Starting Position = End *Unmasked Length* = 4 *Maintain Formatting* = (Yes or No)	123456789	#####6789	6789
Starting Position = End *Unmasked Length* = 4 *Maintain Formatting* = Yes **i Note** The transform maintains the dashes as formatting characters in the first output column because the *Maintain Formatting* option setting is Yes. When there is no masking character set as in the last output column, the transform does not maintain the formatting so the last column output does not include a dash.	123-45-6789	###-##-6789	6789
Starting Position = End *Unmasked Length* = 4 *Maintain Formatting* = No	123-45-6789	#######6789	6789
Starting Position = End *Unmasked Length* = 5 *Maintain Formatting* = Yes	123-45-6789	###-#5-6789	5-6789

Option settings	Original data	Output: Masking character = #	Output: Masking character = None
Starting Position = End *Unmasked Length* = 5 *Maintain Formatting* = No	123-45-6789	#######-6789	-6789
Starting Position = Start *Unmasked Length* = 3 *Maintain Formatting* = Yes	123-45-6789	123-##-####	123
Starting Position = Start *Unmasked Length* = 3 *Maintain Formatting* = No	123-45-6789	123########	123
Starting Position = Start *Unmasked Length* = 4 *Maintain Formatting* = Yes	123 45 6789	123 4# ####	123 4
Starting Position = Start *Unmasked Length* = 4 *Maintain Formatting* = No	123 45 6789	123 #######	123

Phone numbers

Option settings	Original data	Output: Masking character = #	Output: Masking character = None
Starting Position = End *Unmasked Length* = 4 *Maintain Formatting* = (Yes or No)	16085551212	#######1212	1212
Starting Position = End *Unmasked Length* = 4 *Maintain Formatting* = Yes	1 (608) 555-1212	# (###) ###-1212	1212
Starting Position = End *Unmasked Length* = 4 *Maintain Formatting* = No	1 (608) 555-1212	############1212	1212
Starting Position = End *Unmasked Length* = 5 *Maintain Formatting* = Yes	555-1212	##5-1212	5-1212
Starting Position = End *Unmasked Length* = 5 *Maintain Formatting* = No	555-1212	###-1212	-1212

Dates

Option settings	Original data	Output: Masking character = #	Output: Masking character = None
Starting Position = End *Unmasked Length* = 4 *Maintain Formatting* = (Yes or No)	04142014	####2014	2014
Starting Position = End *Unmasked Length* = 4 *Maintain Formatting* = Yes	04/14/2014	##/##/2014	2014
Starting Position = End *Unmasked Length* = 4 *Maintain Formatting* = No	04/14/2014	######2014	2014
Starting Position = End *Unmasked Length* = 4 *Maintain Formatting* = Yes	April 14, 2014	##### ##, 2014	2014
Starting Position = End *Unmasked Length* = 4 *Maintain Formatting* = No	April 14, 2014	##########2014	2014
Starting Position = Start *Unmasked Length* = 5 *Maintain Formatting* = Yes	July 4, 2014	July 4, ####	July 4
Starting Position = Start *Unmasked Length* = 5 *Maintain Formatting* = No	July 4, 2014	July #######	July

Email addresses

Unlike other personally identifiable transform input fields (Date, Numeric_Data, Phone, SSN, and General_Data), the software considers only portions of an email address to have “formatting” characters.

There are two parts to an email address. Each part is separated by the “@” symbol:

- The portion to the left of the @ symbol is the email name portion
- The portion to the right of the @ symbol is the domain name.

Email

Option settings	Original data	Output: Masking character = #	Output: Masking character = None
Starting Position = End *Unmasked Length* = 4 *Maintain Formatting* = Yes	some.one@sap.com	#######@##p.com	p.com

Option settings	Original data	Output: Masking character = #	Output: Masking character = None
Starting Position = End *Unmasked Length* = 4 *Maintain Formatting* = No	some.one@sap.com	############.com	.com
Starting Position = Start *Unmasked Length* = 3 *Maintain Formatting* = Yes	some.one@sap.com	som#####@###.###	som
Starting Position = Start *Unmasked Length* = 3 *Maintain Formatting* = No	some.one@sap.com	som##############	som

General data

Option settings	Original data	Output: Masking character = #	Output: Masking character = None
Starting Position = End *Unmasked Length* = 4 *Maintain Formatting* = Yes	High & low.	###h & low.	h & low.
Starting Position = End *Unmasked Length* = 4 *Maintain Formatting* = No	High & low.	#######low.	low.
Starting Position = Start *Unmasked Length* = 2 *Maintain Formatting* = Yes	High & low.	Hi## & ###.	Hi
Starting Position = Start *Unmasked Length* = 2 *Maintain Formatting* = No	High & low.	Hi#########	Hi
Starting Position = End *Unmasked Length* = 3 *Maintain Formatting* = Yes	Jon Q. Boyd-Low	### #. ####-Low	-Low
Starting Position = End *Unmasked Length* = 2 *Maintain Formatting* = No	Jon Q. Boyd-Low	#############ow	ow

Related Information

Mask out maintain format characters [page 858]
Data Mask transform input fields [page 849]
Data Mask transform Mask Out Group options [page 857]

3.6.2.7 Data Mask transform Seed value

The value that you enter for the *Seed* option in the *Processing Options* group ensures that SAP Data Services randomizes a field in the same way for each job run.

If you have set up the Data Mask transform to randomize a primary key field that you use to uniquely identify each record in the database, it is important that the software outputs the same randomized value for that primary key field for each time you process the database.

The best practice for setting a seed value is to use a name or number or a combination of the two that you can easily remember. For example, if you are using the `<East_Sales_Region>` database, and the data is from 2014 sales, you may want to set a seed value of **`eastsales2014`**.

After you run the job the first time, you may decide to make job file changes for the second and subsequent runs. To maintain referential integrity, do not change any of the following values:

- Seed value
- Number variance options
- Date variance options
- Pattern variance options
- Specified variance input field values

> **i Note**
>
> The transform allows one seed value. Therefore, if you set data masking for more than one field, the seed value affects any of the fields for which you include one or all of the following option groups:
>
> - *Number Variance Group*
> - *Date Variance Group*
> - *Pattern Variance Group*.

3.6.2.8 Data Mask transform Number Variance Group

Set options in *Number Variance* under the *Number Variance Group* to set specific variance types and set minimum and maximum values for each applicable input field.

Duplicate the *Number Variance* option group to apply the number variance settings to different output fields.

Set up number variance on any type of input field, such as varchar or integer. The transform processes any non-numeric fields as varchar by recognizing numeric data. When the transform recognizes numeric data, it varies the numeric data based on settings in the transform. The transform retains the field type from input to output for all of the fields used for number variance.

Use the *Seed Value* setting under *Processing Options* to maintain referential integrity. Referential integrity means that each time you run the job, the software maintains the same altered number from one run to the next.

Data Mask Number Variance [page 866]

Set options in *Number Variance* to alter numeric input values based on a variance type, such as fixed number, percentage, or range.

Related Information

Data Mask transform Seed value [page 865]

3.6.2.8.1 Data Mask Number Variance

Set options in *Number Variance* to alter numeric input values based on a variance type, such as fixed number, percentage, or range.

Number Variance options

Option	Description
Mapped Input Field	Select the name of the mapped input field that you want to output with number variance masking. **i Note** The drop-down list includes only the fields that you have mapped to one of the Numeric_Data[1-6] fields that are listed in the *Transform Input Field Name* column in the *Input* tab.

Option	Description
Type of Variance	**Percentage**: Changes the input number to a random number that is within a determined minimum and maximum range (see Note). The transform determines the range based on input value, percentage value, and the user-set minimum and maximum values. **Fixed Value**: Changes the input number to a random number that is within a determined minimum and maximum range (see Note). The transform determines the range based on input value, fixed number value, and the user-set minimum and maximum values. **i Note** For Percentage and Fixed Value, the software calculates internal minimum and maximum values. If the calculated minimum and maximum values do not violate (fall outside of) any user-set minimum and maximum values, the transform outputs a random number that is at or within the calculated minimum and maximum values. Sometimes the transform needs to use a combination of user-set and calculated minimum and maximum values. **Range**: Changes the input number to any number that is greater than or equal to the user-defined minimum value without going over the maximum value, regardless of the input value. **i Note** The *Variance* option is not applicable for Range, and the transform ignores any value in *Variance* when you choose Range for this option.
Variance	Specify the variance number based on the *Type of Variance*: Percentage or fixed (not applicable for range). Variance must be greater than zero and can have decimal places. **i Note** The *Variance* option is not applicable for the range variance type. The transform ignores any value in *Variance* when you select Range.

Option	Description
Minimum Value	Required for range variance type and optional for percentage and fixed variance types. Specify the lowest value that the transform can output for the mapped input field. This number can be a decimal. For best results, always set a realistic minimum value. You may see some unrealistic numeric variations or null values if you do not set a realistic minimum value. **i Note** The transform calculates an internal minimum value for each record. If the calculated minimum doesn't violate (is less than) the user-defined minimum set for this option, the transform bases the random output on the calculated minimum.
Maximum Value	Required for range variance type and optional for percentage and fixed variance types. Specify the highest value that the transform can output for the mapped input field. This number can be a decimal. For best results, always set a realistic maximum value. You may see some unrealistic numeric variations or null values if you do not set a realistic maximum value. **i Note** The transform calculates an internal maximum value for each record. If the calculated maximum doesn't violate (is greater than) the user-defined maximum set for this option, the transform bases the random output on the calculated maximum.

i Note

Best practice is to always complete the *Minimum Value* and the *Maximum Value* options regardless of the variance type, and use realistic values to avoid unrealistic numeric values or null output.

3.6.2.8.2 Number variance calculated min and max values

The Data Mask transform uses minimum and maximum values that SAP Data Services calculates internally to determine a random output value.

Data Services does not ues the calculated minimum and maximum values when the calculated values violate the user-set minimum and—or maximum values set in the transform.

The transform uses the calculated minimum and—or maximum values to output a random number that is equal to or within the calculated values. Sometimes the determined minimum and maximum values are a combination of the user-set and the calculated minimum and maximum values.

> **Example**
>
> If the calculated minimum value falls outside of the user-set minimum (calc. min. is less than set min.), the transform uses the user-set minimum value. If the calculated maximum falls outside of the user-set maximum (calc. max. is greater than set max.), the transform uses the user-set maximum.

Calculated minimum and maximum for fixed variance type

The transform uses the following formula for calculated minimum and maximum values for fixed number variance:

input – variance = calc. min.

input + variance = calc. max.

For example, given an input value of 500, and a variance of 25, the calculated minimum and maximum formulas are:

Calculated minimum: 500 – 25 = 475

Calculated maximum: 500 + 25 = 525

Calculated minimum and maximum for percentage variance type

The transform uses the following formula for calculated minimum and maximum values for percentage number variance:

Input – (input x variance %) = calc. min.

Input + (input x variance %) = calc. max.

For example, given an input value of 500 and a variance percent of 25, the calculated minimum and maximum formulas are:

Calculated minimum: 500 – (500 x 25%) = 375

Calculated maximum: 500 + (500 x 25%) = 625

> **Note**
>
> (500 x 25% = 125)

3.6.2.8.3 Number variance min and max examples

Use examples to understand how the minimum and maximum values and the software calculated numbers affect your job setup.

Example

The following examples illustrate how the software calculates number variance, especially when one or both of the user set *Minimum Value* or *Maximum Value* fields are blank or fall outside of the calculated minimum and maximum values.

> **i Note**
>
> Set minimum and maximum values that result in realistic variances. For example, in the following table there are some cases when the output is null, which may not be realistic for your purposes. The best practice is to always set a *Minimum Value* and *Maximum Value* in the transform options.

Input = 10
Type of Variance = Fixed Number
Variance = 5
Minimum Value = (as set in the User-Defined minimum column in the following table)
Maximum Value = (as set in the User-Defined maximum column in the following table)

Calculated minimum value: 5

Calculated maximum value: 15

Example	User minimum	User maximum	Minimum used	Maximum used	Notes
1	Not set	Not set	5 (calculated)	15 (calculated)	Output is a random number between the calculated range of 5-15. **i Note** There are no user minimum and maximum settings so the software uses the calculated minimum and maximum.

Example	User minimum	User maximum	Minimum used	Maximum used	Notes
2	2 (under calculated minimum)	20 (over calculated maximum)	5 (calculated)	15 (calculated)	Output is a random number between the calculated range of 5-15. **i Note** The user-set minimum value falls outside of the calculated minimum. The user-set maximum also falls outside of the calculated maximum.
3	Not set	4 (under calculated **minimum**)	--	--	Output is null. **i Note** There is no user-set minimum value, and the user-set maximum value falls outside of the 5-15 calculated range so there is no range for randomization.
4	Not set	5 (equal to calculated **minimum**)	--	--	Output is null. **i Note** There is no user-set minimum value and the user-set maximum value is equal to the calculated minimum. Therefore there is no range for randomization.

Example	User minimum	User maximum	Minimum used	Maximum used	Notes
5	Not set	20 (over calculated maximum)	5 (calculated)	15 (calculated)	Output is a random number between the calculated range of 5-15. **i Note** There is no user-set minimum value, and the user-set maximum value falls outside of the calculated range.
6	Not set	14 (under calculated maximum)	5 (calculated)	14 (user)	Output is a random number between 5-14. **i Note** There is no user-set minimum value. However, the user-set maximum value falls within the calculated maximum. The software does not violate the user-set maximum value because it is within the calculated range.

Example	User minimum	User maximum	Minimum used	Maximum used	Notes
7	15 (equal to calculated **maximum**)	Not set	--	--	Output is null. **Note** The user-set **minimum** value equals the calculated **maximum** value, and there is no user-set maximum value. Therefore, there is no range for randomization.
8	20 (over calculated range)	Not set	--	--	Output is null. **Note** The user-set minimum value falls outside of the calculated maximum value (and the calculated range), and there is no user-set maximum value. Therefore there is no range for randomization.

3.6.2.8.4 Number variance type: Percentage

Use the following examples to help you better understand how the percentage variance type works.

The transform outputs a random number that is within a determined range. It determines the range by calculating minimum and maximum values based on the input value, percentage value, and the minimum and maximum values that you set in the job.

Because the transform considers both calculated and user-set minimum and maximum values when determining an output range, complete the *Minimum Value* and *Maximum Value* options when you set up your job (even though they are optional) with realistic values. You may see some unrealistic numeric variations or null values if you do not set realistic minimum and maximum values.

Example

Number Variance option settings:

Mapped Input Field = Income
Type of Variance = Percentage
Variance = 25

> **i Note**
>
> The Variance must be greater than zero and it can be a decimal.

Minimum Value = 100,000
Maximum Value = 500,000

Calculated minimum = input - (input x 25%)

Calculated maximum = input + (input x 25%)

Record	Input	Calculated min/max	Output
1	250,000	187,500 and 312,500	The output is a random number from 187,500 to 312,500. **i Note** The transform uses the calc. min. because it is greater than the user-set min. of 100,000. The transform uses the calc. max. because it is less than the user-set max. of 500,000.
2	95,000	71,250 and 118,750	The output is a random number between 100,000 and 118,750. **i Note** The transform uses a mixture of the user-set minimum and the calculated maximum. The calc. min. of 71,250, violates (is less than) the user-set min. of 100,000. The calc. max. of 118,750 does not violate (is less than) the user-set max. of 500,000.

3.6.2.8.5 Number variance type: Fixed number

Use the following examples to help you better understand how the fixed number variance type works.

The transform outputs a random number that is within a determined range. It determines the minimum and maximum range values to use based on the input value, fixed number value, and the minimum and maximum range that you set in the job.

Because the transform considers both calculated and user-set minimum and maximum values when determining an output range, it is important that you complete the *Minimum Value* and *Maximum Value* options when you set up your job (even though they are optional) with realistic values. You may see some unrealistic numeric variations or null values if you do not set realistic minimum and maximum values.

Example

Number Variance option settings:

Type of Variance = Fixed Number
Mapped Input Field = monthly_income
Variance = 2500

> **i Note**
>
> You must set the variance to greater than zero, and you can use decimal values.

Minimum Value = 1,000
Maximum Value = 5,000

For this example, the mapped input field contains a phrase consisting of words, spaces, and multiple number groupings. When your mapped input field contains alphanumeric data like this, the transform applies the fixed number variance to only the first group of numbers in the field, and does not change any subsequent groups of numbers in the field. Therefore, the second number group, "2014", remains unchanged.

Record	Input	Calculated min/max	Output
1	Income/month $1200 for 2014	-1,300 and 3,700	The output string is a random number from 1000 to 3,700: **i Note** The transform uses the user-set min. of 1000 because the calc. min. is greater than the set min. of -1,300. The transform uses the calc. max. because it is less than the set max of 5,000.

Record	Input	Calculated min/max	Output
2	Income/month $2500 for 2014	0 and 5,000	The output string is a random number from 1,000 to 5,000. **i Note** The transform uses the user-set min because the calc. min. of 0 is less than the set. min. The cal. max. and user-set max are equal so the value 5,000 is used for the max.

Example

If the input phrase lists "2014" as the first number group, the transform randomizes "2014" in all records, leaving the monthly income unmasked. Make sure you consider your input when setting up number variance.

For the following example, we used the same Number Variance option settings as in the first example:

Input	Calculated min/max	Output
"Income/month for 2014 is $1200"	-486 and 4,514 (calculated using 2014 as the input number)	"Income/month for 3527 is $1200" **i Note** The transform uses the set min. of 1,000 because the calc.min of -486 is less than the set min. The transform uses the calc. max. of 4,514 because it is less than the set max. of 5,000.

3.6.2.8.6 Number variance type: Range

Use the following examples to help you better understand how the range variance type works.

The transform randomly changes the input number to any number that is greater than or equal to the *Minimum Value* without going over the *Maximum Value*, regardless of the input value. The transform does not use a calculated minimum or maximum value for range variance type. The transform ignores any value in the *Variance* option.

Example

Number Variance option settings:

Mapped Input Field = Income
Type of Variance = Range
Variance = (not set)
Minimum Value = 100,000
Maximum Value = 500,000

Record	Input	Output
1	45,000	A random number between 100,000 and 500,00
2	525,000	A random number between 100,000 and 500,00

3.6.2.8.7 Number variance number formats

Use the following examples to help you understand how the transform processes numbers and number formatting.

The Data Mask transform outputs several types of number indicators when they are present in the input field:

- Thousand separators (comma, period, and space)
- Decimal separator (comma and period)
- Sign indicator (+/-)
- Number of significant figures

However, even when the field contains numbers and number formatting, the transform may not always recognize the entire input string as a number. The following table contains some examples of how the transform may vary numbers based on the format of the input number.

Examples: Number formats

Input data	Portion that transform recognizes as number (internal process)	Example of random output with number indicators from input data
1000000	1000000	2175924
1,000,000	1000000	2,175,924
1.000.000	1000000	2.175.924
1 000 000	1000000	2 175 924
10000	10000	25831
10,000	10000	25,831
10 000	10000	25 831
10.000	10.000	9.024

Input data	Portion that transform recognizes as number (internal process)	Example of random output with number indicators from input data
10	10	16 **i Note** Notice the difference between the output for the previous row (9.024) and the output for this row (16). Even though the input numbers are both ten, the transform assigned the output in the previous row as a decimal because there is a decimal in the input value.
-15,2	-15.2	-1,9
+775	775	+801
-,9823	-0.9823	1,4761
+.23400	0.23400 **i Note** The transform adds a zero before the decimal before calculating the number variance output.	+1.23456
10,01,560	10.01	65,29,560 **i Note** The transform recognizes the first number in the string (10,01) as a decimal. The transform does not modify the ",560" portion of the string because it doesn't match the pattern of a decimal.
9 10 11	9	26 10 11 **i Note** The transform recognizes the first number in the string (9). The transform does not modify the "10" and "11" portions of the string.

Input data	Portion that transform recognizes as number (internal process)	Example of random output with number indicators from input data
100,250 25	100250	201,873 25 **i Note** The transform recognizes the first number in the string (201,873). However, because there is a space after the first number, the transform does not modify the "25" portion of the string.
-.123	-0.123	-0.531 **i Note** The transform adds a zero ("0") before the decimal before it calculates the number variance output.

3.6.2.9 Data Mask transform Date Variance group

Use the options in the *Date Variance Group* to output randomized dates based on a date range, or a fixed number of days, months, or years.

Duplicate the *Dave Variance* options group to apply the date variance settings to different input fields.

Date Variance for data masking works with input fields that have been mapped to one of the `<Date[1-6]>` fields that are listed in the *Transform Input Field Name* column in the *Input* tab.

The transform parses and then randomizes the date data based on the date variance settings in the transform. The transform retains the field type from input to output for all of the fields used for date variance.

Use the *Seed Value* setting under *Processing Options* to maintain referential integrity. Referential integrity means that each time you run the job, the software maintains the same altered number from one run to the next.

Date Variance options [page 880]
Set options in *Date Variance* in the *Date Variance Group* to alter input dates based on a date variance type or by a date range.

Date variance input fields [page 883]
Date variance for data masking works with date or varchar data types that you map to a `<Date[1-6]>` field listed under the *Transform Input Field Name* column in the *Input* tab.

Date variance calculated min and max dates [page 884]
SAP Data Services uses date variance settings in your job to calculate minimum and maximum dates to help determine a random output date.

Date variance examples: Calculated values for fixed options [page 885]
Use examples to understand how the software calculates minimum and maximum values for fixed days, months, and years.

Date variance examples [page 886]
Use the following examples to help you understand how date variance works.

Related Information

Data Mask transform Seed value [page 865]

3.6.2.9.1 Date Variance options

Set options in *Date Variance* in the *Date Variance Group* to alter input dates based on a date variance type or by a date range.

Date Variance options

Option	Description
Mapped Input Field	Name of the input field that you want to output with date variance masking. **i Note** The drop-down list includes only the fields that you have mapped to one of the Date[1-6] fields that are listed in the *Transform Input Field Name* column in the *Input* tab.

Option	Description
Type of Variance	Specifies how you want to vary a date. • *Fixed Days*: Specifies to randomize the input date to somewhere within plus or minus a fixed number of days. • *Fixed Months*: Specifies to randomize a date to somewhere within plus or minus a fixed number of months. • *Fixed Years*: Specifies to randomize a date to somewhere within plus or minus a fixed number of years. **i Note** For Fixed Days, Fixed Months, and Fixed Years, the software calculates internal minimum and maximum dates. If the calculated minimum and maximum dates do not violate (fall outside of) any user-set minimum and maximum dates, the transform outputs a random date that is at or within the calculated minimum and maximum dates. Sometimes the transform needs to use a combination of user-set and calculated minimum and maximum dates. • *Range*: Specifies to randomize the input date to a date that is at or within a set minimum and maximum date range. You must complete the *Minimum Date* and *Maximum Date* options when you choose the Range variance type. **i Note** *Variance* option is not applicable for Range, and the transform ignores any value in *Variance* when you choose Range for this option.
Variance	Required only for the fixed variance types (days, months, years). Determines the number (days, months, years) by which to randomize the input. The value must be greater than zero (0).The **i Note** If you choose the range variance type, the software ignores any value in this option.

Option	Description
Minimum Date	Required for range variance and optional (but recommended) for fixed date variance types. Specify the minimum date allowed on output. Either type in a date in the **`YYYY.MM.DD`** format, or click the drop-down arrow in the field and click *<Select Date...>* to use the *Set Date* dialog box to set a date. The minimum date that can be set is September 14, 1752 (1752.09.14). **i Note** The transform calculates an internal minimum date for each record. If the calculated minimum date doesn't violate (is less than or earlier than) the user-defined minimum set for this option, the transform bases the random output on the calculated minimum.
Maximum Date	Required for range variance type and optional (but recommended) for fixed variance types. Specify the maximum date allowed on output. Either type in a date in the **`YYYY.MM.DD`** format, or click the drop-down arrow in the field and click *<Select Date...>* to use the *Set Date* dialog box to set a date. The maximum date that can be set is December 31, 9999 (9999.12.31). **i Note** The transform calculates an internal maximum date for each record. If the calculated maximum date doesn't violate (is greater than or later than) the user-defined maximum setting for this option, the transform bases the random output on the calculated maximum.

3.6.2.9.2 Date variance input fields

Date variance for data masking works with date or varchar data types that you map to a `<Date[1-6]>` field listed under the *Transform Input Field Name* column in the *Input* tab.

Date variance input field

Input data type	Date variance processing
Date	The software varies the date based on your *date variance* settings in the job. The software has minimum and maximum date limits for date data types (these dates are different from the limits for character data types): • Minimum = September 14, 1752 • Maximum = December 31, 9999 If the input or randomized date falls outside of this range, or it is an invalid date for other reasons, then the software outputs null for the record.
Datetime	The software varies the date portion of the input field based on your *Date Variance* settings in the job. The software outputs the time portion of the input field unaltered. The minimum and maximum date limits for datetime are the same as for Date data types. If the input or randomized date falls outside of this range, or it is an invalid date for other reasons, then the software outputs null for the record.

Input data type	Date variance processing
Character (like Varchar)	The software parses and randomizes the date portion of a character string and outputs the randomized date. The output format is the same as it is on input. If there are two dates in a field, or there are other characters in the field, the software parses and randomizes the first date and outputs the other characters in the field (including the second date) unchanged. For example, if the input character string is `2013.05.01 – 2013.05.05`, the software parses and randomizes the first date, and outputs the rest of the string unchanged. The output could be a character string of `2013.05.07 – 2013.05.05` (the software varies the first date based on the *Date Variance* settings in the job). The software has minimum and maximum date limits for character data type (these dates are different from the limits for date data types): • Minimum is January 1, 1900 • Maximum is December 31, 2099 If the software does not recognize the date, then it outputs null for the record. The software may not recognize the date because it is outside of the acceptable range, or it is an invalid date for other reasons. ()

Related Information

Data Types [page 1044]
Data Mask transform input fields [page 849]
Data Mask transform Date Parsing options [page 854]

3.6.2.9.3 Date variance calculated min and max dates

SAP Data Services uses date variance settings in your job to calculate minimum and maximum dates to help determine a random output date.

Data Services also considers the user-set minimum and maximum settings from the job. Data Services uses the calculated minimum and—or maximum dates only when the user-set minimum and maximum dates fall outside of (violate) the calculated range.

Sometimes the minimum and maximum dates that the software uses are a combination of the user-set and the calculated minimum or maximum dates.

3.6.2.9.4 Date variance examples: Calculated values for fixed options

Use examples to understand how the software calculates minimum and maximum values for fixed days, months, and years.

Example: Fixed days date variance

Given an input value of `2014.01.29` and a variance of 10 days, the calculated minimum and maximum dates are:

- Calculated minimum: 2014.01.29 - 10 days = 2014.01.19
- Calculated maximum: 2014.01.29 + 10 days = 2014.02.08

Example: Fixed months date variance

Given an input value of `2014.01.29` and a variance of 10 months, the calculated minimum and maximum dates are:

- Calculated minimum: 2014.01.29 - 10 months = 2013.03.29
- Calculated maximum: 2014.01.29 + 10 months = 2014.11.29

Example: Fixed years date variance

Given an input value of `2014.01.29` and a variance of 10 years, the calculated minimum and maximum dates are:

- Calculated minimum: 2014.01.29 - 10 years = 2004.01.29
- Calculated maximum: 2014.01.29 + 10 years = 2024.01.29

3.6.2.9.5 Date variance examples

Use the following examples to help you understand how date variance works.

Example

The following table features some examples for one record in a database. The examples help to illustrate how the software calculates the date variance when one or both of the user-set *Minimum Date* or *Maximum Date* fields are blank or set to unrealistic date ranges.

> **i Note**
>
> It is important to set minimum and maximum values that result in realistic variances. For example, in the following table there are some cases when the output is null, which may not be realistic for your purposes. The best practice is to always set a realistic *Minimum Date* and *Maximum Date* in the transform options.

> **i Note**
>
> The input date format is YYYY.MM.DD.

- *Input Date* = 2014.09.17
- *Variance Type* = Fixed Days
- *Variance* = 7
- *Minimum Date* = (as set in the User minimum column in the following table)
- *Maximum Date* = (as set in the User maximum column in the following table)
- Calculated minimum date: 2014.09.10
- Calculated maximum date: 2014.09.24

Example	User minimum	User maximum	Minimum used	Maximum used	Notes
1	Not set	Not set	2014.09.10 (calculated)	2014.09.24 (calculated)	Output is a random date somewhere between the calculated minimum and maximum dates. **i Note** Because there are no user-set minimum and maximum dates, the transform uses the calculated minimum and calculated maximum dates to keep the randomized date within the transform settings.
2	Not set	2014.09.09 (under the calculated **minimum**)	--	--	Output is null. **i Note** There is no user-set minimum date, and the user-set maximum date falls outside of the calculated date range of 2014.09.10 and 2014.09.24. There is no range for randomization.

Example	User minimum	User maximum	Minimum used	Maximum used	Notes
3	Not set	2014.09.10 (equal to the calculated **minimum** range)	--	--	Output is null. **i Note** There is no user-set minimum date, and the user-set **maximum** date is equal to the calculated **minimum** date of 2014.09.10. There is no range for randomization.
4	Not set	2014.09.23 (within the calculated range)	2014.09.10 (calculated)	2014.09.23 (user)	Output is a random date somewhere between the calculated and user-set dates. **i Note** The user-set minimum is not set. The user-set **maximum** date is within the calculated range, and it is **less than (earlier than)** the calculated maximum date of 2014.09.24. The software won't violate the user-set maximum date.

Example	User minimum	User maximum	Minimum used	Maximum used	Notes
5	Not set	2014.09.24 (equal to calculated maximum)	2014.09.10 (calculated)	2014.09.24 (user)	Output is a random date somewhere between the calculated and user-set dates. **i Note** The software uses the calculated minimum date because the user did not set a minimum date. It uses the user-set maximum date because it is equal to the calculated maximum date, and therefore does not violate it.

Example	User minimum	User maximum	Minimum used	Maximum used	Notes
6	Not set	2014.09.30 (over calculated maximum)	2014.09.10 (calculated)	2014.09.24 (calculated)	Output is a random date somewhere between the calculated minimum and calculated maximum dates. **i Note** There is no user-set minimum date so the software uses the calculated minimum date. The user-set maximum date is **greater than (later than)** the calculated maximum date. The software won't violate (go later than) the calculated maximum, so it uses the calculated maximum date.

Example	User minimum	User maximum	Minimum used	Maximum used	Notes
7	2014.07.01 (under calculated minimum range)	Not set	2014.09.10 (calculated)	2014.09.24 (calculated)	Output is a random date somewhere between the calculated minimum of 2014.09.10 and the calculated maximum of 2014.09.24. **i Note** The user-set minimum date is **less than (earlier than)** the calculated minimum date, and there is no user-set maximum date. Therefore the software uses the calculated minimum date because the user-set minimum violates the calculated date range, and it uses the calculated maximum because the user did not set a maximum date.

Example	User minimum	User maximum	Minimum used	Maximum used	Notes
8	2014.07.01	2014.07.31	2014.07.01 (user)	2014.07.31 (user)	A random number somewhere between the user-set minimum and maximum dates. **i Note** The user-set date range is totally outside of the calculated range of 2014.09.10 and 2014.09.24. Therefore, the software uses the user-set date range for randomization.
9	2014.09.24 (equal to calculated maximum range)	Not set	--	--	Output is null. **i Note** The user-set **minimum** date is equal to the calculated **maximum** date, and there is no user-set maximum date. There is no range for randomization.

Example	User minimum	User maximum	Minimum used	Maximum used	Notes
10	2014.09.30 (greater than calculated maximum)	Not set	--	--	Output is null. **i Note** The user-set **minimum** date is greater than (later than) the calculated **maximum** date, and there is no user-set maximum date. There is no range for randomization.

Example	User minimum	User maximum	Minimum used	Maximum used	Notes
11	2014.09.11 (later than calculated minimum)	2014.09.30 (later than calculated maximum)	2014.09.11 (user)	2014.09.24 (calculated)	The output is a random number somewhere between the user-set minimum date and the calculated maximum date. **i Note** The user-set minimum date **is greater than (later than)** the calculated minimum date of 2014.09.10, but still **less than (earlier than)** the calculated maximum of 2014.09.24, so the user-set minimum is still within the calculated range. The user-set maximum date is **greater than (later than)** the calculated maximum date. Therefore the user maximum violates the calculated date range.

3.6.2.10 Data Mask transform Pattern Variance Group

Use the option groups under the *Pattern Variance Group* to mask input data based on a specified pattern that you define.

The transform parses and then randomizes an input field based on your settings in the *Pattern Variance Group*. Set options in the *Pattern Variance Group* to mask input data based on the following pattern variance types:

- Default

- Preserve
- Character
- String

Duplicate the *Pattern Variance* group of options under the *Pattern Variance Group* for each mapped input field that you want to mask.

The *Pattern Variance* group contains a secondary group named *Definition*. Set options in the *Definition* group to define a substring of the mapped input field. Complete at least one *Definition* group per *Pattern Variance* group. Duplicate the *Definition* group to define additional substrings of the mapped input field.

> **i Note**
>
> When you include multiple *Definition* groups for a mapped input field, each group must specify a unique substring of the input field. The software issues a verification error if you have defined overlapping substrings for the specified input field.

Use the *Seed Value* setting under *Processing Options* to maintain referential integrity. Referential integrity means that each time you run the job, the software maintains the same altered number from one run to the next.

Pattern variance for data masking works with varchar input fields that are mapped to the following transform input fields: `<Email[1-6]>`, `<Phone[1-6]>`, `<SSN[1-6]>`, and `<General_Data[1-12]>`.

Data Mask Pattern Variance [page 896]
Use the *Pattern Variance Group* to mask an input field substring with a specific pattern variance type.

Pattern Variance input fields [page 898]
During job set up, map varchar input fields to specific transform fields so they are available to select for pattern variance.

Pattern Variance type: Default [page 899]
The default variance type replaces input characters with like characters and retains spaces and special characters.

Pattern Variance type: Preserve [page 900]
For the variance type Preserve, the software outputs the defined substring as it was input, with no masking.

Pattern Variance type: Character [page 901]
The character variance type masks each character in the defined substring with a character from the *Value* field.

Pattern Variance type: String [page 902]
For the variance type string, the software masks the entire defined substring with a random character or string from the *Value* field.

Pattern Variance: Controlling replacement value [page 903]
Control the randomness of character and string replacement by increasing the odds that SAP Data Services uses one replacement value over another.

Pattern Variance: Character vs. string processes [page 904]
The software processes character and string pattern variances differently when the input *Value* field or the input substring contains specific character types.

Pattern Variance: Use pipe delimiters for multiple values [page 906]

When you list multiple characters or strings in the *Value* field for pattern variance, use a pipe character (|) to delimit the items in the list.

Pattern Variance examples [page 910]

Use pattern variance examples to help you understand how SAP Data Services outputs masked data using settings in the job.

Related Information

Data Mask transform Seed value [page 865]

3.6.2.10.1 Data Mask Pattern Variance

Use the *Pattern Variance Group* to mask an input field substring with a specific pattern variance type.

Pattern Variance option descriptions

Group/Option	Description
Pattern Variance group. Duplicate for each mapped input field that you want to mask.	
Mapped Input Field	Select the mapped input field for pattern variance from the dropdown list. The dropdown list includes all varchar input fields that you have mapped to the following transform input fields: `<Email[1-6]>`, `<Phone[1-6]>`, `<SSN[1-6]>`, or `<General_Data[1-12]>`.
Definition group: Complete at least one *Definition* group per *Pattern Variance* group to define a substring and set pattern variance options. You can duplicate the *Definition* group to define additional substrings of the input field. **i Note** When you include multiple *Definition* groups for a mapped input field, each group must specify a unique substring of the input field. The software issues a verification error if you have defined overlapping substrings for the specified input field.	
Starting Position	Specify the starting position for the substring. The software includes each alpha, numeric, space, and other printable character (special characters such as @, #, _, &) in the position count. The default starting position is 1. The software uses the *Starting Position* setting and the *Length* setting to identify the substring.

Group/Option	Description
Length	Specify the number of positions (characters) to include in the substring. The default length is blank. The software uses the *Length* setting and the *Starting Position* setting to identify the substring. Leave the *Length* field blank to randomize the mapped input field from the set *Starting Position* to the end of the string. For example, set the *Length* to **2** for a two-character substring. If the *Starting Position* is set to **5**, the substring consists of position five and six of the specified input field. Leave the *Length* blank in this example to mask all positions starting at position five to the end of the string.
Type of Variance	Specify the pattern variance type to apply to the defined substring of the *Mapped Input Field*. Choose one of the following pattern variance types: • *Preserve*. Outputs the defined substring the same as it is input. • *Character*. Masks the defined substring by randomly replacing each of the characters in the defined substring with values that you specify in the *Value* field. Retains spaces and special input characters in the output field. • *String*. Masks the defined substring by randomly replacing the entire substring with values that you specify in the *Value* field. Does not retain spaces or special input characters in the output field. • *Default*. Masks each applicable character with like characters for alpha and numeric content. Retains any special input characters and spaces in the output field.

Group/Option	Description
Value	**Note** You may leave this field blank for the preserve and the default pattern variance types. The software issues a validation error if you leave this field blank for any other pattern variance type. Specify alpha and numeric characters, spaces, and special characters for masking the substring. The values you enter must comply with the pattern variance type you choose. For example, when you choose the string pattern variance type, enter alpha or numeric strings or numeric ranges. String pattern variance does not accept alphabetic ranges. See each pattern variance type section for more information about what you can include in the *Value* field. When you include more than one value, separate each value with a pipe delimiter (|). For example, a value for the string pattern variance could include several characters that spell a word or an acronym; the entire string is considered one value. If you include another value after the first value, enter a pipe character (|) between the values to create a pipe-delimited list. For example: `BMI|BP|HR`

3.6.2.10.2 Pattern Variance input fields

During job set up, map varchar input fields to specific transform fields so they are available to select for pattern variance.

Map varchar input fields to specific transform fields that are listed under the *Transform Input Field Name* column in the *Input* tab. Applicable transform fields for pattern variance are:

- Email[1-6]
- Phone[1-6]
- SSN[1-6]
- General_Data[1-12]

After you perform this mapping, the software lists the fields in the *Mapped Input Field* under *Pattern Variance* in the *Options* tab.

> **Note**
> To mask date or numeric data types, use either mask out, date variance, or number variance instead of pattern variance.

Related Information

Data Mask transform input fields [page 849]

3.6.2.10.3 Pattern Variance type: Default

The default variance type replaces input characters with like characters and retains spaces and special characters.

The software masks each character with like characters for alpha and numeric content and retains any spaces and special characters, such as @, #, _, &.

When you select *Default* for Type of *Variance*, leave the *Value* option blank. The software ignores any values that you enter in the *Value* field.

The following table describes how the default pattern variance masks input characters with like characters.

Mask characters with like characters

Input character	Mask values
Alphabetic	• Masks lower-case alpha character with random lower-case alpha character • Masks upper-case alpha character with random upper-case alpha character
Numeric	Masks each numeric character with a random numeric character from 0 up to and including 9.
• Special character • Space	Does not mask special characters or spaces, but outputs them as they are input, unmasked. For example, when the input substring contains a dash (-), the default pattern variance keeps the dash in the output. When the input substring contains a space, the default pattern variance keeps the space in the output.

The following table describes best practices for using the default pattern variance.

Best practices

Description	Best practice	Example Definition options
Mask an entire input field using the default pattern variance.	Accept the default settings in the *Definition* group for a mapped input field.	Default *Definition* group settings: *Starting Position* = 1 *Length* = `<blank>` *Type of Variance* = Default *Value* = `<blank>`

Description	Best practice	Example Definition options
Automatically apply the default pattern variance to substrings of a mapped input field that are not defined.	Define input field substrings using one or more of the other pattern variance types, and leave portions of the input field undefined.	**Definition 1:** *Starting Position* = 1 *Length* = 3 *Type of Variance* = Preserve *Value* = `<blank>` **Definition 2:** *Starting Position* = 4 *Length* = 2 *Type of Variance* = String *Value* = 20-25|50-55|90-95 **Undefined:** Position six to the end of the field. **Results** **Definition 1:** The software preserves position one, two and three. **Definition 2:** The software masks the entire substring (position four and five) with a random number that is included in the ranges specified in the *Value* field (20-25, 50-55, or 90-95). **Undefined:** The software automatically masks position six to the end of the field with the default pattern variance.

3.6.2.10.4 Pattern Variance type: Preserve

For the variance type Preserve, the software outputs the defined substring as it was input, with no masking.

The software applies the default pattern variance to any undefined portions of the input field. Undefined portions are the sections of the input field that have not been defined with preserve, character, or string pattern variance.

The following table contains an example of the preserve pattern variance.

Example

The values listed in the **Input/Possible output** column in the following table show the masked substring for the preserve pattern variance in bold text.

Preserve pattern variance example

Strategy	Definition options	Input/Possible output	Notes
Preserve the unit identification number in each record. Mask the rest of the field with the default pattern variance. User sets up one *Definition* group.	**Undefined:** Character one is not defined. **Definition:** *Starting Position* = 2 *Length* = 2 *Type of Variance* = Preserve *Value* = `<ignored>` **Undefined:** Character three to the end of the field are not defined.	Input: A12:DER The following are possible output values: R**12**:KQG N**12**:UEP T**12**:RSL	**Undefined:** The software automatically masks the first position with a like character using the default pattern variance. **Definition:** The software preserves position two and three with the preserve pattern variance. **Undefined:** The software automatically masks the fourth position to the end of the string using the default pattern variance. • The colon in the input field (character four) is included in the undefined portion. The software outputs the colon as it was input based on the default pattern variance definition.

3.6.2.10.5 Pattern Variance type: Character

The character variance type masks each character in the defined substring with a character from the *Value* field.

The *Value* field can include the following values, or any combination of the values in a pipe-delimited list:

- Individual upper or lower case alpha characters
- Numeric characters from 0 to 9
- Ranges of alpha characters
- Ranges of numeric characters using numbers from 0 to 9
- Spaces
- Special characters, such as @, #, _, &

> **i Note**
>
> Each alpha and numeric value must be one character in length. Numerics must be 0 to 9, including ranges.

When you include an alpha or numeric range, list the minimum value first in the range followed by the maximum value. For example: “0-9” and not “9-0”; “A-Z” and not “Z-A”. The software uses values from the

Unicode table when it chooses masked values from a range. Therefore, masked values are the characters that fall into the stated range based on Unicode.

The software counts all alphanumeric characters, spaces, and special characters when it determines the substring length. Additionally, the software outputs special characters as they are input, without masking them, when they are present in a defined substring for character pattern variance.

Character and string variance types are similar. However, there are several differences between the two variance types.

Example

The values listed in the **Input/Possible output** column in the following table show the masked substring for the character pattern variance in bold text.

Character pattern variance

Strategy	Definition options	Input/Possible output	Notes
Mask an identification code with specific alpha or numeric values, and apply the default pattern variance to the remaining portion of the field. User sets up one *Definition* group.	**Undefined:** Position one. **Definition:** *Starting Position* = 2 *Length* = 2 *Type of Variance* = Character *Value* = J-L|B|W-Y|2 **Undefined:** Position three to the end of the string.	Input: 123a The following are possible output values: 8**KB**x 3**2W**t 7**LX**r	**Undefined:** The software automatically masks the first position using the default pattern variance. **Definition:** The software masks position two and three using the character pattern variance and randomly chooses a character specified in the *Value* field to mask each position. **Undefined:** The software masks the fourth position to the end of the field using the default pattern variance.

3.6.2.10.6 Pattern Variance type: String

For the variance type string, the software masks the entire defined substring with a random character or string from the *Value* field.

The *Value* field can include the following values, or any combination of the values in a pipe-delimited list:

- One or more alpha or numeric characters such as “MILK” or “2458”
- Spaces
- Special characters such as @, #, _, &
- Numeric ranges

When you include a numeric range, list the minimum value first in the range followed by the maximum value. For example: “0-9” and not “9-0”. The software uses values from the Unicode table when it chooses masked values from a range. Therefore the software uses values from the Unicode table that fall into the stated range.

The software counts all alphanumeric characters, spaces, and special characters when it determines the substring length. However, the software does not retain the special characters or spaces in the output when they are present in a defined substring for string pattern variance.

Character and string variance types are similar. However, there are several differences between the two variance types.

Example

The following table contains an example of the string pattern variance.

The values listed in the **Input/Possible output** column in the following table show the masked substring for the string pattern variance in bold text.

String pattern variance

Strategy	Definition options	Input/Possible output	Notes
Preserve the product code, but mask the type of milk (white, chocolate, soy, and so on) with the general term MILK. User sets up two *Definition* groups. **i Note** You could use mask out data masking for this example. However, when you use the pattern variance data masking, you can distinguish between parts of the whole string and have more control over the mask values.	**Definition 1:** *Starting Position* = 1 *Length* = 5 *Type of Variance* = Preserve *Value* = `<blank>` **Definition 2:** *Starting Position* = 6 *Length* = `<blank>` *Type of Variance* = String *Value* = MILK	Input: 5428-WTMLK 5429-CHMLK 5430-SOYMLK The following are possible output values: 5428-**MILK** 5429-**MILK** 5430-**MILK**	**Definition 1:** The software preserves the first through the fifth positions, including the dash, as part of preserve pattern variance. • The software includes the dash when determining the substring. Additionally, the software outputs the dash as it was input because it is included in the preserve pattern variance. **Definition 2:** The software masks position six to the end of the field with the value “MILK”.

3.6.2.10.7 Pattern Variance: Controlling replacement value

Control the randomness of character and string replacement by increasing the odds that SAP Data Services uses one replacement value over another.

Including a character or string multiple times in the *Value* option may increase the odds that the software chooses the character or string over other characters or strings.

> **Example**
>
> Increase the odds that the software chooses A over B with the following value list: `A|A|A|B`. The software uses the letter "A" about 75% more than the letter "B" when substituting characters in the defined substring.

> **Example**
>
> A value that includes `VOID|VOID|VOID|ACTIVE` causes the software to use the character string "VOID" about 75% more times than "ACTIVE".

This process is random, so the actual percent may not turn out to be exactly the ratio you have set in the substring definition.

3.6.2.10.8 Pattern Variance: Character vs. string processes

The software processes character and string pattern variances differently when the input *Value* field or the input substring contains specific character types.

Comparison of character and string pattern variance processes

Value field	Character	String
Single alpha or numeric characters. For example: `T|9|S`	Allowed. The software replaces each character in a defined substring with a single alpha or numeric character that is specified in the *Value* field. **Example** If substring contains five characters, software replaces each character with a single character from *Value*. Total of five replacement characters.	Allowed. The software replaces each defined substring (the entire substring) with a value that is specified in the *Value* field. **Example** If substring contains five characters, software replaces the five characters with a single character from *Value*. Total of one replacement character.
Strings of alpha or numeric characters. For example: `MILK|399|abc`	Not allowed. Character pattern variance accepts single alpha, numeric characters. The software issues an error if the *Value* list includes more than one character per value, except for ranges.	Allowed. The software replaces each defined substring with alpha or numeric strings that are specified in the *Value* field. **Example** The software replaces an input substring consisting of five characters with a string from the *Value* field that is ten characters.

Value field	Character	String

Alpha or numeric ranges. For example:
`D-M`
`2-9`
	Allowed. The software allows both alpha and numeric ranges. Alpha ranges can be from A to Z, upper or lower case. The numeric range can be single-digit numbers from 0 to 9.	Not allowed: Alpha ranges.
		Allowed: Numeric ranges.
		Numbers in ranges can have more than one digit, and they can include zero padding to the left. For example:
		`005-250`

Spaces included with alpha characters or special characters. For example: `*| - | a` (space before asterisks, space before and after dash, space before letter "a")	Not allowed. The software issues an error if the *Value* field includes more than one character, including spaces. **i Note** Single spaces and single special characters are allowed. For example, the values `*|  |a` (asterisk, space, letter "a") are allowed in the *Value* field.	Allowed. The software replaces the defined substring with a value from the *Value* field, including the spaces. **i Note** When the defined substring includes a space in the input, the software does not retain the space on output.

Value field	Character	String
Zero-padded individual numbers and zero-padded numbers in a range. For example: `05|010-350`	Not allowed. The software issues an error if the *Value* field includes more than one character. The software allows the single-digit numbers from 0 to 9 stated individually or in a range. **Example** The values `8|9|0-5` include numbers 8, 9, 0, 1, 2, 3, 4, and 5 for replacement values.	Allowed. The software allows zero-padded numbers in the *Value* field for individual numbers or numeric ranges. When the defined substring contains more characters than a zero-padded number or numeric range in the value list, the software adds zeros to the left of the number to the length of the substring. **Example** The software replaces the four-character substring "1250" with "0005" even when the listed value is "05". **Example** The values `8|9|0-5` include numbers 8, 9, 0, 1, 2, 3, 4, and 5 for replacement values. Possible masked values are "0010" or "0350".

3.6.2.10.9 Pattern Variance: Use pipe delimiters for multiple values

When you list multiple characters or strings in the *Value* field for pattern variance, use a pipe character (|) to delimit the items in the list.

You can include more than one value in the *Value* option when you define your pattern. When you list more than one value, use the pipe character (|) between values.

If the values or strings that you include in the value list contain one of the following characters, use a back slash (\) as an escape character:

- Pipe (|)
- Dash (-)
- Back slash (\)

The escape character tells SAP Data Services that the character following the escape character is part of the string.

Example

The following table contains pipe-delimited value lists. The examples demonstrate how to use the back slash to indicate that the character following the escape character is part of the string. The examples also demonstrate how to include a space or a leading zero in the value lists.

Multiple values, escape character, zeros, and spaces

Example	Description
`A\-C`	Include a dash character (-) in your value list by preceding it with the back slash (\) escape character. **Example** For string pattern variance, the software replaces the defined substring with "`A-C`". The replacement value is not an inclusive range from A to C.
`A-C`	Include a dash character (-) to indicate a range. Ranges can be alpha or numeric for character pattern variance, and only numeric for string pattern variance. **Example** For character pattern variance, the software replaces each character in a defined substring with A, B, or C.
`\\|<another value>`	Include a back slash (\) character in your value list by preceding it with a back slash as an escape character. Applies to either character type or string type pattern variance. **Example** The first back slash is the escape character that indicates to use a back slash character in the mask value output.
`||<another value>`	Include the pipe character (|) in your value list by preceding it with a back slash (\) as an escape character. Applies to either character type or string type pattern variance. **Example** The back slash is the escape character that indicates to use the pipe character in the mask value output. The second pipe is a value separater between the first value and the next value.

Example	Description
`A|545`	For string pattern variance, include both alpha and numeric values in a value list by separating them with the pipe character (|) as a delimiter. For string variance, include values of variable length as the example at left shows. **Example** A job setup uses the following definitions for a select input field: **Definition 1:** *Starting Position*: 1 *Length*: 5 *Type of Variance*: Preserve *Value*: `<ignored>` **Definition 2:** *Starting Position*: 6 *Length*: 3 *Type of Variance*: String *Value*: A|545 For the input field value "MLK63GAL", the software could output variable lengths such as: • MLK63**545** • MLK63**A**

Example	Description
`1-3|X-Z|a`	For character pattern variance, include both alpha and numeric characters in a value list as the example shows at left. The software randomly chooses a masked value without regard to the type of character being masked. For example, the software could replace a numeric character with an alpha character from the value list. **Example** The job setup for the example at left includes the following Definition groups: **Definition 1:** *Starting Position*: 1 *Length*: 5 *Type of Variance*: Preserve *Value*: `<ignored>` **Definition 2:** *Starting Position*: 6 *Length*: `<blank>` *Type of Variance*: String *Value*: 1-3|X-Z|a Possible output values for the input field "MLK63GAL" are: • MLK63**X** • MLK63**2**
| `ab` | <another value>	For string or character pattern variance, include a space as a value, or include a space before or after alpha characters. **Example** The software replaces a defined substring with "`ab`", a space, and two alpha characters, no matter how many characters the substring contains.

Example	Description
07-99	For string pattern variance, control the number of characters in a mask value by adding zeros to the left of a number. **Example** The user wants to replace the defined substring with a value that has the same number of characters as the defined *Length* of the substring. If the *Length* is 4, possible masked values could be 0008, or 0075. **Note** The user needed to include only one zero to the left of the first number or the second number in the range. The user could include a zero before both numbers to get the same result (07-099), but it isn't necessary.
`A\-\\C\\%`	Include multiple characters in a pipe-delimited list by: • Separating each value with the pipe character • Preceding each applicable special character with a back slash as an escape character **Example** For the example at left, the software uses the characters in the following list to mask a defined substring based on the character pattern variance type: • `A` • `-` • `\` • `C` • `\` • `%` The value list includes the back slash character (\) as an escape character to include the dash (-), back slash (\), and pipe (|) characters.

3.6.2.10.10 Pattern Variance examples

Use pattern variance examples to help you understand how SAP Data Services outputs masked data using settings in the job.

The following table contains examples of various pattern variance types by showing example *Definition* option settings, input values, and possible output values. The notes column describes the software behavior.

Strategy	Definition options	Input/Possible output	Notes
Mask the weight and the unit of measure from a product code, but preserve the product type. User sets up two *Definition* groups for the same input field.	**Definition 1:** *Starting Position*: 1 *Length*: 3 *Type of Variance*: Preserve *Value*: `<ignored>` **Undefined:** Position 4 and 5. **Definition 2:** *Starting Position*: 6 *Length*: 3 *Type of Variance*: String *Value*: GAL|qt|pt|oz|CUP	Input: MLK12CUP The following are possible output strings: MLK63GAL MLK18pt MLK04oz	**Definition 1:** Preserves the first three positions using the preserve pattern variance. **Undefined:** Masks the fourth and fifth positions using the default pattern variance, which replaces each numeric character with a value from 0-9. **Definition 2:** Masks the sixth through the eighth positions with one of the values listed using the string pattern variance. • Notice that in some cases, the software replaces a 3-character substring with a 2-character value.
Mask the product type and the weight from a product code, but preserve the unit of measure. User sets up three *Definition* groups for the same input field.	**Definition 1:** *Starting Position*: 1 *Length*: 3 *Type of Variance*: String *Value*: ALMLK|SOYMLK|RCEMLK|WTMLK|CHMLK **Definition 2:** *Starting Position*: 4 *Length*: 2 *Type of Variance*: String *Value*: 01-12|32|16 **Definition 3:** *Starting Position*: 6 *Length*: `<blank>` *Type of Variance*: Preserve *Value*: `<blank>`	Input: MLK12CUP The following are possible output string: WTMLK32CUP RCEMLK16CUP ALMLK08CUP	**Definition 1:** Masks the first three positions using one of the values specified for string pattern variance. • The software masks the 3-character substring with values that may be longer than 3 characters. **Definition 2:** Masks the fourth and fifth positions using the string pattern variance. • The first value listed in the *Value* field for Definition 2 is a range beginning with a zero-padded number. This ensures that the mask value is the length of the defined substring, 2 characters. **Definition 3:** Preserves the sixth through the eighth positions.

Strategy	Definition options	Input/Possible output	Notes
Mask the number of paper sheets per package, and the type of packaging from the product description field. User sets up two *Definition* groups.	**Definition 1:** *Starting Position*: 1 *Length*: 4 *Type of Variance*: Character *Value*: 0-9 **Undefined:** Position 5. **Definition 2:** *Starting Position*: 6 *Length*: `<blank>` *Type of Variance*: String *Value*: Ream|Case|Pack|Box	Input: 1500/Ream The following are possible output strings: 0950/Case 8945/Box 2639/Pack	**Definition 1:** Masks the first through the fourth positions with a number from 0-9. • The user could leave the first through the fifth position undefined so the software masks the substring using the default pattern variance to get similar output values. The forward slash would be output as part of the substring in this case. **Undefined:** Outputs the forward slash (/) character in the fifth position using the default pattern variance (maintains special characters on output). **Definition 2:** Mask the sixth position to the end of the field with one of the character strings listed in the *Value* field.
Mask the school district, the state, and the enrollment number. Preserve the type of school. User sets up three *Definition* groups.	**Definition 1:** *Starting Position*: 1 *Length*: 3 *Type of Variance*: String *Value*: DST **Definition 2:** *Starting Position*: 4 *Length*: 2 *Type of Variance*: String *Value*: ST **Undefined:** Position 6, 7, 8, and 9. **Definition 3:** *Starting Position*: 10 *Length*: `<blank>` *Type of Variance:* Preserve *Value*: `<ignored>`	Input: INDNE7321MID BANMA7321HIGH SNBCA7321ELEM Possible output: DSTST3829MID DSTST5784HIGH DSTST0789ELEM **i Note** The mask out variance could also mask the fields in this example. However, with pattern variance, you can distinguish between parts of the whole string and have more control over the mask values.	**Definition 1:** Masks position one to three with the string "DST". **Definition 2:** Mask the fourth and the fifth position with the string "ST". **Undefined:** Automatically masks position six through nine with the default pattern variance. **Definition 3:** Preserves the tenth position to the end of the field.

3.6.2.11 Data Mask transform number generalization

Use the *Number Generalization Group* to specify options that anonymize data based on ranges and replacement values, and default replacement values.

Number generalization for data masking works with all number-based input fields that you have mapped to `<Numeric_Data[1-6]>`. Map the fields in the *Input* tab under the *Transform Input Field Name* column in the transform editor. Input fields can be of any data type.

If the input field contains non-numeric data combined with numeric-based data, the software parses the first number from the input field to use for generalization. Under certain circumstances, the software cannot represent all numbers.

Use the *Number Generalization Group* multiple times, but only once for each input field that you want to generalize.

Number Generalization group options [page 913]
Use the options in the *Number Generalization Group* to mask output data with a set replacement value or a software-generated range.

Number Generalization examples of generated number ranges [page 915]
Use the following examples to help you understand how the software generates number ranges for number-generalized output values for Data Masking.

Number Generalization examples [page 916]
Use examples to understand how the number generalization feature works in the Data Mask transform.

3.6.2.11.1 Number Generalization group options

Use the options in the *Number Generalization Group* to mask output data with a set replacement value or a software-generated range.

Number Generalization Group, subgroups, and options

Group/Option	Description
Number Generalization Group (duplicate for each input field that you want to mask)	
Mapped Input Field	Input field to output with number generalization masking. **i Note** The dropdown list only includes the fields that you mapped to Numeric_Data[1-6].

Group/Option	Description
Default Replacement Value	Value to output when the input field value does not fall into any of the defined ranges. Default is *Other*. This field cannot be blank. Enter your own value or leave the default setting.
Range Definition (duplicate to create multiple ranges for the same input field)	
Minimum Value	Lowest value for the numeric range. Default is blank. If you leave this blank, the software requires that you set a value for *Maximum Value*. The following situations apply to *Minimum Value*: • If you enter a *Minimum Value* but leave the *Maximum Value* blank, the range is all values greater than the minimum value. • The *Minimum Value* cannot be greater than the *Maximum Value*. • The *Minimum Value* can be greater than or equal to -4294967296.000000 and less than or equal to 4294967295.000000.
Minimum Value Inclusive	Include the minimum value in the numeric range. • *Yes* = Default setting. Include the value in the range. • *No* = Do not include the value in the range.
Maximum Value	Highest value for the numeric range. Default is blank. If you leave this blank, the software requires that you set a *Minimum Value*. The following situations apply to *Maximum Value*: • If you enter a *Maximum Value* but leave the *Minimum Value* blank, the range is all values less than the maximum value. • The *Maximum Value* cannot be less than the *Minimum Value*. • The value can be greater than or equal to -4294967296.000000 and less than or equal to 4294967295.000000.
Maximum Value Inclusive	Include the maximum value in the numeric range. • *Yes* = Default setting. Include the value in the range. • *No* = Do not include the value in the range.

Group/Option	Description
Replacement Value	Output value when the input value falls within the defined range. If you leave this field blank, the software generates a range based on the values that you set for the minimum and maximum options.

Related Information

Number variance number formats [page 877]

3.6.2.11.2 Number Generalization examples of generated number ranges

Use the following examples to help you understand how the software generates number ranges for number-generalized output values for Data Masking.

If you leave the *Replacement Value* in the *Range Definition* group blank, and the input value falls into the defined range, the replacement value is the software-generated number range.

Example

If you leave the *Replacement Value* blank, the software generates an output value based on the given range. For example:

- *Minimum Value* = 1
- *Minimum Value Inclusive* = Yes
- *Maximum Value* = 10
- *Maximum Value Inclusive* = No

Output value is: `[1,10)`, where "[" indicates inclusive, and ")" indicates exclusive.

Example

The following table contains descriptions of possible software-generated output ranges that include mathematical symbols to indicate inclusion or exclusion of values.

Number ranges with two values

Range output	Description
[1,4]	Low and high values are inclusive: 1 <= value <= 4
[1,4)	Low value inclusive. High value not inclusive: 1 <= value < 4

Range output	Description
(1,4]	Low value not inclusive. High value inclusive: 1 < value <= 4
(1,4)	Low value and high value are not inclusive: 1 < value < 4

Example

The following table contains descriptions of possible software-generated output ranges that contain a single value or open-ended ranges.

Single-value or open-ended ranges

Range output	Description
>= 4	All values greater than or equal to 4.
> 4	All values greater than 4.
<= 4	All values less than or equal to 4.
< 4	All values less than 4.
4	The value 4

3.6.2.11.3 Number Generalization examples

Use examples to understand how the number generalization feature works in the Data Mask transform.

Example

You have a database that includes age data. You want to group the records into the following groups:

- First range is for ages 12 or younger. Replace with *Default Replacement Value* “Other”.
- Second range is for ages 13 to 19. Replace with “Teenager”.
- Third range is for ages 20 to 24. Replace with “Young Adult”.
- Fourth range is for ages 30 and older. Replace with “Adult”.

Job set up:

- *Mapped Input Field* = Age
- *Default Replacement Value* = Other
- *Range Definition* (1)

  Minimum Value =
  Minimum Value Inclusive = YES
  Maximum Value = 12
  Maximum Value Inclusive = YES
  Replacement Value =

- *Range Definition* (2)

  Minimum Value = 13
  Minimum Value Inclusive = YES
  Maximum Value = 19
  Maximum Value Inclusive = YES
  Replacement Value = Teenager
- *Range Definition* (3)

  Minimum Value = 19
  Minimum Value Inclusive = NO
  Maximum Value = 25
  Maximum Value Inclusive = NO
  Replacement Value = Young Adult
- *Range Definition* (4)

  Minimum Value = 30
  Minimum Value Inclusive = YES
  Maximum Value =
  Maximum Value Inclusive = YES
  Replacement Value = Adult

Sample data

Input	Output	Explanation
4	<=12	Falls into the first range, which has no minimum. The *Replacement Value* is blank so the software outputs the software-generated range.
12	<=12	Falls into the first range, which has no minimum. The maximum value is inclusive. The *Replacement Value* is blank so the software outputs the software-generated range.
13	Teenager	Falls into the second range because it is equal to the minimum value of 13. Minimum value is inclusive.
17	Teenager	Falls into the second range because it is greater than 13 but less than 19.
19	Teenager	Falls into the second range because it is equal to the maximum of 19. Maximum value is inclusive.

Input	Output	Explanation
25	Other	The third range specifies 25 as the *Maximum*, but the *Maximum Number Inclusive* is set to *NO*. Therefore, 25 does not fall into any range so the software outputs the *Default Replacement Value*.
32	Adult	Falls into the fourth range because it is greater than 30.
$13.75	Teenager	Falls into the second range. The software parses the field until it finds the first number, which is 13.75. The number 13.75 is within the range of 13 - 19.
Hi mom	NULL	The software cannot parse number-based data from the input data. Therefore the output is NULL.
NULL	NULL	The software cannot parse number-based data from the input data. Therefore the output is NULL.

3.6.2.12 Data Mask transform Date Generalization

To mask output data with a common label or date range, use the Data Mask Date Generalization feature.

Date generalization for data masking works with all date-based input fields that you map to `<Date[1-6]>`. Map the fields in the input tab under the *Transform Input Field Name* column in the transform editor. Input fields can be of any data type.

Use the *Date Generalization Group* once for each input field that you want to generalize.

There are two types of ranges:

- *Range Definition*: You define the date ranges. Use the *Range Definition* group multiple times on an eligible input field.
- *Auto Range Definition*: The software calculates date ranges based on your settings. Use the *Auto Range Definition* group once per eligible input field.

> **i Note**
>
> Define only one type of range for a selected input field.

Additionally, designate a default replacement value when dates do not fit into the defined ranges.

Date Generalization group options [page 919]

Use *Date Generalization* options to set range definition options and a default replacement value on specified mapped input fields.

3.6.2.12.1 Date Generalization group options

Use *Date Generalization* options to set range definition options and a default replacement value on specified mapped input fields.

Date Generalization Group, subgroups, and options

Group/Option	Description
Date Generalization Group (duplicate for each input field that you want to mask)	

Group/Option	Description
Mapped Input Field	Input field to output with date generalization masking. **i Note** The dropdown list only includes the fields that you mapped to Date[1-6].
Default Replacement Value	Value to output when the input field value does not fall into any of the defined ranges. The default is *Other*. This field cannot be blank. Enter your own value or leave the default setting.
Range Definition (duplicate to set multiple ranges for the same input field)	
Minimum Date	Earliest date for the range. Default is blank. If you leave the option blank, the software requires that you enter a date for the *Maximum Date* option. The following situations apply to *Minimum Date*: • The *Minimum Date* cannot be later than or equal to the *Maximum Date*. • If you specify a *Minimum Date* but you do not specify a *Maximum Date*, the software uses a date range of later than or equal to the *Minimum Date*. • The *Minimum Date* cannot be earlier than January 1, 1900.
Minimum Date Inclusive	Specifies to include the value for *Minimum Date* in the date range. • *Yes* = Default setting. Include in the range. • *No* = Do not include in the range.

Group/Option	Description
Maximum Date	Maximum date for the range. Default is blank. If you leave the option blank, the software requires that you enter a date for the *Minimum Date* option. The following situations apply to *Maximum Date*: • The *Maximum Date* cannot be earlier than or equal to the *Minimum Date*. • If you specify a *Maximum Date* but you do not specify a *Minimum Date*, the software uses a range of earlier than or equal to the *Maximum Date*. • The *Maximum Date* cannot be later than December 31, 2099.
Maximum Date Inclusive	Specifies to include the value for *Maximum Date* in the date range. • *Yes* = Default setting. Include in the range. • *No* = Do not include in the range.
Replacement Value	Specifies the output value when the input value falls within the defined range in the *Range Definition* options. Default is blank. When you leave the *Replacement Value* blank, the software outputs a system-generated range based on your settings.
Auto Range Definition (Use once per applicable input field. If you have defined the Range Definition group for the same input field, you may not use this group.)	

Group/Option	Description
Auto Range Scale	Defines the scale on which to base the auto range. • *Not_In_Use* = Default setting. Indicates that you are not using auto range for the specified input field. This setting is appropriate when you complete the *Range Definition* options for the input field, or when you do not use this feature. • *Calendar_Year* = Group records based on the calendar year. The software defines a calendar year as 1/1/yyyy to 12/31/yyyy. • *Calendar_Month* = Group records based on the calendar month. The software defines a calendar month as mm/01/yyyy to mm/eom/yyyy, where "eom" is end of month. **i Note** For both *Calendar_Year* and *Calendar_Month*, the software can use settings that you make in the *Start Date (Inclusive)* and *End Date (Inclusive)* as part of the range. If you choose a setting that is not compatible with the *Date Format* in the *Auto Range Scale* option, the software issues an error. For example, the software issues an error when *Auto Range Scale* is set to *Calendar_Year* and *Date Format* is set to *Month*.
Duration	Number of years or months to include in the range. Default is *1*. If you set the *Duration* to greater than 1, you must also set a value for *Start Date (Inclusive)*. If you do not set a start date and the duration is set to greater than 1, the software issues an error.

Group/Option	Description
Start Date (Inclusive)	Starting date in auto range. Default is blank. The following situations apply to *Start Date (Inclusive)*: • If you enter a *Start Date (Inclusive)* and an *End Date (Inclusive)*, the start date must be earlier than the end date. • If the *Duration* is set to 1, you can leave the *Start Date (Inclusive)* and *End Date (Inclusive)* blank. • The minimum start date cannot be earlier than January 1, 1900. The software outputs the *Default Replacement Value* for all dates that fall outside of the range.
End Date (Inclusive)	Ending date in auto range. Optional. Default is blank. The software uses this date as the last date in the calculated range. The following situations apply to *End Date (Inclusive)*: • If you enter both a *Start Date (Inclusive)* and an *End Date (Inclusive)*, the end date must be later than the start date. • If the *Duration* is set to 1, you can leave the *Start Date (Inclusive)* and *End Date (Inclusive)* blank. • The maximum end date cannot be later than December 31, 2099. The software outputs the *Default Replacement Value* for all dates that fall outside of the range.
Auto Range Replacement Value (specific to the auto range defined in the parent group *Auto Range Definition*)	

Group/Option	Description
Date Format	Determines the format of the output *Auto Range Replacement Value*. • *Month_Day_Year* Default setting • *Day_Month_Year* • *Year_Day_Month* • *Year_Month_Day* • *Year* • *Year_Month* • *Month_Year* • *Month* **i Note** If the setting that you choose is not compatible with the setting in *Auto Range Scale*, the software issues an error.
Date Delimiter	Determines the delimiter to use in the *Auto Range Replacement Value*. • *Slash* Default setting • *Dash* • *Backslash* • *Space* • *None* • *Period* • *Chinese_Japanese*
Numeric Format	Determines the numeric format to use in the *Auto Range Replacement Value*. • *Arabic_Numbers* Default setting • *Chinese_Japanese_Numbers*
Enable Zero Pad	Pad a one-digit number with zero when the format includes the month and day. • *No* = Default setting. Do not pad dates with zeros. Example: Jan 2, 2016 is output as Jan 2 2016. • *Yes* = Pad dates with zeros. Example: Jan 2, 2016 is output as Jan 02 2016.

Group/Option	Description
Month Format	Determines the month format to use in the *Auto Range Replacement Value*. • *Numeric* = Default setting. Represent the month with a number. • *Short_Text* = Represent the month using the month abbreviation. • *Full_Text* = Represent the month using the full spelling.
Language	Determines the language to use in the *Auto Range Replacement Value*. This setting is applicable when the *Month Format* is set to *Short_Text* or *Full_Text*. *English* Default setting **i Note** The software does not verify that the user-defined default language corresponds to the language of the input month.
Year Format	Specifies the number of digits to use for the year. • *Full_Year* = Default setting. Always output a 4-digit year. • *Short_Year* = Always output a 2-digit year.

3.6.2.12.2 Date Generalization user-defined and system generated range definitions

To identify specific input data and to generalize the output with a set value, define date ranges in the *Range Definition* group .

Use the *Range Definition* options multiple times for a single input field to set multiple ranges.

> **i Note**
>
> If you set multiple ranges for the same input field, make sure that the ranges do not overlap.

> **i Note**
>
> When you use *Range Definition* for a specific input field, you cannot also use *Auto Range Definition* for the same input field.

Enter a value for *Replacement Value* for each range that you set. Leave *Replacement Value* blank to use the default replacement value.

- Set a value to output when an input date meets the range parameters.

- Accept the default, which is blank. When an input date meets the range parameters, the software generates a date range to output based on your settings in the *Range Definition* group.

How system-generated ranges work

For calendar year ranges, the software starts the first range at the *Start Date (inclusive)* and ends the range on December 31, `<year>`. The software calculates the range based on the set *Duration*. Each subsequent range starts on January 1 and ends on December 31 of the calculated year. The last range ends on the set *End Date (inclusive)*.

For calendar month ranges, the software starts the first range at the *Start Date (inclusive)* and ends the range on the last day of the appropriate month and calculated year. The software calculates the range based on the set *Duration*. Each subsequent range starts on the first day of the month and ends on the last day of the month of the calculated year. The last range ends on the date specified in the *End Date (inclusive)*.

How the generated ranges are output

The software outputs a generated range using mathematical symbols that indicate less than, equal to, inclusive, or exclusive. The following are examples of output generated ranges.

Example

Date ranges with two values:

Range output	Description
[3/17/2017, 7/4/2017]	Both low and high dates are inclusive: March 17, 2017 <= date <= July 4, 2017
[3/17/2017, 7/4/2017)	Low date inclusive. High date not inclusive: March 17, 2017 <= date < July 4, 2017
(3/17/2017, 7/4/2017]	Low date not inclusive. High date inclusive: March 17, 2017 < date <= July 4, 2017
(3/17/2017, 7/4/2017)	Low and high dates are not inclusive: March 17, 2017 < date < July 4, 2017

Example

Single-value or open-ended ranges

Range output	Description
>= 3/17/2017	All dates later than or equal to March 17, 2017.

Range output	Description
> 3/17/2017	All dates later than March 17, 2017.
<= 3/17/2017	All dates earlier than or equal to March 17, 2017.
< 3/17/2017	All dates earlier than March 17, 2017.
3/17/2017	The date is March 17, 2017

3.6.2.12.3 Date Generalization range definition examples

Use examples to understand how to set date ranges in the *Range Definition* options in the *Date Generalization Group*.

Example

You want to customize an advertising campaign based on generation groups. You create three databases, each containing records that qualify for the generational groups based on birth date. The generational groups are:

- Baby Boomer
- Generation X
- Millennial

Date Generalization Group, *Range Definition* option settings:

Date Generalization Group

- *Date Generalization*

  Mapped Input Field = BirthDate
  Default Replacement Value = Other

- *Range Definition* (1)

  Minimum Date =
  Minimum Date Inclusive = YES
  Maximum Date = January 1, 1945
  Maximum Date Inclusive = NO
  Replacement Value = N/A
- *Range Definition* (2)

  Minimum Date = Jan 1, 1945
  Minimum Date Inclusive = YES
  Maximum Date = 12/31/1964
  Maximum Date Inclusive = YES
  Replacement Value = Baby Boomer
- *Range Definition* (3)

Minimum Date = Jan 1, 1965
Minimum Date Inclusive = YES
Maximum Date = 12/31/1976
Maximum Date Inclusive = YES
Replacement Value = GEN-X

- *Range Definition* (4)

  Minimum Date = 01/01/1977
  Minimum Date Inclusive = YES
  Maximum Date = December 31, 1995
  Maximum Date Inclusive = YES
  Replacement Value = Millennial

Sample data

Input	Output	Explanation
7/31/1924	N/A	The input date falls into the first range definition of less than January 1, 1945. The software outputs the replacement value "N/A"
1/1/45	Baby Boomer	The input date falls into the second range definition. 1/1/45 is greater than or equal to 1/1/1945 and less than or equal to 12/31/1964.
Nov 4, 1974	GEN-X	The input date falls into the third range definition. Nov 4, 1974 is greater than or equal to 1/1/1965 and less than or equal to 12/31/1976.
July 4, 1988	Millennial	The input date falls into the fourth date range definition. July 4, 1988 is greater than or equal to 1/1/1977 and less than or equal to 12/31/1995.
December 6, 2016	Other	The input date does not fall into any defined range. December 6, 2016 is greater than the maximum date of 12/31/1995. Therefore, the software outputs the *Default Replacement Value* "Other".
Born: 11/4/1960	Baby Boomer	The input data contains both text and date-based data. The software parses the date data. The input date falls into the second date range definition. 11/4/1960 is greater than or equal to 1/1/1945 and less than or equal to 12/31/1964.

Input	Output	Explanation
NULL	NULL	The input does not contain any parsable date data. The software issues a warning.
Hi Mom!	NULL	The input does not contain any parsable date data. The software issues a warning.

3.6.2.12.4 Date Generalization calculated date ranges

Use options in the *Auto Range Definition* group to have SAP Data Services calculate date ranges to use for date generalization.

Use the *Auto Range Definition* options once per input field.

> **i Note**
>
> When you use *Auto Range Definition* for a specific input field, you cannot also use *Range Definition* for the same input field.

Unlike the multiple ranges you can set for an input field in the user-define date range options, you enter a start and end date and the software calculates all of the ranges.

Instead of setting a value for an output replacement, the software calculates ranges to output. You define the format of the generated ranges by making settings in the *Auto Range Replacement Value* group.

How the software calculates ranges

- The software calculates ranges based on calendar year in the following manner:
  - The first range starts at the set *Start Date (Inclusive)* and goes to December 31 of the year specified by the *Duration* setting.
  - All subsequent ranges start on January 1 and end on December 31 of the calculated year.
  - The last range starts on January 1 of the calculated year and ends on the specified *End Date (Inclusive)* setting.
- The software calculates ranges based on calendar month in the following manner:
  - The first range starts at the set *Start Date (Inclusive)* and goes to the last day of the stated month based on the *Duration* setting.
  - All subsequent ranges start on the first of the calculated month and end on the last day of the calculated month.
  - If the user has set an *End Date (Inclusive)* date, the last range starts on the first of the calculated month and ends on the specified *End Date (Inclusive)* setting.

3.6.2.12.5 Date Generalization caluclated date range examples

Use examples to understand how the software calculates auto-defined ranges based on your settings in the *Date Generalization Group*.

Example

You want to run date generalization on the `<PublishDate>` field in your database. You want to group the dates in five-year durations beginning with January 1, 2000. If any date is earlier than the start date of January 1, 2000, you want the software to output "Not this millennium" in the `<PublishDate>` field.

- *Date Generalization Group*

  Mapped Input Field = PublishDate
  Default Replacement Value = Not this millennium
- *Range Definition*

  You do not enter values for this group because the software does not allow both range definition and auto range definition for the same input field.
  Minimum Date =
  Minimum Date Inclusive = YES
  Maximum Date =
  Maximum Date Inclusive = YES
  Replacement Value =
- *Auto Range Definition*

  Auto Range Scale = CALENDAR_YEAR
  Duration = 5
  Start Date (Inclusive) = January 1, 2000
  End Date (Inclusive) =
- *Auto Range Replacement Value*

  Date Format = YEAR_MONTH
  Date Delimiter = SLASH
  Numeric Format = ARABIC_NUMBERS
  Enable Zero Pad = YES
  Month Format = NUMERIC
  Language = ENGLISH
  Year Format = FULL_YEAR

Sample data

Input	Output	Explanation
7/31/1924	Not this millennium	The input date is out of range because it is earlier than the start date of January 1, 2000. The software outputs the *Default Replacement Value*.

Input	Output	Explanation
1/1/2000	[2000/01, 2004/12]	The input date is in range because it is equal to the *Start Date (Inclusive)* value. The output is in YEAR_MONTH date format as specified in the options.
Dec 6, 2016	[2015/01, 2019/12]	The input date is in range because it is later than the *Start Date (Inclusive)* and there is no *End Date (Inclusive)*.
Date: 2016	NULL	The input does not contain any parsable date data. The software issues a warning that the date could not be parsed.
December 10	NULL	The input does not contain any parsable date data. The software issues a warning that the date could not be parsed.

3.6.2.12.6 Date Generalization calculated partial ranges

For calendar year and calendar month ranges, SAP Data Services uses a partial range for the first or last range of the duration when the start or end date isn't on the first or last day of the calendar year or month.

Data Services calculates auto-ranges based on your settings in the *Auto Range Definition* group for either a calendar month or a calendar year. Entering a start date for the first of a year and an end date for the last day of the year enables Data Services to calculate full year ranges. However, enter a start date that doesn't start on the first of the year, 1/1/yyyy, causes the first year range to be a partial year. Enter an end date that doesn't end on the last day of the year, 12/31/yyyy, causes the last year range to be a partial year. The same reasoning is true for month calculations. Enter the start date that is not mm/1/yyyy and the first range is a partial month. Enter the end date that is not mm/31/yyyy and the last range is partial month.

Data Services doesn't require that you enter the start date and end date so that the software calculates full durations.

i Note

If you do not enter a start or end date, the software requires that you set the *Duration* to 1.

Example

For *Auto Range Scale* of *CALENDAR_YEAR* with *Duration* of *1*, the start dates result in the calculated ranges shown in the following table:

Full year	Partial year
Start Date (inclusive) = 1/1/2017 **End Date (inclusive) = 12/31/2020**	**Start Date (inclusive) = 7/4/2017** **End Date (inclusive) = 12/31/2020**
First calculated range = 1/1/2017 to 12/31/2017	First calculated range = 7/4/2017 to 12/31/2017
Second calculated range = 1/1/2018 to 12/31/2018	Second calculated range = 1/1/2018 to 12/31/2018
Third calculated range = 1/1/2019 to 12/31/2019	Third calculated range = 1/1/2019 to 12/31/2019
Fourth calculated range = 1/1/2020 to 12/31/2020	Fourth calculated range = 1/1/2020 to 12/31/2020

To further illustrate full and partial year and month range calculations, see the examples for both calendar year and calendar month.

3.6.2.12.7 Date Generalization calculated ranges based on calendar year

Use examples to understand how the software calculates and outputs date ranges based on calendar year and date format.

Example

- *Auto Range Definition*
  Auto Range Scale = CALENDAR_YEAR
  Duration = 1
  Start Date (Inclusive) =
  End Date (Inclusive) =
- *Auto Range Replacement Value*

  Date Format = `<see table>`
  Date Delimiter = SLASH
  Numeric Format = ARABIC_NUMBERS
  Enable Zero Pad = YES
  Month Format = NUMERIC
  Language = ENGLISH
  Year Format = FULL_YEAR

The software calculates the ranges based on the input date because there are no start and end dates specified. The input date is July 4, 2016. The output values differ based on the set *Date Format* in the first column. All dates in the Output column are inclusive so the ranges are enclosed with square brackets.

Note that not all Date Format values are shown in the examples

Date Format =	Input date = July 4, 2016 Output	Notes
YEAR_DAY_MONTH	[2016/01/01, 2016/31/12]	The output appears in yyyy/dd/mm format.
YEAR_MONTH_DAY	[2016/01/01, 2016/12/31]	The output appears in yyyy/mm/dd format.
YEAR	2016	The output is not a range. Because the scale is calendar year, and the duration is 1, the only possible output is the year. The format is yyyy.
YEAR_MONTH	[2016/01, 2016/12]	The output appears in yyyy/mm format.
MONTH	NULL	No output. This date format causes the software to issue an error. The *Auto Range Scale* of Calendar Year and the *Date Format* are not compatible.

Example

- *Auto Range Definition*

  Auto Range Scale = CALENDAR_YEAR
  Duration = 2
  Start Date (Inclusive) = July 15, 2013
  End Date (Inclusive) =
- *Auto Range Replacement Value*

  Date Format = `<see table>`
  Date Delimiter = DASH
  Numeric Format = ARABIC_NUMBERS
  Enable Zero Pad = NO
  Month Format = FULL_TEXT
  Language = ENGLISH
  Year Format = FULL_YEAR

The start date begins in the middle of a year. Therefore, the first software-calculated 2-year range isn't a full two years. The first two software-calculated date ranges are as follows:

- First 2-year range: July 15, 2013 to December 31, 2014
- Second 2-year range: January 1, 2015 to December 31, 2016

In the following table, the input date is July 4, 2016. This date falls into the second calculated range. The output values differ based on the set *Date Format* in the first column. All dates in the Output column are inclusive so the ranges are enclosed with square brackets.

Note that not all Date Format values are shown in the examples

Date Format	Input date = July 4, 2016 Output	Notes
DAY_MONTH_YEAR	[1-January-2015, 31-December-2016]	The output appears with dashes and full text month format: d-full text month-yyyy.
YEAR	[2015, 2016]	The output appears with dashes and includes just the year format: yyyy.
YEAR_DAY_MONTH	[2015-1-January, 2016-31-December]	The output appears with dashes and full text month format: yyyy-d-full text month.
MONTH_DAY_YEAR	[January-1-2015, December-31-2016]	The output appears with dashes and full text month format: full text month-d-yyyy.

Example

- *Auto Range Definition*

  Auto Range Scale = CALENDAR_YEAR
  Duration = 5
  Start Date (Inclusive) = July 15, 2013
  End Date (Inclusive) =

- *Auto Range Replacement Value*

  Date Format = `<see table>`
  Date Delimiter = PERIOD
  Numeric Format = ARABIC_NUMBERS
  Enable Zero Pad = YES
  Month Format = NUMERIC
  Language = ENGLISH
  Year Format = FULL_YEAR

The input date is November 22, 2016. This date falls into the first 5-year calculated range:

- First 5-year range: 7/15/2013 to 12/31/2017

Note that not all Date Format values are shown in the examples

Date Format	Input date = Nov 22, 2013 Output	Notes
DAY_MONTH_YEAR	[15.07.2013, 31.12.2017]	The output appears in dd.mm.yyyy format.

	Input date = Nov 22, 2013	
Date Format	**Output**	**Notes**
YEAR_DAY_MONTH	[2013.15.07, 2017.31.12]	The output appears in yyyy.dd.mm format.
YEAR	2013, 2017	The output appears in yyyy format.
YEAR_MONTH	[2013.07, 2017.12]	The output appears in yyyy.mm format.
DAY_YEAR_MONTH	[15.2013.07, 31.2017.12]	The output appears in dd.yyyy.mm format.

3.6.2.12.8 Date Generalization calculated ranges based on calendar month

Use examples to understand how the software calculates and outputs date ranges based on calendar month and date format.

Example

- *Auto Range Definition*

  Auto Range Scale = CALENDAR_MONTH
  Duration = 1
  Start Date (Inclusive) =
  End Date (Inclusive) =

- *Auto Range Replacement Value*

  Date Format = `<see table>`
  Date Delimiter = SLASH
  Numeric Format = ARABIC_NUMBERS
  Enable Zero Pad = YES
  Month Format = NUMERIC
  Language = ENGLISH
  Year Format = FULL_YEAR

The software uses the input date to calculate ranges because there are no start or end dates specified. The input date is July 4, 2016. All ranges in the output column appear in square brackets because the start and end dates are inclusive. The output values are the same, however, they change based on the listed date format in the first column.

Note that not all date formats are shown in the examples.

Date Format =	Input date = July 4, 2016 Output	Notes
DAY_MONTH_YEAR	[01/07/2016, 31/07/2016]	The output appears in dd/mm/yyyy format.
MONTH_DAY_YEAR	[07/01/2016, 07/31/2016]	The output appears in mm/dd/yyyy format.
YEAR_DAY_MONTH	[2016/01/07, 2016/31/07]	The output appears in yyyy/dd/mm format.
YEAR	NULL	The software issues an error because the *Auto Range Scale* of calendar month and the *Date Format* are incompatible.
MONTH_YEAR	07/2016	The output appears in mm/yyyy format. The output is not a range. The software cannot calculate a range based on the scale, date format, and duration.
MONTH	07	The output appears in mm format. The output is not a range. The software cannot calculate a range because of the scale, date format, and duration.

Example

- *Auto Range Definition*

  Auto Range Scale = CALENDAR_MONTH
  Duration = 3
  Start Date (inclusive) = July 15, 2013
  End Date (inclusive) =

- *Auto Range Replacement Value*

  Date Format = `<see table>`
  Date Delimiter = PERIOD
  Numeric Format = ARABIC_NUMBERS
  Enable Zero Pad = YES
  Month Format = NUMERIC
  Language = ENGLISH
  Year Format = FULL_YEAR

The first software-calculated range begin with a partial month:

- First software-calculated range: 7/15/2013 to 9/30/2013
- Second software-calculated range: 10/1/2013 to 12/31/2013

The input date is July 15, 2013. It falls into the first software-calculated range. The start date and end date are inclusive, so the output ranges appear in square brackets.

Note that not all date formats are shown in the examples.

	Input date = July 15, 2016	
Date Format =	**Output**	**Notes**
MONTH_DAY_YEAR	[07.01.2016, 09.30.2016]	The output appears in mm.dd.yyyy format.
YEAR_MONTH_DAY	[2016.07.01, 2016.09.30]	The output appears in yyyy.mm.dd format.
YEAR	NULL	The software issues an error because the *Auto Range Scale* of calendar month and the *Date Format* of year are incompatible.
YEAR_MONTH	[2016.07, 2016.09]	The output appears in yyyy.mm format.
MONTH_YEAR	[07.2016, 09.2016]	The output appears in mm.yyyy format.
MONTH	[07, 09]	The output appears in mm format.

Example

- *Auto Range Definition*

  Auto Range Scale = CALENDAR_MONTH
  Duration = 3
  Start Date (inclusive) = July 15, 2013
  End Date (inclusive) =

- *Auto Range Replacement Value*

  Date Format = `<see table>`
  Date Delimiter = DASH
  Numeric Format = ARABIC_NUMBERS
  Enable Zero Pad = NO
  Month Format = SHORT_TEXT
  Language = ENGLISH
  Year Format = SHORT_YEAR

The input date is July 21, 2013. It falls within the first software-calculated range of 7/15/2013 to 9/30/2013. Therefore, the first date of the range does not fall at the beginning of the month. The software does not issue a warning.

Date Format =	Input date = July 21, 2013 Output	Notes
DAY_MONTH_YEAR	[15-Jul-13, 30-Sept-13]	The output appears in dd-short text month-yy format.
MONTH_DAY_YEAR	[Jul-15-13, Sept-30-13]	The output appears in short text month-dd-yy format.
YEAR_DAY_MONTH	[13-15-Jul, 13-30-Sept]	The output appears in yy-dd-short text month format.
YEAR	NULL	The software issues an error. The settings in *Auto Range Scale* and the *Date Format* are incompatible.
YEAR_MONTH	[13-Jul, 13-Sept]	The output appears in yy-short text month format.
MONTH	[Jul, Sept]	The output appears in short text month format.

3.6.2.12.9 Date Generalization math equation symbols in calculated ranges

The mathematical equation symbols that the software uses in generated output values indicate if a value is included or not included in the range.

The following table describes the symbols that the software uses in the software-generated ranges for both number generalization and date generalization.

Symbol	Description	Meaning
[	Left square bracket	Low value is inclusive
]	Right square bracket	High value is inclusive
(	Left parenthesis	Low value is exclusive
)	Right parenthesis	High value is exclusive
>= “val”	Right pointing arrow and equal sign	Value is greater than or equal to “val”.
> “val”	Right pointing arrow	Value is greater than “val”.
<= “val”	Left pointing arrow and equal sign	Value is less than or equal to “val”.

Symbol	Description	Meaning
< “val”	Left pointing arrow	Value is less than “val”.
“val”	Exact value	An exact value has no symbols

3.6.3 DQM Microservices

Use the DQM Microservices transform to configure and execute SAP Data Quality Management, microservices for location data services within SAP Data Services.

DQM Microservices transform information

Characteristic	Description
	DQM Microservices transform icon
Use	SAP Data Quality Management, microservices for location data offers cloud-based microservices for address cleansing, geocoding, and reverse geocoding. Embed address cleansing and enrichment services within any business process or application to reap the value of complete and accurate address data. The DQM Microservices transform supports the following DQM microservices: • *addressCleanse service:* Address cleansing corrects, parses, standardizes, validates, and enhances address data, and assigns geocodes. • *reverseGeo service:* Reverse geocoding translates latitude and longitude coordinates into addresses. **! Restriction** The DQM Microservices transform does not support suggestion lists. The transform supports non-suggestion list processing only. For more information about DQM microservices, see the documentation set at https://help.sap.com/viewer/d95546360fea44988eb614718ff7e959/Cloud/en-US. **ℹ Note** The DQM Microservices functionality is available through a productive landscape and a trial landscape.

Characteristic	Description
Content objects	Transform configurations and sample blueprints.

DQM Microservices options [page 940]
Set datastore connection options and the options for the services and configuration for SAP Data Quality Management, microservices for location data.

DQM Microservices transform input fields [page 941]
Input fields for the DQM Microservices transform depend on the service or configuration that you select.

DQM Microservices transform output fields [page 942]
Output fields for the DQM Microservices transform depend on the service or configuration that you select.

Map DQM Microservices fields for the addressCleanse service [page 942]
Use the mapping tables to assist you in mapping DQM Microservices fields to Data Services fields for the addressCleanse service.

Map DQM Microservices fields for the reverseGeo service [page 950]
Use the mapping tables to assist you in mapping DQM Microservices fields to Data Services fields for the reverseGeo service.

Related Information

Connect to SAP Data Quality Management, microservices for location data
Download Data Quality blueprints and other content objects [page 340]

3.6.3.1 DQM Microservices options

Set datastore connection options and the options for the services and configuration for SAP Data Quality Management, microservices for location data.

The Options group includes settings that control how the DQM Microservices transform connects to the datastore and specifies the services and configurations used for SAP Data Quality Management, microservices for location data.

Option	Description
Datastore	Specifies an existing SAP DQM Microservices datastore, which configures connection information that Data Services uses to access DQM microservices. To add a datastore, create a new SAP DQM Microservices datastore using the Datastore Editor. For detailed instructions, see the Supplement for SAP.

Option	Description
Service	Specifies the DQM microservices service to use. *addressCleanse:* corrects, parses, standardizes, validates, and enhances address data, and assigns geocodes. *reverseGeo:* translates latitude and longitude coordinates into addresses. This option becomes available after you specify a datastore.
Configuration	Contains the available configurations for the selected service in the default configuration of the specified datastore. In DQM microservices, you can define configurations for the addressCleanse service. The configurations tell the addressCleanse service how to map or route addresses for processing. (For example, French addresses are routed for French-specific processing, German addresses are routed for German-specific processing, and so on.) The configurations also contain the addressCleanse service options. If the Service option is set to reverseGeo, None is the only available value. If the Service option is set to addressCleanse, all available DQM microservices configurations for the specified datastore are displayed. This option becomes available after you specify a service. For more information about creating configurations, see the DQM microservices documentation at https://help.sap.com/viewer/d95546360fea44988eb614718ff7e959/Cloud/en-US.
Settings	Open the Settings window, which displays the settings retrieved from DQM microservices. The options displayed are different for each type of supported service. For descriptions of the available settings for each service, see the DQM microservices documentation at https://help.sap.com/viewer/d95546360fea44988eb614718ff7e959/Cloud/en-US. This option becomes available if the Configuration option is set to None. If a configuration has been selected, the option is not available.

3.6.3.2 DQM Microservices transform input fields

Input fields for the DQM Microservices transform depend on the service or configuration that you select.

The transform displays input fields in the DQM Microservices transform based on the fields available in the service or configuration that you select for SAP Data Quality Management, microservices for location data. Each service or configuration can have a different set of input fields. The input fields are displayed after you specify the service and configuration in the Options tab.

If you change a service or configuration, the available input fields change. If you map input fields and then select a different service or configuration, remap the input fields.

For descriptions of the available fields, see the DQM microservices documentation at https://help.sap.com/viewer/d95546360fea44988eb614718ff7e959/Cloud/en-US.

3.6.3.3 DQM Microservices transform output fields

Output fields for the DQM Microservices transform depend on the service or configuration that you select.

The transform displays output fields DQM Microservices transform based on the fields available in the service or configuration that you select for SAP Data Quality Management, microservices for location data. Each service or configuration can have a different set of output fields. The transform displays the output fields after you specify the service and configuration in the *Options* tab.

If you change a service or configuration, the available output fields change. If you map output fields and then select a different service or configuration, remap the output fields.

For descriptions of the available fields, see the DQM microservices documentation at https://help.sap.com/viewer/d95546360fea44988eb614718ff7e959/Cloud/en-US.

3.6.3.4 Map DQM Microservices fields for the addressCleanse service

Use the mapping tables to assist you in mapping DQM Microservices fields to Data Services fields for the addressCleanse service.

To set up mapping for the addressCleanse service, you map SAP Data Quality Management, microservices for location data fields to Global Address Cleanse and Geocoder transform fields. The following list shows the order of mapping:

- Map SAP Data Quality Management, microservices for location data input field attributes to the Global Address Cleanse input fields
- Map SAP Data Quality Management, microservices for location data generated attributes to the Global Address Cleanse output fields
- Map SAP Data Quality Management, microservices for location data generated attributes to the Geocoder output fields

If you use both a Global Address Cleanse or Geocoder transform and a DQM Microservices transform to process your data in the same data flow, add a Query transform downstream from one of the transforms to harmonize the transform output schemas. Then add a Merge transform to combine the data sets into a single output data set. For more information, see the *Supplement for SAP*.

Map data input attributes to Global Address Cleanse input fields [page 943]
Map SAP Data Quality Management, microservices for location data input field attributes to the Global Address Cleanse input fields for use with the addressCleanse service.

Map generated attributes to Global Address Cleanse output fields [page 944]
Map SAP Data Quality Management, microservices for location data generated attributes to the Global Address Cleanse output fields for use with the addressCleanse service.

Map generated attributes to Geocoder output fields [page 949]
Map SAP Data Quality Management, microservices for location data generated attributes to the Geocoder output fields for the addressCleanse service.

3.6.3.4.1 Map data input attributes to Global Address Cleanse input fields

Map SAP Data Quality Management, microservices for location data input field attributes to the Global Address Cleanse input fields for use with the addressCleanse service.

Data Quality Management, microservices for location data input attribute	Global Address Cleanse input field
building	NW_Building
country	Country or NW_Country
floor	NW_Floor_Num
house_num	NW_House_Num1
house_num2	NW_House_Num2
locality	Locality1 or NW_City1
locality2	Locality2 or NW_Home_City
locality3	Locality3 or NW_City2
location	NW_Location
mixed	Multiline1
mixed10	Multiline10
mixed11	Multiline11
mixed2	Multiline2
mixed3	Multiline3
mixed4	Multiline4
mixed5	Multiline5
mixed6	Multiline6
mixed7	Multiline7
mixed8	Multiline8
mixed9	Multiline9
po_box	NW_PO_Box
po_box_country	NW_PO_Box_Country
po_box_locality	NW_PO_Box_City
po_box_postcode	NW_PO_Box_Postcode
po_box_region	NW_PO_Box_Region
postcode	Postcode or NW_Postcode
region	Region1 or NW_Region

Data Quality Management, microservices for location data input attribute	Global Address Cleanse input field
region2	Region2
roomnumber	NW_Room_Num
street	NW_Street
street_suppl	NW_Str_Suppl1
street_suppl2	NW_Str_Suppl2
street_suppl3	NW_Str_Suppl3
suggestionReply	Suggestion_Reply1

3.6.3.4.2 Map generated attributes to Global Address Cleanse output fields

Map SAP Data Quality Management, microservices for location data generated attributes to the Global Address Cleanse output fields for use with the addressCleanse service.

Data Quality Management, microservices for location data generated attribute	Global Address Cleanse output field
addr_address_rem	Address_Line_Remainder1.Best.Component.Delivery
addr_address_rem2	Address_Line_Remainder2.Best.Component.Delivery
addr_address_rem3	Address_Line_Remainder3.Best.Component.Delivery
addr_address_rem4	Address_Line_Remainder4.Best.Component.Delivery
addr_asmt_info	Assignment_Info.Best.Component.Delivery
addr_asmt_level	Assignment_Level.None.Assignment_Info.None
addr_asmt_type	Assignment_Type.Best.Component.Delivery
addr_change_sig	Change_Significance.None.Assignment_Info.None
addr_extra	Extra1.Best.Component.Delivery
addr_extra2	Extra2.Best.Component.Delivery
addr_extra3	Extra3.Best.Component.Delivery
addr_extra4	Extra4.Best.Component.Delivery
addr_info_code	Info_Code.None.Assignment_Info.None
addr_language	Language.Best.Component.Delivery
addr_po_box_asmt_info	NW_PO_Box_Assignment_Info.Best.Component.Delivery

Data Quality Management, microservices for location data generated attribute	Global Address Cleanse output field
addr_po_box_asmt_level	NW_PO_Box_Assignment_Level.None.Assignment_Info.None
addr_po_box_asmt_type	NW_PO_Box_Assignment_Type.Best.Component.Delivery
addr_po_box_change_sig	NW_PO_Box_Change_Significance.None.Assignment_Info.None
addr_po_box_info_code	NW_PO_Box_Info_Code.None.Assignment_Info.None
addr_remainder_extra_pmb_full	Remainder_Extra_PMB_Full.Best.Component.Delivery
addr_remainder_full	Remainder_Full.Best.Component.Delivery
addr_sugg_count	Count.None.Suggestion.None
addr_sugg_error	Error.None.Suggestion.None
addr_sugg_list	Suggestion_List.None.Suggestion.None
addr_sugg_status	Status.None.Suggestion.None
sap_formatted_po_box_postcode	NW_PO_Box_NW_Formatted_Postcode.Best.Component.Delivery
sap_formatted_postcode	NW_Formatted_Postcode.Best.Component.Delivery
sap_in_fmt_po_box_postcode	NW_PO_Box_NW_Postcode_In_Supported_Format.Best.Component.Delivery
sap_in_fmt_postcode	NW_Postcode_In_Supported_Format.Best.Component.Delivery
std_addr_additional_info	Additional_Info1.Best.Component.Delivery
std_addr_additional_info2	Additional_Info2.Best.Component.Delivery
std_addr_additional_info3	Additional_Info3.Best.Component.Delivery
std_addr_additional_info4	Additional_Info4.Best.Component.Delivery
std_addr_additional_info5	Additional_Info5.Best.Component.Delivery
std_addr_additional_info6	Additional_Info6.Best.Component.Delivery
std_addr_addr_deldual	Primary_Secondary_Addr_Delivery_Dual.Best.Component.Delivery
std_addr_addr_deldual_bldg	Building_Primary_Secondary_Addr_Delivery_Dual.Best.Component.Delivery
std_addr_address_delivery	Primary_Secondary_Address.Best.Component.Delivery
std_addr_address_dual	Primary_Secondary_Address.Best.Component.Dual
std_addr_area_name	Area_Name1.Best.Component.Delivery
std_addr_block_desc	Block_Description.Best.Component.Delivery
std_addr_block_full	Block_Full.Best.Component.Delivery
std_addr_block_number	Block_Number.Best.Component.Delivery
std_addr_building_name	Building_Name1.Best.Component.Delivery
std_addr_building_name1_2	Building_Name1_2.Best.Component.Delivery
std_addr_building_name2	Building_Name2.Best.Component.Delivery

Data Quality Management, microservices for location data generated attribute	Global Address Cleanse output field
std_addr_country_2char	ISO_Country_Code_2char.Best.Component.Delivery
std_addr_country_3char	ISO_Country_Code_3char.Best.Component.Delivery
std_addr_country_3digit	ISO_Country_Code_3Digit.Best.Component.Delivery
std_addr_country_name	Country_Name.Best.Component.Delivery
std_addr_delinst_full	Delivery_Installation_Full.Best.Component.Delivery
std_addr_firm	Firm.Best.Component.Delivery
std_addr_floor_desc	Floor_Description.Best.Component.Delivery
std_addr_floor_full	Floor_Full.Best.Component.Delivery
std_addr_floor_number	Floor_Number.Best.Component.Delivery
std_addr_floor_qual	Floor_Qualifier.Best.Component.Delivery
std_addr_full_address	Full_Address.Best.Component.Delivery
std_addr_lastline	Lastline.Best.Component.Delivery
std_addr_locality	Locality1_Name.Best.Component.Delivery
std_addr_locality_desc	Locality1_Description.Best.Component.Delivery
std_addr_locality_full	Locality1_Full.Best.Component.Delivery
std_addr_locality1_2	Locality1_2_Name.Best.Component.Delivery
std_addr_locality1_2_full	Locality1_2_Full.Best.Component.Delivery
std_addr_locality1_4	Locality1_4_Name.Best.Component.Delivery
std_addr_locality1_4_full	Locality1_4_Full.Best.Component.Delivery
std_addr_locality2	Locality2_Name.Best.Component.Delivery
std_addr_locality2_4	Locality2_4_Name.Best.Component.Delivery
std_addr_locality2_4_full	Locality2_4_Full.Best.Component.Delivery
std_addr_locality2_desc	Locality2_Description.Best.Component.Delivery
std_addr_locality2_full	Locality2_Full.Best.Component.Delivery
std_addr_locality3	Locality3_Name.Best.Component.Delivery
std_addr_locality3_4	Locality3_4_Name.Best.Component.Delivery
std_addr_locality3_4_full	Locality3_4_Full.Best.Component.Delivery
std_addr_locality3_desc	Locality3_Description.Best.Component.Delivery
std_addr_locality3_full	Locality3_Full.Best.Component.Delivery
std_addr_locality4	Locality4_Name.Best.Component.Delivery
std_addr_locality4_desc	Locality4_Description.Best.Component.Delivery
std_addr_locality4_full	Locality4_Full.Best.Component.Delivery
std_addr_pmb_full	PMB_Full.Best.Component.Delivery

Data Quality Management, microservices for location data generated attribute	Global Address Cleanse output field
std_addr_pname_secaddr	PName_Secondary_Addr.Best.Component.Delivery
std_addr_po_box_country_2char	NW_PO_Box_ISO_Country_Code_2char.Best.Component.Delivery
std_addr_po_box_delinst_full	NW_PO_Box_Delivery_Installation_Full.Best.Component.Delivery
std_addr_po_box_full	NW_PO_Box_Primary_Address.Best.Component.Delivery
std_addr_po_box_locality_full	NW_PO_Box_Locality1_Full.Best.Component.Delivery
std_addr_po_box_number	NW_PO_Box_Primary_Number.Best.Component.Delivery
std_addr_po_box_postcode_full	NW_PO_Box_Postcode_Full.Best.Component.Delivery
std_addr_po_box_region	NW_PO_Box_Region1.Best.Component.Delivery
std_addr_po_box_region_code	NW_PO_Box_Region1_Code.Best.Component.Delivery
std_addr_po_box_region_full	NW_PO_Box_Region1_Full.Best.Component.Delivery
std_addr_po_box_region2	NW_PO_Box_Region2.Best.Component.Delivery
std_addr_po_box_region2_code	NW_PO_Box_Region2_Code.Best.Component.Delivery
std_addr_po_box_region2_full	NW_PO_Box_Region2_Full.Best.Component.Delivery
std_addr_point_of_ref	Point_Of_Reference1.Best.Component.Delivery
std_addr_point_of_ref1_2	Point_Of_Reference1_2.Best.Component.Delivery
std_addr_point_of_ref2	Point_Of_Reference2.Best.Component.Delivery
std_addr_postcode_full	Postcode_Full.Best.Component.Delivery
std_addr_postcode1	Postcode1.Best.Component.Delivery
std_addr_postcode2	Postcode2.Best.Component.Delivery
std_addr_prim_address	Primary_Address.Best.Component.Delivery
std_addr_prim_name	Primary_Name1.Best.Component.Delivery
std_addr_prim_name_full	Primary_Name_Full1.Best.Component.Delivery
std_addr_prim_name1_2	Primary_Name_Full1_2.Best.Component.Delivery
std_addr_prim_name1_4	Primary_Name_Full1_4.Best.Component.Delivery
std_addr_prim_name2	Primary_Name2.Best.Component.Delivery
std_addr_prim_name2_full	Primary_Name_Full2.Best.Component.Delivery
std_addr_prim_name3	Primary_Name3.Best.Component.Delivery
std_addr_prim_name3_4	Primary_Name_Full3_4.Best.Component.Delivery
std_addr_prim_name3_full	Primary_Name_Full3.Best.Component.Delivery
std_addr_prim_name4	Primary_Name4.Best.Component.Delivery
std_addr_prim_name4_full	Primary_Name_Full4.Best.Component.Delivery

Data Quality Management, microservices for location data generated attribute	Global Address Cleanse output field
std_addr_prim_number	Primary_Number.Best.Component.Delivery
std_addr_prim_number_desc	Primary_Number_Description.Best.Component.Delivery
std_addr_prim_number_extra	Primary_Number_Extra.Best.Component.Delivery
std_addr_prim_number_full	Primary_Number_Full.Best.Component.Delivery
std_addr_prim_postfix	Primary_Postfix1.Best.Component.Delivery
std_addr_prim_prefix	Primary_Prefix1.Best.Component.Delivery
std_addr_prim_type	Primary_Type1.Best.Component.Delivery
std_addr_prim_type2	Primary_Type2.Best.Component.Delivery
std_addr_prim_type3	Primary_Type3.Best.Component.Delivery
std_addr_prim_type4	Primary_Type4.Best.Component.Delivery
std_addr_primaddr_deldual	Primary_Address_Delivery_Dual.Best.Component.Delivery
std_addr_primaddr_deld-ual_bldg	Building_Primary_Addr_Delivery_Dual.Best.Component.Delivery
std_addr_region	Region1.Best.Component.Delivery
std_addr_region_code	Region1_Code.Best.Component.Delivery
std_addr_region_full	Region1_Full.Best.Component.Delivery
std_addr_region1_2	Region1_2_Name.Best.Component.Delivery
std_addr_region1_2_full	Region1_2_Full.Best.Component.Delivery
std_addr_region2	Region2.Best.Component.Delivery
std_addr_region2_code	Region2_Code.Best.Component.Delivery
std_addr_room_full	Room_Full.Best.Component.Delivery
std_addr_room_number	Room_Number.Best.Component.Delivery
std_addr_sec_address	Secondary_Address.Best.Component.Delivery
std_addr_secaddr_no_floor	Secondary_Address_No_Floor.Best.Component.Delivery
std_addr_se-caddr_no_floor_room	Secondary_Address_No_Floor_No_Room.Best.Component.Delivery
std_addr_secaddr_no_room	Secondary_Address_No_Room.Best.Component.Delivery
std_addr_single_address	Single_Address.Best.Component.Delivery
std_addr_stairwell_desc	Stairwell_Description.Best.Component.Delivery
std_addr_stairwell_full	Stairwell_Full.Best.Component.Delivery
std_addr_stairwell_name	Stairwell_Name.Best.Component.Delivery
std_addr_unit_desc	Unit_Description.Best.Component.Delivery
std_addr_unit_full	Unit_Full.Best.Component.Delivery

Data Quality Management, microservices for location data generated attribute	Global Address Cleanse output field
std_addr_unit_number	Unit_Number.Best.Component.Delivery
std_addr_unit_qual	Unit_Qualifier.Best.Component.Delivery
std_addr_wing_desc	Wing_Description.Best.Component.Delivery
std_addr_wing_full	Wing_Full.Best.Component.Delivery
std_addr_wing_name	Wing_Name.Best.Component.Delivery
sugg_addr_address_delivery (in addr_sugg_list)	Sugg_Full_Addressline.None.Suggestion.None
sugg_addr_lastline (in addr_sugg_list)	Sugg_Full_Lastline.None.Suggestion.None
sugg_addr_prim_number_high (in addr_sugg_list)	Primary_Number_High.None.Suggestion.None
sugg_addr_prim_number_low (in addr_sugg_list)	Primary_Number_Low.None.Suggestion.None
sugg_addr_prim_side_indica-tor (in addr_sugg_list)	Primary_Side_Indicator.None.Suggestion.None
sugg_addr_sec_side_indicator (in addr_sugg_list)	Secondary_Side_Indicator.None.Suggestion.None
sugg_addr_selection (in addr_sugg_list)	Selection.None.Suggestion.None
sugg_addr_single_address (in addr_sugg_list)	Sugg_Single_Address.None.Suggestion.None
sugg_addr_unit_number_high (in addr_sugg_list)	Unit_Number_High.None.Suggestion.None
sugg_addr_unit_number_low (in addr_sugg_list)	Unit_Number_Low.None.Suggestion.None

3.6.3.4.3 Map generated attributes to Geocoder output fields

Map SAP Data Quality Management, microservices for location data generated attributes to the Geocoder output fields for the addressCleanse service.

Data Quality Management, microservices for location data generated attribute	Geocoder output field
addr_latitude	Latitude

Data Quality Management, microservices for location data generated attribute	Geocoder output field
addr_longitude	Longitude
geo_info_code	Info_Code
geo_asmt_level	Assignment_Level

3.6.3.5 Map DQM Microservices fields for the reverseGeo service

Use the mapping tables to assist you in mapping DQM Microservices fields to Data Services fields for the reverseGeo service.

To set up mapping for the reverseGeo service, you map SAP Data Quality Management, microservices for location data fields to Geocoder transform fields. The following list shows the order of mapping:

- Map SAP Data Quality Management, microservices for location data input attributes to the Geocoder input fields
- Map SAP Data Quality Management, microservices for location data generated attributes to the Geocoder output fields

Map input attributes to Geocoder input fields [page 950]
Map SAP Data Quality Management, microservices for location data input attributes to the Geocoder input fields for the reverseGeo service.

Map generated attributes to Geocoder output fields [page 951]
Map SAP Data Quality Management, microservices for location data generated attributes to the Geocoder output fields for the reverseGeo service.

3.6.3.5.1 Map input attributes to Geocoder input fields

Map SAP Data Quality Management, microservices for location data input attributes to the Geocoder input fields for the reverseGeo service.

Data Quality Management, microservices for location data input attribute	Geocoder input field
latitude	Latitude
longitude	Longitude

3.6.3.5.2 Map generated attributes to Geocoder output fields

Map SAP Data Quality Management, microservices for location data generated attributes to the Geocoder output fields for the reverseGeo service.

Data Quality Management,microservices for location data generated attribute	Geocoder output field
geo_info_code	Info_Code
geo_search_results	Result_List
geo_search_results_count	Result_List_Count
search_geo_addr_latitude (in geo_search_results)	Latitude
search_geo_addr_longitude (in geo_search_results)	Longitude
search_geo_address_delivery (in geo_search_results)	Address_Line
search_geo_asmt_level (in geo_search_results)	Assignment_Level
search_geo_country_2char (in geo_search_results)	Country_Code
search_geo_distance (in geo_search_results)	Distance
search_geo_group_id (in geo_search_results)	Group_Number
search_geo_group_master (in geo_search_results)	Group_Rand
search_geo_locality (in geo_search_results)	Locality1
search_geo_locality2 (in geo_search_results)	Locality2
search_geo_poi_name (in geo_search_results)	POI_Name
search_geo_poi_type (in geo_search_results)	POI_Type
search_geo_population_class (in geo_search_results)	Population_Class_Locality1
search_geo_postcode_full (in geo_search_results)	Postcode
search_geo_region (in geo_search_results)	Region1
search_geo_side_of_street (in geo_search_results)	Side_Of_Primary_Address

3.6.4 Map_Operation

The Map_Operation transform enables you to modify input data using mapping expressions and operation codes.

Map_Operation transform information

Characteristic	Description
	Map_Operation icon

Characteristic	Description
Use	Modifies data based on mapping expressions and current operation codes. The operation codes can be converted between data manipulation operations. If you configure both mapping expressions and operation codes, Data Services runs mapping expressions before any row type conversions. Mapping expressions are optional. If you don't have mapping expressions configured, Data Services performs operation codes only. When you write map expressions per column and per row type (INSERT/UPDATE/DELETE) you can: • change the value of data for a column. • execute different expressions on a column, based on its input row type. • use the before_image function to access the before image value of an UPDATE row. **i Note** Use mapping expressions on top-level schemas only. Mapping expressions do not work on nested schemas. The Map_Operation transform can also change operation codes on data sets to produce the desired output. **Example** A row in your input data set was updated in some previous operation in the data flow. To preserve the row on output, use the Map_Operation transform to map an UPDATE operation to an INSERT operation. The result of converting UPDATE rows into INSERT rows is that the row is preserved in the target. Data Services can push Map_Operation transforms to the source database for efficient processing.
Content objects	Sample transform.

Related Information

Effective_Date transform [page 267]
real [page 1055]
before_image [page 1087]

3.6.4.1 Map_Operation transform options

Set options in the Map_Operations transform to create expressions and to change the row type on output.

Map_Operations transform option descriptions

Option	Description
Mapping tab	
Insert/Normal mapping	Displays an expression that performs an insert operation. Use the Smart Editor or function wizard to create a SQL statement to perform the insert operation.
Update mapping	Displays an expression that performs an update operation Use the Smart Editor or function wizard to create a SQL statement to perform the update operation.
Delete mapping	Displays an expression that performs a delete operation. Use the Smart Editor or function wizard to create a SQL statement to perform the delete operation.
Functions button	Opens a function wizard to create an expression.
Ellipses button (...)	Opens the Smart Editor to create an expression.
Map Operation tab	
Input row type	Specifies the input mapping row type. You cannot change the values in this column.
Output row type	Specifies the mapping for the output row type. You can change the values in this column. For example, to include a Delete row on output, change the Delete input row type to Insert.

Operation codes

Operation code	Description
NORMAL	Creates a new row in the target. All rows in a data set are flagged as NORMAL when they are extracted from a source. If a row is flagged as NORMAL when loaded into a target, the transform inserts the row as a new row in the target.

Operation code	Description
INSERT	Creates a new row in the target. Rows can be flagged as INSERT by transforms in the data flow to indicate that a change occurred in a data set as compared with an earlier image of the same data set. The change is recorded in the target separately from the existing data.
DELETE	Is ignored by the target. Rows flagged as DELETE are not loaded. Rows can be flagged as DELETE only by the Map_Operation transform.
UPDATE	Overwrites an existing row in the target. Rows can be flagged as UPDATE by transforms in the data flow to indicate that a change occurred in a data set as compared with an earlier image of the same data set. The change is recorded in the target in the same row as the existing data.

3.6.4.2 Map_Operation transform data inputs and outputs

Data that you input to the Map_Operation transform contains rows that are flagged with operation codes.

After processing, the data output from the Map_Operation transform contains rows flagged as specified by the mapping operation.

Data inputs

Input data for the Map_Operation transform should contain rows flagged with operation codes. Input data can also be hierarchical.

> **i Note**
>
> If you use columns with the data type of **real** to compare data in the Map_Operation transform, be aware that comparison results are unpredictable.

Rows in your input data set can contain any of the following operation codes:

- NORMAL
- INSERT
- DELETE
- UPDATE

Map the input operation codes to the following operation codes:

- NORMAL
- INSERT
- DELETE
- UPDATE

Data outputs

The transform outputs data with rows flagged as specified by your mapping operations.

In addition, you can assign the DISCARD option. The transform does not pass rows flagged as DISCARD to the output.

By default, every input operation type maps to itself. For each specified mapping, every row in the input data set that matches the input mapping operation is converted to the specified output operation.

Example: Showing a change in cost

You have a target table that contains a row with a `<Cost>` field value of 9,000, and a `<cost_change>` field value of 0. Later on, the input data changes so that the row now contains a `<Cost>` field value of 1,000.

You want to find out what the change in the `<Cost>` field value is after the input data changed. To determine the change in the `<Cost>` field value:

1. Set up a Table_Comparison transform. The Table_Comparison transform determines the row type, such as INSERT or UPDATE.
2. Set up a Map_Operation transform with an expression in the *Mapping* tab. Create an expression for each column and each row type.
   - For the INSERT row type, fill the `<cost_change>` field with 0.
   - For the UPDATE row type, create an expression as follows: `T_PRODUCT1.COST - before_image(T_PRODUCT1.COST)`
     The expression subtracts the initial (before_image) value from the new value in the `<Cost>` field. The result is a negative value in the `<cost_change>` field of -8,000.
   - For the DELETE row type, set the cost_change field to:
     `0 - T_PRODUCT1.COST`

3. Open the *Map Operation* tab to see the following mappings:

Input row type	Output row type
normal	normal
update	update
insert	insert
delete	update

Example: Changing a field value and a row type

You want to change a field value and a row type

The `<VALID>` field in your data can have the following values:

- 1 = valid. The product is in production.
- 0 = invalid. The product is not in production

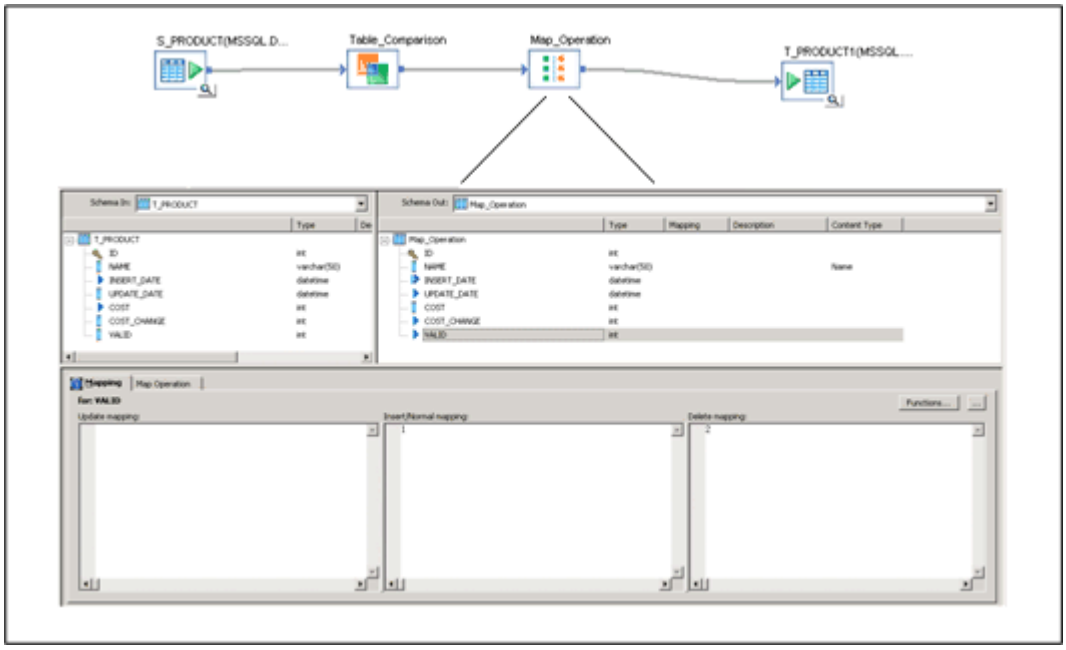
You do not want to delete the product from the target even though it has been deleted from the source. Instead, you want to update the `<VALID>` field value in the target to 0.

> **i Note**
>
> When configuring both mapping expressions and operation codes, Data Services runs mapping expressions before any row type conversions.

You define two expressions:

- The expression changes the `<VALID>` field value to 0.
- The expression converts the DELETE row type to UPDATE.

The following screen capture shows the data flow and the *Mapping* tab of the Map_Operation transform.



The *Map Operation* tab has the following mapping:

Input row type	Output row type
normal	normal
update	update
insert	insert
delete	update

Related Information

before_image [page 1087]
Table_Comparison transform [page 318]

3.6.5 Merge transform

Use the Merge transform to combine multiple input sources to produce a single output data set that has the same schema as the input data sets.

Merge transform information

Characteristic	Description
	Merge transform icon
Use	Combines two or more data sets that have the same schema and outputs to one target that also has the same schema. The transform outputs every row from each source to the target. Special situations: • Hierarchical data sets: All names and data types match at every level. • Nested data: Passed to output without change.
Content objects	Sample transform.

3.6.5.1 Merge transform options

The Merge transform consists of input and output schema, with no additional options to configure.

The Merge transform consists of a *Schema In* and a *Schema Out* pane. The transform does not allow deletions, additions, or edits. You can view the field properties dialog if applicable, but you cannot edit the properties.

The *Schema In* pane contains all of the data sources listed with the input fields. Each source contains the same input fields and schema. The *Schema Out* pane lists one target file that contains a single list of the same fields as the input sources, and the same schema.

3.6.5.2 Merge transform data inputs and outputs

The Merge transform takes two or more input data sets with identical schemas and outputs to one target data set that has the same schema as the input data sets.

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.

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.

Apply other operations such as DISTINCT in a query following the Merge transform in the data flow.

3.6.6 Query transform

The Query transform retrieves a data set that satisfies conditions that you specify. This transform is similar to a `SQL SELECT` statement.

Query transform information

Characteristic	Description
	Query transform icon

Characteristic	Description
Use	The Query transform performs the following operations: • Choose (filter) the data to extract from sources • Join data from multiple sources • Map columns from input to output schemas • Perform transformations and functions on the data • Perform data nesting and unnesting • Add new columns, nested schemas, and function results to the output schema • Assign primary keys to output columns
Content objects	Sample transform.

Related Information

Smart Editor and the function wizard [page 1061]

3.6.6.1 Query transform editor

Use the Query editor to perform transform set up tasks including field mapping and manipulation, and creating expressions.

The Query Editor contains the following areas:

- *Schema In* pane in the upper left
- *Schema Out* pane in the upper right
- Options pane in the lower area

The *Schema In* and *Schema Out* panes can contain columns, nested schemas, and functions (output only).

3.6.6.2 Query transform options pane

Use the tabs in the options pane to create SQL statements to manipulate input data and to output data that meets the specific expression requirements.

The options area of the Query editor contains several tabs to enter information about the data you want retrieved. Specifying information on these tabs is similar to specifying a SQL SELECT statement. When you work with the tabs in the Query editor, the software shows the tabs that contain entries with 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 more complex mappings, complete the options in the appropriate tabs.

> **i Note**
>
> Use the *SELECT* through *ORDER BY* tabs to specify additional constraints for the current schema, similar to SQL SELECT statement clauses.

Query Editor tab descriptions

Tab	Description
Mapping	Specifies how the software maps or derives the selected output column. Contains a *Schema Remapping* button to replace an obsolete schema with a correct schema.
SELECT	Specifies the distinct rows to output, and optionally 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 transform displays the resulting SQL FROM clause in the tab.
WHERE	Specifies conditions that determine the rows to output. Use the *Functions*, *Domains*, and *Smart Editor* buttons to build expressions.
GROUP BY	Specifies how the transform combines the output rows.
ORDER BY	Specifies how the transform sorts the output rows.
Advanced	Creates separate data flows so the transform processes resource-intensive query clauses separately.
Find	Uses search criteria to find specific words, terms, or other objects.

3.6.6.3 Query transform schema panes

Use the *Schema In* and *Schema Out* panes to map input fields to output, to create new schemas, to unnest nested data, and much more.

The Schema In pane lists your input data sets with the input fields and schema information. Add a target table with an existing schema, and the software automatically generates the Schema Out fields. Use the many techniques to rearrange output schema content, such as the following:

- Drag a column from the *Schema In* to the *Schema Out* pane
- Move a column up or down in the column order
- Change a column type, size, and other attributes

For information about working with nested schemas, and other techniques for using the Query Schema panes, see the *Designer Guide*.

3.6.6.4 Query transform data inputs and outputs

A Query transform takes data from upstream objects in the data flow and sends data to downstream objects in the data flow.

Data that is input to a Query transform can be from one source, or many sources. You control the data that is output by using the various features of the Query transform. You view and manipulate the data using the *Schema In* and *Schema Out* panes in the Query transform.

> **Note**
>
> If you aren't familiar with the many things you can do with your data using the Query transform, see the Data services *Tutorial* for more information.

> **Note**
>
> Use caution when using columns of data type `real` in this transform. Comparison results are unpredictable for the `real` data type.

3.6.6.5 Query transform Mapping tab

Use the options in the *Mapping* tab to specify how the selected output column is derived (or mapped). You can specify any valid expression.

Mapping expressions contain table columns and functions:

- Enter input column names or drag columns from the *Schema In* pane 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**
>
> The transform doesn't allow you to add comments to a mapping clause in a Query transform. For example, the transform does not support the following syntax in the *Mapping* tab:

```
table.column # comment
```

If you add comments, the job does not run and you cannot successfully export it. If you add comments, use the object description or workspace annotation.

Automatic remapping

The transform automatically remaps the input fields if you edit a data flow and add a transform before the Query transform.

> **Example**
>
> You set up a data flow with a source object connected to a Query transform. You add a target object, configure the Query transform and save the data flow. The Query transform contains the input mapping based on the source file. Then you add a Validation transform between the source and the Query transform.

> You configure the Validation transform to ensure that only data with a specific format flows into the Query transform.

Data Services checks the existing top-level mappings to determine if it needs to remap the Schemas in the Query transform.

- If the existing top-level mapping contains a column that exists in the new schema, but the source is different, the software changes the source name to the new source name. The software checks for the obsolete source name and replaces it in all applicable tabs in the Query editor.
- If the existing top-level mapping contains a column that does not exist in the new schema, and the new schema has not changed, you manually remove the column from the schema.

Schema Remapping

Use the Schema Remapping tool when Data Services does not automatically remap the input schema. For example:

- When you connect a new source to the Query transform before you disconnect the old source. To remap the schema, click *Schema Remapping* in the *Mapping* tab. The software then updates 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. To remap the schema, click *Schema Remapping* in the *Mapping* tab to update the input schema name in the *Mapping* tab.

3.6.6.6 Query transform SELECT tab

Use the *SELECT* tab to output distinct rows, and discard duplicates.

The *SELECT* tab contains one option. Select *Distinct rows* to output only the rows that are not duplicate rows. The software discards any duplicate rows. Selecting this option is similar to specifying a `SELECT DISTINCT` SQL statement.

3.6.6.7 Query transform FROM tab

Use the *FROM* tab to construct joins between input schemas.

Create left outer joins, create inner joins, set join rank, set join conditions to use in the output schema, and select the cache type. The specified information is similar to using the `FROM` clause in a SQL SELECT statement.

The *FROM* tab is divided into three areas:

- The uppermost area contains the input schema names and other settings for join construction. Select the input schemas for the join. The transform automatically populates the Input schemas column with the schemas in the Schema In pane.
- The middle area contains options to specify the type of join and to construct join conditions.
- The lower area displays the `FROM` clause that results from the join you define. The software automatically updates the clause as you construct joins. This area is read-only, but can be copied to the clipboard.

The middle area contains a *Join pairs* table where you define the join type and construct join conditions. The transform provides tools to help you construct the join condition. The following table describes the columns displayed in the *Join pairs* table.

Column name	Description
Left	Specifies the source for the left portion of the join. Select the input schema from the dropdown list of available schemas. If you create more than one join, the transform designates the input schema for the left join as the result of the previous join.
Join Type	Specifies the type of join to create. The type of join. Select *Left outer join* or *Inner join*. If a table is not explicitly joined to another table, then it is cross-joined (Cartesian Product) to the result of the final join defined by the join pairs.
Right	Specifies the source for the right portion of a join. The dropdown list contains the input schemas that have not been used as the left portion or in any other defined join.
Smart Editor icon ...	Opens the Smart Editor. The ellipses icon appears in the cell next to the join you are constructing. When you have the first three columns populated, the ellipses acts as a button to open the Smart Editor. The Smart Editor is a tool to help you construct a join condition.
Propose join icon	Proposes a join expression based on the options you choose in the first three columns. The *Propose join* icon appears in the cell next to the *Smart Editor* icon. If you change the left source, right source, or join type, the software automatically updates the SQL clause .
Join condition	Specifies a join condition. Create a join condition for each join pair. Where possible, the transform automatically suggests a join condition based on the input schemas of the join pair. To edit the join condition, enter the join condition into the field or click the *Smart Editor* icon and use the Smart Editor to construct a join.

> **i Note**
>
> If your expression contains varchar comparisons, the software ignores trailing blanks in the data. For Oracle data, use the `RTRIM` or `RPAD` functions if the number of trailing blanks might differ on either side of the comparison.

3.6.6.8 Query transform 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 Join* button generates possible join conditions. The following table lists the applicable conditions.

Condition	Example
Data set filters	`TABLE1.EMPNO > 1000`
Multiple conditions using logical operators	`TABLE1.EMPNO > 1000` `OR TABLE2.EMPNO < 9000`
Join conditions for inner joins only **i Note** Although it is technically possible to specify inner join conditions in the *WHERE* tab, we recommend that you specify inner join conditions in the *FROM* tab.	`TABLE1.EMPNO = TABLE2.EMPNO`

You can specify any valid expression in your condition. 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 specifying the clause.

> **i Note**
>
> If your expression contains varchar comparisons, Data Services ignores trailing blanks in the data. For Oracle data, use the `RTRIM` or `RPAD` functions if the number of trailing blanks might differ on either side of the comparison.

3.6.6.9 Query transform 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, group sales order records by order date to find the total sales ordered on a particular date. This tab is similar to the `GROUP BY` clause in a SQL Select statement.

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 transform adds the column to the bottom of the list.

The transform uses the first column listed for primary grouping, the second column listed for secondary grouping, and so on. 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.

When you specify a Group By list, use all columns in the output schema in the list. However, if you do not include all columns, map columns any remaining columns to an aggregate function, such as `avg`, `count`, `max`, `min`, or `sumColumns`

To group columns by complex expressions instead of by specific column values, create a new Query transform. Place the transform before the current Query transform in the data flow. In the new Query transform, produce a single column that contains the grouping expression. The new column appears in the Schema In pane. Specify the new column in the *GROUP BY* tab.

3.6.6.10 Query transform ORDER BY tab

Use the *ORDER BY* tab to specify the columns for sorting 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 transform adds the column to the bottom of the list.

The transform uses the first column in the list for primary sorting, the second column in the list for secondary sorting, and so on. 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 *Descending* from the adjacent dropdown list.

3.6.6.11 Query transform Advanced tab

Use the *Advanced* tab to set up separate processes.

Each process forms a data flow. Set up separate processes for the resource-intensive queries that you configured in the other tabs. The Advanced tab contains options for the following resource-intensive query clauses:

- DISTINCT
- GROUP BY
- JOIN
- ORDER BY

Related Information

Smart Editor and the function wizard [page 1061]
Data Services Functions and Procedures [page 1064]
pushdown_sql [page 1235]

3.6.6.12 Query transform Find tab

Use the Find tab to search for a specific item or to replace a specific item in the Query tabs and in the input schema or output schema.

Complete the parameters in the *Find* tab to provide information about the object to find.

Find tab option descriptions

Option	Description
Find dropdown arrow	Specifies the action to perform. • *Find What*: Select to find the item you enter or select in the text box. • *Replace*: Select to find a schema or other object to replace.
Find text box	Specifies what to find or replace. Enter a search term or select a previous search term from the dropdown list.
Find	Finds the term based on the options you set in the *Find* tab. Click *Find* after you complete all applicable criteria options. Appears only when you select *Find what*.
Replace All	Replaces all items that match your search criteria with the content of the *With* text box. Appears only when you select *Replace*.
With	Enter a value or select a value from the dropdown list of previous values. When the search engine finds the item, it replaces the item with what you enter in the *With* text box. Appears only when you select *Replace*.

Option	Description
Schemas	Specifies which schema to search in for the defined value. • *Input* • *Output* • *Both*
Elements	Specifies the element type of the defined value. The drop-down list includes all elements that you find in the input or output schema. For example, select column if you are searching for a specific column.
Where	Specifies the part of the element to match. For example, if you are replacing a column name, select Name.
Match case	Specifies whether to find the item that matches the casing you use when you enter the value in the text box. Also replaces the value based on the casing you use in the *With* text box.

3.6.6.13 Query transform joins

Use the Query editor to define joins that involve two or more tables.

Specifying information on the *FROM* and *WHERE* tabs is like 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.

> **i Note**
>
> Best practice is to define all joins in the *FROM* tab. However, define inner joins 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.

> **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.

> **Example**
>
> 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 Data Services does not find a foreign key or primary key join condition, it does not propose a join condition for that pair of sources.

3.6.6.13.1 Query transform join pair lists

Create a join pair list in the *FROM* tab to create a `FROM` clause.

In the *FROM* tab, create a join pair list that includes the table to join, join type, and join condition. When you create join pairs, keep in mind the following requirements:

- To use a table in a join pair, use the table as an input schema to the query.
- You may specify inner joins and left outer joins in the same `FROM` clause.
- Create at most one join pair that uses a table from the schema list as the source. If you do not explicitly join one table to another table, the software cross joins the table to the result of the final join in the join pairs list. A cross join is also known as a Cartesian product. A Cartesian product is an inner join with an `ON` condition that always evaluates to `TRUE`. A cross join matches every row of one table with every row of another table.
- Select a left source for the first join pair only. The software takes the results of the first join pair to create the left source for the next join pair. The software continues to use the result of the previous join as the left source for the next join until there are no more joins.

  > **Example**
  >
  > In the following screen shot, the first join pair is a left outer join with `ODS_SALESORDER` as the left source and `ODS_CUSTOMER` as the right source. The software takes the result of that join and places it as the left source for the second join pair. The second join pair is a left outer join with `ODS_SALESITEM` as the right source. Finally, the software uses the result of the two left joins as the left source of an inner join with `ODS_MATERIAL` as the right source.
  >
  > Join pairs:
  >
  > | Left | Join Type | Right |
  > |---|---|---|
  > | ODS_SALESORDER | Left outer join | ODS_CUSTOMER |
  > | ↳ | Left outer join | ODS_SALESITEM |
  > | ↳ | Inner join | ODS_MATERIAL |

- The software requires a join condition for each join pair. The join condition consists of the tables from previous join pairs and the table from the current join pair.

  > **Example**
  >
  > The second join pair in the following example may not refer to the table `ODS_MATERIAL` in its join condition because `ODS_MATERIAL` is not used in the previous join pair. The second join pair may only refer to `ODS_SALESORDER`, `ODS_CUSTOMER`, and `ODS_SALESITEM`.

	Join Condition
	ODS_SALESORDER.CUST_ID = ODS_CUSTOMER.CUST_ID
	ODS_SALESITEM.SALES_ORDER_NUMBER = ODS_SALESORDER.SALES_ORDER_NUMBER
	ODS_MATERIAL.MTRL_ID = ODS_SALESITEM.MTRL_ID

- If the Query transform contains only inner joins, you may use the *WHERE* tab to specify join conditions. Though valid, we do not recommend that you specify join conditions in both the *FROM* tab and *WHERE* tab. If you specify conditions in both the *WHERE* and *FROM* tabs, the software combines the conditions to form the join condition for the query at job execution time.
- If a query contains a left outer join, the software treats any conditions specified in the *WHERE* tab as filters. The join conditions for a left outer join may include multiple tables, which you define in the *FROM* tab.
- A Query transform in an ABAP data flow cannot contain mixed inner and left outer joins. The Query transform can have only all inner joins, all left outer joins, or no joins.
- The *Join Pairs* list section can be empty in the following cases:
  - When the input schema for the query contains only one source.
  - When each of the input schemas is intended to be cross-joined.
  - When you define inner joins with a `WHERE` clause in the *WHERE* tab.

3.6.6.13.2 Query transform constructing a join query

Use the Query editor to specify joins with two or more tables with the following join types: Inner join, left outer join, or cross product.

1. Within a data flow, connect the source tables to a Query transform.
2. Double-click the Query transform to open the Query editor.
3. Optional. To exclude input schemas, open the *FROM* tab and deselect the checkbox in the *From* column.

   You may want to exclude an input schema if you no longer need its columns for the:
   - Output schema
   - `ORDER BY` clause
   - `GROUP BY` clause

   However, select at least two input schemas to create a join.
4. In the *FROM* tab, the *Join pairs* pane, select a source from the dropdown list in the *Left* column.
5. Select an option from the *Join Type* column dropdown list. Options include *Left outer join* and *Inner join*.
6. Select a source from the dropdown list in the *Right* column. The dropdown list contains any input schema not used in previous join pairs.
7. Optional. Click the *Smart Editor* icon (···) in the next column. Then drag columns from the *Data* tab in the Smart Editor to the work area to specify the join condition.
8. Optional. Click the *Propose Join* icon ( ) in the next column to have the software generate a join expression.

   If you change the left source, right source, or join type, the software automatically updates the SQL clause.
9. Create a Join Condition in the following ways:

- Use the join condition automatically suggested by the software, if applicable.
- Type the join condition directly in the applicable cell.
- Use the Smart Editor to create a join condition.

Optional. If your `FROM` clause contains only left outer joins or a mix of left outer joins and inner joins, change the order that the software executes the join pairs. 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.

10. When necessary: Filter the result set of a left outer join by placing a restriction in the `ON` clause or within the *WHERE* tab.
11. Optional. Specify the join ranks for each table in the *Join rank* column.
12. When necessary: Specify caching in the *Cache* column.

    The cache value in the Query transform takes precedence over the value specified in a source. For newly created data flows, we recommend that you 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.

    > **i Note**
    >
    > Set the cache type, either in-memory or pageable, at the data flow level.

3.6.6.13.3 Query transform join ranks

Setting the join rank for each join pair does not affect the results, but may enhance performance.

The join rank indicates the rank of the source relative to other tables and files in the data flow. 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.

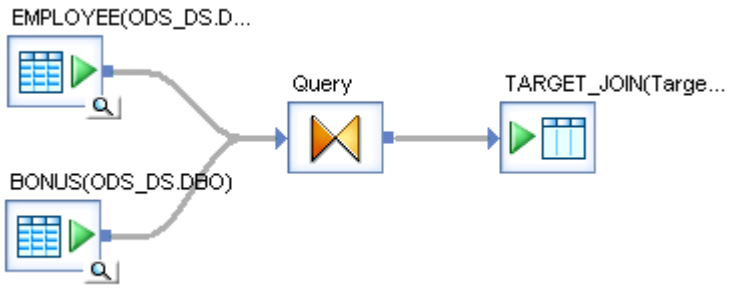
The join rank must be a non-negative integer. When set to its default value of 0, the software determines the join order. The join rank specified in the Query editor overrides any join rank specified in a source. For new jobs, specify the join rank in the Query editor.

3.6.6.13.4 Query transform examples of joins with two sources

The join type in a Query transform determines the results of joining two sources.

The following examples illustrate the different types of joins and the results when you use two source tables.

The following screen shot shows the two tables in a data flow as sources to a Query transform and the target object contains the join results.



Inner and left outer joins

When joining two sources, an inner join returns rows from both sources when 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, the software reproduces all rows in the left source at least once in the result. The software includes in the result only data from the right source that satisfies the join condition.

For rows from the left table that do not have corresponding data from the inner table, the software assigns missing values as null. In an inner join between the tables, the same rows are absent in the result.

> **Example**
>
> The following diagram shows the different join results based an inner join and a left outer join.

Sources joined in a query produce different results based on the join type

Sources

Bonus Table

EmpID	Bonus
1008	1000
1009	1500
1011	1000

Employee Table

EmpID	LastName
1008	Alverez
1009	Davis
1010	Tanaka
1011	Laprais

Join Condition

Employee.EmpID = Bonus.EmpID

Join Results

Inner Join

EmpID	LastName	Bonus
1008	Alverez	1000
1009	Davis	1500
1011	Laprais	1000

An inner join produces 3 rows

Left Outer Join

EmpID	LastName	Bonus
1008	Alverez	1000
1009	Davis	1500
1010	Tanaka	NULL
1011	Laprais	1000

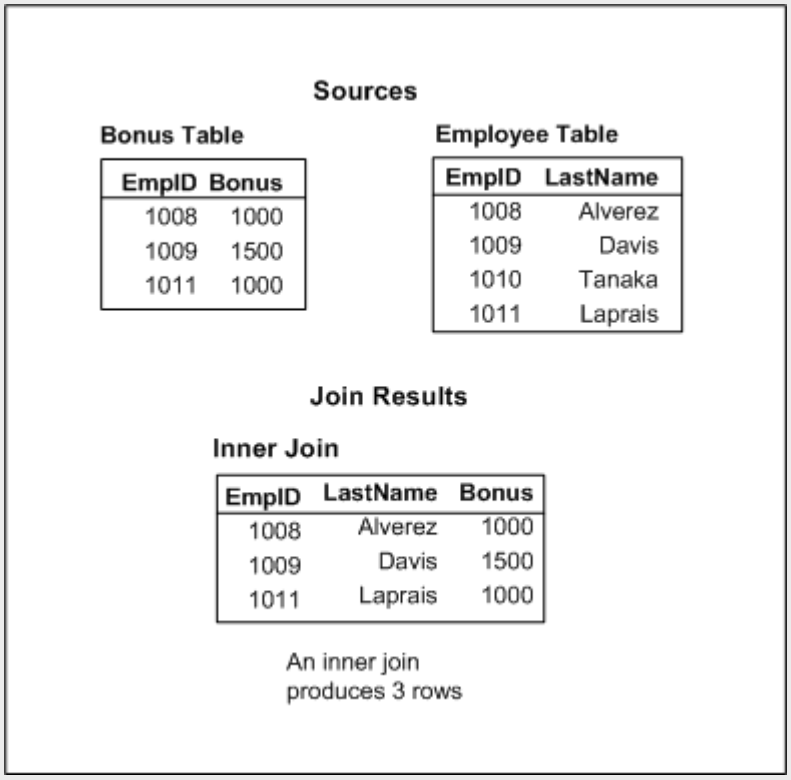
A left outer join produces 4 rows where the additional row is filled with NULLs

Inner joins

Example

Your goal is to produce a list of only the employees receiving bonuses. You have two tables: Employee and Bonus. You join the data in the Employee Table and Bonus Table sources using an inner join to produce three rows, each row contains values from both sources.

The outer table is `Employee` and the inner table is `Bonus`.



SQL syntax:

```
SELECT     EMPLOYEE.EMPID, EMPLOYEE.LASTNAME,
BONUS.BONUS
FROM EMPLOYEE INNER JOIN BONUS
ON     (EMPLOYEE.EMPID = BONUS.EMPID)
```

You define the following join pairs in the *FROM* tab of the Query Editor.

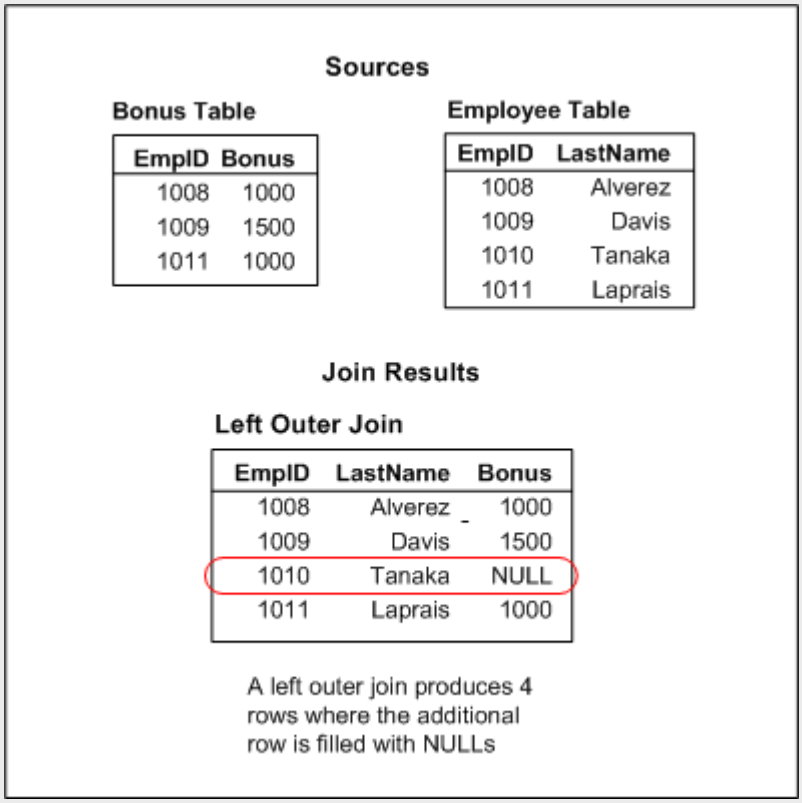
Left	`Employee`
Join Type	Inner join
Right	`Bonus`
Join Condition	`EMPLOYEE.EMPID = BONUS.EMPID`

Left outer joins

Example

Your goal is to produce a list of all employees that receive bonuses. You have an Employee table and a Bonus table. You join the tables 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`.



SQL syntax for the query:

```
SELECT     EMPLOYEE.EMPID, EMPLOYEE.LASTNAME,
BONUS.BONUS
FROM EMPLOYEE LEFT OUTER JOIN BONUS
ON     (EMPLOYEE.EMPID = BONUS.EMPID)
```

You define the following join pairs in the *FROM* tab of the Query Editor.

Left	`Employee`
Join Type	Left outer join
Right	`Bonus`
Join Condition	`EMPLOYEE.EMPID = BONUS.EMPID`

3.6.6.13.5 Query transform examples of joins with more than two sources

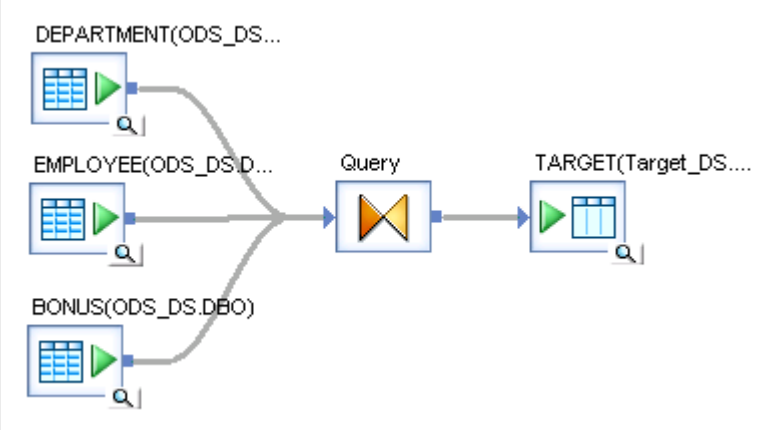
When you have more than two sources for joins, you select the left input source for the first join pair. The software forms all subsequent join pairs using the results of the preceding join pair as the left source.

Mixed joins

Example

Your goal is to produce a table that shows employees listed by department, and the amount of bonus each employee received. You have three tables: Department, Employee, and Bonus. For the first join pair, set the Department table as the left source and the Employee table as the right source. Join them using an inner join. For the second join pair, use the results of the first join as the left source, and the Bonus table as the right source. Join them using a left outer join.

The following illustrates the data flow In Data Services.



Sources:

Department table		Employee table			Bonus table	
DeptID	**Department**	**EmpID**	**LastName**	**DeptID**	**EmpID**	**Bonus**
01	Accounting	1008	Alverez	01	1008	1000
02	Finance	1009	Davis	02	1009	1500
03	Sales	1010	Tanaka	01	1011	1000
04	Marketing	1011	Laprais	01		

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`

SQL syntax:

```
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)
```

The joins produce four rows with the bonus information NULL where there was no value in the bonus table.

DeptID	Department	LastName	Bonus
1	Accounting	Alverez	1000
1	Accounting	Tanaka	NULL
1	Accounting	Laprais	1000
2	Finance	Davis	1500

3.6.6.13.6 Query transform examples of restricted left outer joins

The results for a left outer join that is restricted depends on whether you place the restriction in the `ON` condition or in the `WHERE` clause.

About restricting left outer joins

An unrestricted left outer join between two tables results in all rows from the left outer table with data from the inner table that satisfies the join condition. The software assigns NULL values to rows that do not contain data.

The location of the restriction depends on what information you need the query to return. The following table shows the result set defined for each case.

Result set	Restriction	Query editor tab
All rows of the left source, including null values.	ON	FROM
Rows for which the restriction is true.	WHERE	WHERE

> **i Note**
>
> For inner joins, it does not matter where you place the restriction; the result set is the same in either case.

Restriction placed in the ON condition

Example

You have two tables: CUSTOMER and SALESORDER. You place the CUSTOMER table as the left source and SALESORDER as the right source. You join the tables with a left outer join. You create the restriction in the *FROM* tab using an `ON` condition.

Sources

Customer Table

CUST_ID	Name
DT03	Pottery Ceramics
KT02	Major Resellers
SA01	Trusty Manufacturers
SA02	New Times
SA03	Popular Press

Sales Order Table

CUST_ID	Order Number
SA01	PT22221000
SA01	PT22221001
SA01	PT22221002
SA02	PT22221005
SA02	PT22221006
KT02	PT22221009
KT02	PT22221010
DT03	PT22221012

Join Results

Restriction in ON condition

CUST_ID	Name	Order Number
DT03	Pottery Ceramics	<Null>
KT02	Major Resellers	<Null>
SA01	Trusty Manufacturers	PT22221000
SA01	Trusty Manufacturers	PT22221001
SA01	Trusty Manufacturers	PT22221002
SA02	New Times	<Null>
SA03	Popular Press	<Null>

The join condition in the `FROM` clause contains the following clause: `AND CUSTOMER.CUST_ID = 'SA01'`. The result returns all rows of the left source, CUSTOMER, including those rows with NULL values.

The following screen capture shows the *Join pairs* pane of the *FROM* tab:

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'

SQL syntax:

```
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

You have two tables: CUSTOMER and SALESORDER. You place the CUSTOMER table as the left source and SALESORDER as the right source. You join the tables with a left outer join. You create the restriction in the *WHERE* tab using a `WHERE` condition. The restriction is: `CUSTOMER.CUST_ID = 'SA01'`.

The result of a left outer join returns only the rows for which the restriction is true. The following diagram shows the data in the source tables and the results of the join:

Sources

Customer Table

CUST_ID	Name
DT03	Pottery Ceramics
KT02	Major Resellers
SA01	Trusty Manufacturers
SA02	New Times
SA03	Popular Press

Sales Order Table

CUST_ID	Order Number
SA01	PT22221000
SA01	PT22221001
SA01	PT22221002
SA02	PT22221005
SA02	PT22221006
KT02	PT22221009
KT02	PT22221010
DT03	PT22221012

Join Results

Restriction in WHERE clause

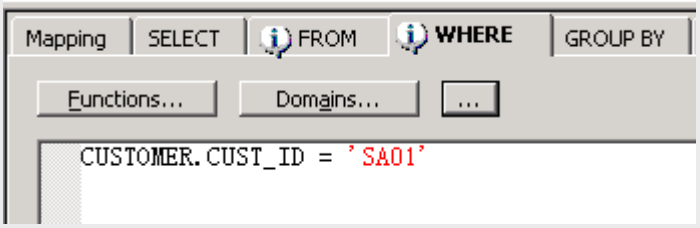
CUST_ID	Name	Order Number
SA01	Trusty Manufacturers	PT22221000
SA01	Trusty Manufacturers	PT22221001
SA01	Trusty Manufacturers	PT22221002

The following screen capture shows the join in the *Join pairs* pane of the *FROM* tab:

Join pairs:

Left	Join Type	Right			Join Condition
ODS_CUSTOMER	Left outer join	ODS_SALESORDER	...		ODS_SALESORDER.CUST_ID = ODS_CUSTOMER.CUST_ID

The following screen capture shows the restriction in the *WHERE* tab:



SQL query:

```
SELECT ODS_CUSTOMER.CUST_ID, ODS_CUSTOMER.NAME1,
 ODS_SALESORDER.SALES_ORDER_NUMBER
FROM DBO.ODS_CUSTOMER ODS_CUSTOMER LEFT OUTER JOIN
 DBO.ODS_SALESORDER ODS_SALESORDER
ON (ODS_SALESORDER.CUST_ID=ODS_CUSTOMER.CUST_ID)
WHERE (ODS_CUSTOMER.CUST_ID = 'SA01')
```

3.6.6.13.7 Query transform viewing optimized SQL

View the optimized SQL before you execute a job to verify that the software generates the commands you expect.

If you determine the generated SQL commands are not what you expect, alter the design to improve the data flow and to improve processing performance. To view the optimized SQL, perform the following steps:

1. In Designer, validate and save the applicable data flow.
2. With the data flow open in the workspace, select *Validation* > *Display Optimized SQL*.

> **Note**
>
> If you select *Display Optimized SQL* when there are no SQL sources in the data flow, Data Services issues an alert message and does not display the SQL.

The *Optimized SQL* dialog box opens displaying a list of datastores on the left, and the optimized SQL code for the selected datastore on the right. By default, the software selects the first datastore. Data Services shows only the SELECT generated for table sources and INSERT INTO...SELECT for table targets. Data Services does not show SQL generated for SQL sources that are not table sources. For example, it does not show SQL for:

- Lookup function
- Key_Generation function

- Key_Generation transform
- Table_Comparison transform

3. Select a datastore name from the list of datastores on the left to view the SQL that the data flow applies against the corresponding database or application.

> **Example**
>
> The following optimized SQL illustrates a full push-down operation (INSERT INTO...SELECT bolded in sample code). In the data flow, a Data_Transfer transform creates a transfer table that Data Services loads directly into the target.
>
> ```
> INSERT INTO "DBO"."ORDER_AGG" ("SHIPCOUNTRY","SHIPREGION", "SALES_AGG")
> SELECT "TS_Query_Lookup"."SHIPCOUNTRY" ,
> "TS_Query_Lookup"."SHIPREGION" ,sum("TS_Query_Lookup"."SALES")
> FROM"DBO"."TRANS2""TS_Query_Lookup"
> GROUP BY "TS_Query_Lookup"."SHIPCOUNTRY" , "TS_Query_Lookup"."SHIPREGION"
> ```

4. Optional. Perform the following tasks with the optimized SQL in view:
   - Click *Find* to search for a string in the optimized SQL.
   - Click *Save As* to save the optimized SQL as an SQL file.
5. If you change anything in your data flow, make sure to save the data flow to update the displayed SQL.

After time, you may change the data flow. After any change, remember to save the data flow and view the updated optimized SQL.

3.6.6.13.8 Query transform Optimized SQL window

The *Optimized SQL* window shows a list of datastores and the optimized SQL code for the selected datastore.

Open the Optimized SQL window by selecting *Validation* > *Display Optimized SQL* while you have a saved data flow opened in your workspace. The software shows only the `SELECT` statement generated for table sources and `INSERT INTO... SELECT` for targets. It only shows the SQL generated for SQL sources that are table sources. For example, the software does not show the SQL generated for the following:

- Lookup function
- Key_generation function
- Key_Generation transform
- Table_Comparison transform

> **Note**
>
> The *Optimized SQL* window displays the existing SQL statement that is currently in the repository. If you change your data flow, save the data flow to update the repository and so that the *Optimized SQL* window displays your current SQL statement.

3.6.6.13.9 Query transform outer join vs. Lookup function

A `LOOKUP` function and an outer join have similar results, however, the `LOOKUP` function is limited certain actions.

The following table compares the actions that the `LOOKUP` function and the left outer join can perform.

Lookup function	Left Outer join
Returns only one column value for each comparison	Returns all column values for each comparison
Use against only one source at a time	Provides similar capability to multiple lookup calls.
Does not produce the same results as non equality joins such as `A.x<B.y`	Allows non equality joins
Permits default values other than nulls	
Can be cached when desired	

In addition, an outer join query requires that you join the sources in a hierarchical order:

- A source can only be the inner table of one left outer join
- A source cannot be joined with itself using a left outer join in a single Query transform

3.6.6.14 Query transform remapping schemas

Remap schemas in the Query transform if you remove a source and add a different source in your data flow.

1. Open the Query transform in your data flow.
2. In the Mapping tab, click the *Schema Remapping* button in the lower right.

   The *Replace Obsolete Schema* dialog box opens.
3. Select the source schema that you disconnected from the Query transform in the *Specify obsolete schema* dropdown list.

   The obsolete schema list displays only the top level input schemas. If you disconnected a nested schema, select the name of the related top-level schema.
4. Select the new source schema that you connected to the Query transform from the *Choose correct schema* dropdown list.
5. Click *Remap*. A message displays the number of columns that were remapped.

   Example

   ```
   Schema "ODS_SALESORDER" was replaced by schema "Validation_Pass" in 11
   column names.
   ```

3.6.7 Row_Generation transform

Use the Row_Generation transform to produce a data set that contains one column.

Row_Generation transform information

Characteristic	Description
	Row_Generation transform icon
Use	Produces a data set with a single column. The column contains integers. 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.
Content objects	Sample transform

3.6.7.1 Row_Generation transform options

The Row_Generation transform editor includes the target schema, and transform options.

Row_Generation transform option descriptions

Option	Description
Cache	Indicates whether the software reads the required data from the source and loads it into memory or pageable cache. The software reads an inner source of a join for each row of an outer source. Therefore, you might want to cache a source when you use it as an inner source in a join. • *Yes*: The source is always cached unless it is the outer-most source in a join. *Yes* is the default setting. • *No*: The source is never cached. The 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.

Option	Description
Join rank	Indicates the rank of the output data set relative to other tables and files joined in a data flow. The software joins sources with higher join ranks before joining sources with lower join ranks. Enter a non-negative integer. Default value is 0. The 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.
Row count	Indicates the number of rows in the output data set. Enter a positive integer. For added flexibility, enter a global variable or substitution parameter.
Row number starts at	Specifies a value with which row numbering starts. Enter a positive integer. If you set to 1, the software labels the first row with 1, the second row with 2, and so on. If you leave blank, the software starts numbering at zero (0). For added flexibility, enter a global variable or substitution parameter.

3.6.7.2 Row_Generation transforrm data inputs and outputs

The Row_Generation transform does not take input data. It outputs a data set that contains one column.

The Row_Generation transform produces a data set with a single column and the number of rows that you specify in the transform editor. The rows contain integer values in sequence starting from the value that you enter in the transform editor, and incrementing by one in each row.

3.6.8 SQL transform

Use the SQL transform to perform the indicated SQL query operation on input data.

SQL transform information

Characteristic	Description
	SQL transform icon

Characteristic	Description
Uses	Use this transform to perform a standard SQL operation when other transforms cannot. Set options in the SQL transform editor to specify a datastore, join rank, cache, array fetch size, and SQL text. **i Note** The SQL transform supports only a single `SELECT` statement.
Content objects	Sample transform

3.6.8.1 SQL transform options

Set options in the SQL transform editor to create and execute a SQL statement on tables in a defined datastore.

SQL transform option descriptions

Option	Description
Array fetch size	Indicates the number of rows retrieved in a single request to a source database. Default value is 1000. Maximum value is 5000. When you set a higher number, you reduce the number of requests, which lowers network traffic. A higher value could improve performance. This option is available for source tables from DB2, Informix, ODBC, Oracle, and SQL Server datastores. When retrieving a column with an Oracle long data type, Data Services automatically sets Array Fetch Size to 1. If a column has an Oracle long data type, Data Services can only retrieve one row at a time.
Cache	Indicates whether the software should load output from the transform into memory or pageable cache. The software reads an inner source of a join for each row of an outer source. Therefore, you might want to cache output when it is used in a subsequent transform as an inner source in a join. • *Yes*: The source is cached. The default is *Yes*. • *No*: The source is not cached. 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.

Option	Description
Database type	Specifies the database type and versions available in the selected datastore. If you have multiple configurations in a datastore, select a *Database type* to enable the software to quickly set SQL transform values in data flows. Additionally, select a database type to supply unique SQL text for each database type and version in any SQL transform instance. When you select a database type, Data Services provides the value you set previously for SQL text in that particular database type. To add or remove items in the *Database type* list box, edit the datastore configuration information using the Datastore Editor. The following describes how Data Services determines SQL text values. • If the datastore has more than one configuration and the configurations use different database types or versions, then Data Services uses information from the *Use values from box* in the *Create New Configuration* dialog box of the datastore editor. > **i Note** > Join rank, Cache, and Array fetch size values remain the same as those set in the initial configuration. You cannot have more than one set of these values in a datastore. • If you select *Restore values if they already exist* in the *Create New Configuration* dialog box of the datastore editor, Data Services looks for 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 the datastore configuration was deleted. Data Services retains all SQL transform values saved with every datastore configuration. If such values exist, then the software restores those values. Otherwise, it gets the values from the configuration you select from the *Use values from* option. If the SQL text in a SQL transform is not correct for the database type, modify the SQL text. If the SQL text contains hard-coded owner names or database names, consider replacing them with variables to limit the number of modifications you need for new database types. > **i Note** > Because Data Services only provides values for variables during runtime, do not use variables in the SQL text of a SQL transform when you use the *Update Schema* button. To support portability, add variables afterwards.
Datastore	Indicates the name of the datastore that Data Services uses to access the tables that you include in the *SQL text* text box.
Join rank	Indicates the rank of the output data set relative to other tables and files joined in a data flow. The software joins sources with higher join ranks before joining sources with lower join ranks. Enter a non-negative integer. Default value is 0. 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.

Option	Description
SQL text	Specifies the text of the SQL query. The software passes the SQL text string to the database server. Enter SQL text directly in the text box, or click the *Smart Editor* icon to create the SQL query. You do not need to put enclosing quotes around the SQL text. However, if the syntax rules for the DBMS require it, include enclosing quotes around the table and column names.
Update schema	Click this option to automatically calculate and populate the output schema for the SQL SELECT statement.

3.6.8.2 SQL transform data inputs and outputs

The SQL transform does not accept input data, but it outputs data that is a result of the SQL select statement that you set.

There are two ways of defining the output schema for a SQL transform: Automatic and manual.

Automatic

Enter a SQL statement and click *Update schema* in the SQL editor. The transform executes the SQL select statement, obtains column information from the database, and populates the output schema.

Manual

You define output columns in the *Schema Out* pane of the SQL transform editor. Define the same number of columns in the *Schema Out* pane as the number of columns returned by the SQL query.

The column names and data types of the output columns do not need to 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. However, when Data Services evaluates an expression, which includes operands of more than one data type, it attempts to convert the operands to the same data type first with one exception. Data Services pushes down national character-set data types. Data Services flags errors for illegal conversions.

The output data set cannot contain hierarchical data.

Make sure that the column data types of the two sets of output columns are exact matches. If you choose to have different data types, ensure that they are compatible. If the data types are not compatible, the underlying database manager issues a runtime error.

Related Information

varchar [page 1058]

3.6.9 User Defined transform

Use the User Defined transform to define Python expressions to control how the transform processes data.

Access options by opening the User Defined transform in a data flow and selecting *Tools* > *User Defined Editor*.

User-Defined transform information

Characteristic	Description
	User-Defined transform icon
Uses	The applications for the User-Defined transform are limitless. The User-Defined transform can do just about anything that you can write Python code to do. Use the User-Defined transform to generate new records, populate a field with a specific value, create a file, connect to a Web site, or send an e-mail, just to name a few possibilities. Place this transform anywhere in your data flow. You as the transform creator can place restrictions for the transform placement. **i Note** The User-Defined transform is flexible and powerful. However, the Query transform is often more scalable and faster, and uses less memory that a User-Defined transform.
Content objects	Sample transform configurations and blueprints.

3.6.9.1 User Defined transform editor

The User Defined transform editor contains the *Input*, *Options*, and *Output* tabs like other data quality transforms, but you set options in the *User-Defined Editor* dialog box.

To set User-Defined transform options, access the *User-Defined Editor*. Open the editor by first opening the *Options* tab and clicking the *Edit Options* button.

The options in the *Options* tab in the transform editor are not editable.

> ⚠ Caution
>
> Map input fields for Python expressions. Map the input field in the *Input* tab to a recognized field name from Data Services. If it is not mapped, and you use the field in a Python expression, the software issues 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.
> ```

For details about configuring the User-Defined transform, see the Designer Guide.

Related Information

Associate, Match, and User-Defined transform editors

3.6.9.2 User Defined transform options

Use full Python scripting language in the User Defined transform for custom processing in a data flow.

The *Mode* options are at the top of the left pane of the editor.

Option	Description
Mode	Specifies how the software applies the Python expression in the transform. *Per collection*: Applies the expression to the entire data collection. Use this option when adding new records to the data flow that did not exist before. When you select *Per collection*, the transform displays the *Group Forming* options in the left pane in which you set break groups and candidate selection. *Per record*: Applies the expression to each record. You cannot add new records into the data flow with this option. When you select *Per collection*, complete the options in the Custom options pane at right. *Per record* is the default option.

When you select the *Per record* mode option, complete the *Custom options* table in the left pane of the Editor. The custom options adjust the runtime behavior for the User-Defined transform. Click *Add Row* to activate the options in the Custom options table.

Option	Description
Custom Option	Specifies variables for use in your Python expression. Type a variable in the *Custom Option* column after you add a row.
Value	Specifies the value for the custom option. Enter the value in the *Value* column for the applicable custom option.

Option	Description
Add Row	Enables the options in the *Custom options* pane when you add the first row. Click to add additional rows to the *Custom options* table.
Remove Row	Deletes the highlighted row.

3.6.9.3 User defined transform Python Expression Editor

Use the Python Expression Editor options to create your expressions.

This option group is required.

Click the *Launch Python Editor* button to access the *Python Expression editor*.

Option	Description
Python	Contains the Python expression that the transform applies to the specified field or fields. Enter manually, or use the Smart Editor.
Launch Python Editor	Opens the Smart Editor dialog box. Use the options to build your Python expression. When you are finished, the software displays your expression in the *Python* pane.

Related Information

Python [page 1467]

3.6.9.4 User Defined transform Group Forming options

When you select the Per collection mode, optionally complete the *Group Forming* options to create break groups.

Group forming enables you to group and prioritize records for better match accuracy and efficiency, as well as performing custom Python processing.

Break groups

With break groups, group records based on common field values. Creating break groups increases performance. For example, if you use the Match transform, break groups lower the number of comparisons and may improve the performance 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* node in the left pane and select *Add Candidate Selection*.

> **i Note**
>
> Candidate selection works with relational databases only; it does not work with flat files.

3.6.9.5 Match and User Defined transform Break Group options

To define common field values for match break groups, set break group options.

Find the *Break Group* options in the Match Editor or the User-Defined Editor.

Break group option descriptions

Option	Description
Split records into break groups	Specifies that the transform forms break groups to reduce the total number of comparisons made. This option is selected by default. The most common case for not selecting this option is for real-time jobs. If data in a real-time job forms one break group, do not select this option. Instead, for optimal real-time matching, use candidate selection. Candidate selection selects a limited number of records from a relational database. **⚠ Caution** Deselect this option with caution within a batch data flow. The size of a break group may not exceed 2 GB. If you use this option in a batch data flow, also set the *Maximum allowable break group size (in records)* option so that the collection does not exceed the size limit. If it does exceed the limit, the software stops the data flow execution. **→ Tip** Calculate break group size by multiplying the record length by the number of records in the break group.
Field	Specifies a mapped input field to include in the break key. For a more complex break key, define that field using an upstream Query transform and select the field in this option.

Option	Description
Add Row	Select to add a new row to the Break key table.
Remove Row	Click to remove a row from the Break key table.
Start Position	Specifies the start position of the field. Valid values for a field length of n are 1 to n and -1 to -n. Negative start values signify that the transform counts the start position from the right. **Example** For example, a field with a length of 7 contains JOHNSON. A start position of 2 means start with "O." A start position of -4 means start with the first "N." If the field has a length of 20, the software still counts the start position of -4 at the letter "N" because the negative start value starts from the actual length of the string, not at the start of the field.
Length	Specifies the number of characters in the field you want included in the break key.
Break key case sensitive	Specifies whether to treat the break key as case sensitive. • *Yes:* Treat the break key as case sensitive. • *No:* Do not treat the break key as case sensitive. **Example** You create a break key using the Primary_Name (street) field. You set *Break key case sensitive* to *Yes*. Input consists of: • One input record contains Primary_Name = Main • Another input record contains Primary_Name = main The software forms separate break groups with the following input values: 1. Break group 1: Main 2. Break group 2: main
Replace NULL with empty string	Specifies whether to convert NULL values with an empty string in the break key. • *Yes:* Convert NULL to an empty string. • *No:* Do not convert to an empty string.
Right pad fields with blanks	Specify whether to right pad fields with blanks. • *Yes:* Right-pad fields with blank spaces. • *No:* Do not right-pad fields. Because the transform uses the break key for sorting and aggregating, the transform is sensitive to the position in which data is placed. By right-padding the break key fields you can ensure that the software forms break groups properly. If the *Replace NULL with empty string* option is set to *Yes* and this option is set to *Yes*, then the transform replaces fields with NULL values with all spaces to the length of the field.

Option	Description
Input already sorted	Specifies that the input data has already been sorted, and the transform should not sort it again. • *Yes:* The transform does not resort the input data. • *No:* The transform sorts the break keys at runtime before forming break groups. → Tip if you require a more complex break key, use a Query transform to create it, and use the ORDER BY operation to order your data.
Maximum allowable break group size (in records)	Specifies the maximum number of records allowed in a break group. An empty value or zero means that there is no limit on the break group size. Control the amount of memory used during processing by specifying the number of records processed at one time. If more records make it into a single break group than the number that you specify, then the software issues an execution error and stops the job.

Related Information

Break keys and candidate selection

3.6.9.6 Match and User Defined transform candidate selection options

To append records from a relational database to an existing data collection before matching, set options in the Candidate Selection editor.

Candidate selection speeds up match processing. The Candidate Selection editor appears in both the Match Editor and the User-Defined Editor.

Candidate Selection Editor option descriptions

Option	Description
Datastore	Specifies the datastore from which you load candidate set. The dropdown list is populated with all valid SQL and persistent cache datastores. For persistent cache datastores, consider using persistent cache when the datastore doesn't often change. In addition to using persistent cache datastores, also use persistent cache datastores created from a flat file. **i Note** Create persistent cache datastores for candidate selection using a data flow in double-byte mode. To change to double-byte mode: 1. Change the locale setting in the Data Services Locale Selector. Set the code page to UTF-8. 2. Run the job to generate the persistent cache. 3. After the job completes, change the code page back to the original setting. If you choose a persistent cache datastore, you cannot use custom SQL.
Cache type	Specifies the type of cache the datastore uses. • *No_Cache:* Specifies that the transform sends each query to the database. No_Cache captures data at a point in time. The data doesn't change until the job restarts. • *Pre_Load_Cache:* Specifies that the entire secondary table is cached to a local disk or memory. Choose this option when the data doesn't often change. Set this option to improve performance. However, a trade-off may be that the cache uses more memory.
Auto-generate SQL	Specifies whether the transform automatically generates the SQL. This option allows you to query a simple single table. If you join tables or create a complex WHERE clause, select the *Create custom SQL* option.
Table	Specifies a table from the datastore.
Use break column from database	Select this option if your database already contains a column that corresponds to the break key field.
Break key field	Select the column from the secondary table that contains the break key field.
Create custom SQL	Select to create custom SQL.
Launch SQL Editor	Opens the SQL editor. This button is only enabled if you select the *Create custom SQL* option.
Use constant source value	Select to assign records to a physical source for generating appropriate statistics.
Physical source value	Type a value for your physical source. This transform places the value in the physical source field you select.
Physical source field	Select the mapped field that contains the physical source name.

Option	Description
Add DB columns to mapping table	If you are using the *Create custom SQL* option, clicking this button adds only the database columns that appear in the SELECT statement and in the order that they appear in the SELECT statement. If you are using the *Auto-generate SQL* option, clicking this button adds ALL database columns, in the order that they appear in the table schema. **i Note** If you do not associate an input field to any of these columns in the column mapping table, the transform removes the columns when you close the window.

Column mapping table

The Column mapping pane allows you to specify which mapped field in the data flow each database selected field is assigned to.

Column	Description
Break key	Specifies whether this field is used as part of your break key.
Field	Each cell contains a list of the mapped names from the input fields in the transform.
DB column	Each cell contains a list of the column names in your database table or the selected columns from a custom query. Match the data of a column in your database to the data of a mapped field.

Related Information

Break keys and candidate selection

3.6.9.7 User Defined transform content objects

We provide content objects to help you create and complete transforms.

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 User-Defined transform.

You can use transform configurations in your data flows or as an example of a typical transform. After you place an instance of the transform configuration in a data flow, you can override these preset defaults. You can also create your own transform configuration, either by replicating an existing transform configuration or creating a new one.

Sample blueprints and other objects

We have created Data Quality blueprints and other content objects to help you set up Data Services jobs. We've identified a number of common scenarios that you are likely to perform with Data Services. For each scenario, we've included a blueprint that is already set up to solve the business problem in that scenario.

Related Information

Downloading blueprints and other content objects [page 347]
Data Quality transforms [page 341]

3.6.10 Validation transform

Use the Validation transform to qualify a data set based on rules for input schema columns.

Validation transform information

Characteristic	Description
	Validation transform icon
Use	The Validation transform qualifies a data set based on rules for input schema columns. Apply multiple rules per column or bind a single reuseable rule to multiple columns. Enter the single rule in the form of a validation function. The Validation transform identifies the row, column, or columns for each validation failure. Also use the Validation transform to filter or replace data that fails your criteria. When you enable a validation rule for a column, a check mark appears next to the column in the input schema.
Content objects	Sample transform

3.6.10.1 Validation transform Rules tab options

Use the options in the Validation Rules tab to create validation rules and to apply substitutions to selected rules.

The Validation Rules tab is one of two tabs in the lower pane of the Validation editor.

Rules tab option descriptions

Option	Description
Add	Button that opens the *Rule Editor* dialog box.
Edit	Button that enables you to edit a selected rule. Opens the *Rule Editor* dialog box.
Remove	Button that removes a selected row or multiple selected rows.
Column descriptions for the Rules table	

Option	Description
Enabled	Indicates that a rule is enabled. There are two ways to enable or disable a rule. • Select or deselect *Enabled* in the *Rule Editor*. • Click the checkbox in the *Rules* table so a checkmark appears or disappears for the corresponding rule.
Rule	Displays the rule syntax.
Ignore if NULL	Indicates whether the transform ignores the rule during processing if the value is NULL. • *Yes*: Transform ignores the rule if value is NULL. • *No*: Transform implements the rule even when the value is NULL Set the option in the *Rule Editor*.
Action on Fail	Identifies the action the transform takes if a column fails the rule. • *Send to Fail*: Sends the failed record to a target specified for failed records. *Send to Fail* is the default. • *Send to Pass*: Sends the failed record to a target specified for passed records. • *Send to Both*: Sends the failed record to the failed target and to the passed target. **i Note** If you choose Send to Pass or Send to Both, choose a substitute value or expression for the failed value. The software displays the failed rule in the bottom pane.
Name	Specifies the name of the rule. Set the name in the Rule Editor.
Description	Specifies the description of the rule. Enter a description in the Rule Editor.

Column descriptions for the substitute for failed and passed rules table in the bottom pane.

i Note

Define the column and substitution expression in the table cells. Optionally use the Smart Editor to construct the substitute expression.

Option	Description
Enabled	Indicates that the substitution expression is enabled or disabled.
Column	Specifies the column name the transform replaces with the substitution value.
Expression	Defines the substitution constant, variable, or function call. During job execution, Data Services converts substitute values to a corresponding column data type: integer, decimal, varchar, date, datetime, timestamp, or time.
Remove	Deletes the selected row or rows.
Ellipses	Button that opens the Smart Editor to create the substitution expression.

3.6.10.2 Validation transform Options tab

Use the options in the *Validation Transform Options* tab to specify post processing actions

Validation Transform Options descriptions

Option	Description
On Failure	Specifies the type of data the software collects for failed columns. View the data in the Data Validation dashboards metadata reports. • *Collect data validation statistics*: Collects data for columns that fail validation. • *Collect sample data*: Collects sample data for columns that failed validation.
Output Rule Violation Information	Specifies whether the transform includes a row identification column in the output schema designated for failed records. *Create column DI_ROWID on Validation_Fail*: Select to have the transform include a row identification column labeled DI_ROWID.

3.6.10.3 Validation transform Rule Editor options

Use the options in the Rule Editor to create validation rules for the Validation transform.

Rule Editor option descriptions

Option	Description
Name	Specifies the name of the validation rule.
Description	Optional. Specifies a description of the validation rule.
Enabled	Indicates that a rule is enabled. There are two ways to enable or disable a rule. • Select or deselect *Enabled* in the Indicates whether the transform ignores the rule during processing if the value is NULL.*Rule Editor*. • Click the checkbox in the *Rules* table so a checkmark appears or disappears for the corresponding rule.
Ignore if NULL	• *Yes*Indicates whether the transform ignores the rule during processing if: Transform ignores the rule if value is NULL. • *No*: Transform implements the rule even when the value is NULL Set the option in the *Rule Editor*.
Action on Fail	Identifies the action the transform takes if a column fails the rule. • *Send to Fail*: Sends the failed record to a target specified for failed records. *Send to Fail* is the default. • *Send to Pass*: Sends the failed record to a target specified for passed records. • *Send to Both*: Sends the failed record to the failed target and to the passed target. **i Note** If you choose Send to Pass or Send to Both, choose a substitute value or expression for the failed value. The software displays the failed rule in the bottom pane.
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 is available in the dropdown list.
Bindings	Each parameter required by the function displays. Provide an argument for each parameter. The argument can be a constant, column (from the input schema), substitution variable, data flow variable, or global variable. You can type the value into the cell, or select the dropdown arrow to display the available columns and variables. Clear the *Score* checkbox to not include the binding in the Fail output. See the following example for more information about Score.
Column Validation	Select to base the rule on a single input column.
Column	Click the dropdown arrow to select a column on which to process the rule.

Option	Description
Condition	Select a condition (usually with an expression) to define the column-based rule. Available condition operators include: <, >, <=, >=, <>, = *IS NULL, IS NOT NULL* *LIKE* *IN SET* *BETWEEN*: Use to specify a range of values. *Match Pattern*: Lets you enter a pattern based on the Data Services match_pattern function. *Exists in Table*: Select to specify that a column value must exist in a column of another table. Click the dropdown arrow to open the *Input Parameter* window and browse to the desired input. This option uses the LOOKUP_EXT function. Define the NOT NULL constraint for the column in the LOOKUP table to ensure the *Exists in table* condition executes properly. *Custom Condition*: Select to create more complex expressions by linking to the smart editor (*Ellipses* button) or function wizard (*Function* button). An edit box opens for you to enter your expression.

Example: Understanding Score

Say that 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 Country and ZIP columns. However in the Rule Editor when defining the parameter, you would select the *Score* checkbox 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 appears as having had bad data, not the Country column.

Related Information

match_pattern [page 1216]
lookup_ext [page 1194]
Smart editor

3.6.10.4 Validation transform data outputs

The Validation transform outputs to multiple target objects, but the main targets are: Pass, Fail, and Rule Violation.

The data that the Validation transform outputs is based on the conditions that you specify in the transform. Set the data outputs when you connect the output of the Validation transform with target objects. For example, target objects can be a Pass object, a fail object, or both pass and fail objects. You can also load Pass and Fail data into multiple targets.

Output object	Description
Pass	When you choose a Pass output, the output schema is identical to the input schema.

Output object	Description
Fail	When you choose a Fail output, the software adds the following output fields to the Fail output schema: `<DI_ERRORACTION>` and `<DI_ERRORCOLUMNS>`. • `<DI_ERRORACTION>`: This field indicates where the software sent Failed data. **Value** — **Description** *B* — Failed data sent to both Pass and Fail outputs. *F* — Failed data sent to the Fail output. **i Note** If you choose to send failed data to the Pass output, the software does not track the results. The software does not add columns to Pass output types. Therefore, you may substitute a value for failed data that you send to the Pass data output. The software maintains the input schema in the Pass output. • `<DI_ERRORCOLUMNS>`: This column displays all error messages for columns with failed rules. The names of input columns associated with each message are separated by colons. For example: `<ValidationTransformName> failed rule(s): c1:c2` **i Note** If a row has conditions set for multiple columns and the Pass, Fail, and Both actions are specified for the row, then the precedence order is Fail, Both, Pass. For example, if you set the action to *Send to Fail* and the column fails, then the transform sends the whole row to the Fail output. The transform ignores other actions for other validation columns in the row.
RuleViolation	You use the RuleViolation output to capture each row that represents the unsuccessful execution of a validation rule. The following output schema fields contain data to help you understand which rule failed: • `<DI_ROWID>`: Multiple validation rules can fail on a single input row. However, the output schema Fail only outputs a single row to report each failure. For complete information about every failed rule, the `<DI_ROWID>` field associates rows that are sent to the Fail output to rows that are recorded in the RuleViolation output. To link the `<DI_ROWID>` to the actual data row, select the *Create column DI_ROWID on Validation_Fail* option in the Validation Transform Options tab. With the option selected, the transform includes `<DI_ROWID>` in the Fail output schema. • `<DI_RULENAME>` and `<DI_COLUMNNAME>`: Associate columns with multiple rules, and associate a rule with multiple columns, so that each row identifies both the rule name and the column name. With this association, you can uniquely identify a particular rule-column pair. The transform adds only columns with *Score* selected in the *Rule Editor*.

3.6.10.5 Validation transform nested schemas

Use the Validation transform with nested schemas.

associate any scalar column in a nested schema with a validation rule. Use other nested columns in a validation condition as long as they share the same parents with the column on which the rule is defined. Data Services

generates additional columns (DI_ERRRORACTION and DI_ERRORCOLUMNS) for the Fail output target at the top level of the schema only. You must express columns with fully qualified names.

3.6.11 XML_Map transform

Use the XML_Map transform to produce a single target data set from one or more hierarchical source data sets.

XML_Map transform information

Characteristic	Description
	XML_Map transform icon
Use	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 transform also supports flat data structures such as database tables or flat files as both source and target data sets. Use the XML_Map transform to perform a variety of tasks. For example: • Create a hierarchical target data structure such as XML or IDoc from a hierarchical source data structure. • Create a hierarchical target data structure based on data from flat tables. • Create a flat target data set such as a database table from data in a hierarchical source data structure. The XML_Map transform works in two modes: Normal and batch. • Normal mode: Transform handles data on a row by row basis before sending it to the next transform. • Batch mode: Transform handles data as a block of rows before sending it to the next transform. **i Note** The XML_Map transform icon changes based on the selected mode.
Content objects	Sample transform.

XML_Map transform data inputs and outputs [page 1006]
The data source and data targets for the XML_Map transform differ based on the mode you choose.

XML_Map transform batch mode operation [page 1006]
In batch mode, the XML_Map transform processes data in blocks of rows before it sends the data to the next object in the data flow.

XML_Map transform normal mode operation [page 1008]
In normal mode, the transform processes data on a row by row basis before sending it to the next object in the data flow.

Example: Nesting data with the XML_Map transform [page 1021]
Use the XML_Map transform to take an input schema and produce a nested output.

Example: Unnesting data with the XML_Map transform [page 1022]
Use the XML_Map transform to unnest a nested input schema and output to a flat file.

Example: Transform hierarchical source to hierarchical target with XML_Map [page 1023]
Use the XML_Map transform to produce a hierarchical target that is different from the hierarchical source.

3.6.11.1 XML_Map transform data inputs and outputs

The data source and data targets for the XML_Map transform differ based on the mode you choose.

Data inputs

- Normal mode: One or more data sets. Each data set can be a hierarchical data source such as XML or IDoc, or a hierarchical output structure from a previous transform. A data set can also be row-based data such as a database table, spreadsheet, or flat file.
- Batch mode: Single data set accepted. The data set can only be a hierarchical data source such as XML or IDoc, or a hierarchical output structure from a previous transform. A data set can also be row-based data such as a database table, spreadsheet, or flat file.

Data outputs

- Normal mode: A single data set. The data set may be a hierarchical structure or row-based flat data.
- Batch mode: A single data set. The data set is always a hierarchical structure or flat dataset. The Batch key columns become first-level columns of the row. The rest of the columns become second-level columns and cannot be edited.

3.6.11.2 XML_Map transform batch mode operation

In batch mode, the XML_Map transform processes data in blocks of rows before it sends the data to the next object in the data flow.

To configure the XML_Map transform in batch mode, you define how the transform forms the blocks of rows. Set the number of rows per block by maximum batch size. Additionally, add a second parameter to set the block using specified batch key columns. The transform forms the block for the first nested level based on your settings.

If you do not specify batch key columns, the transform constructs each batch based on the set maximum batch size. However, when you specify batch key columns, the transform first sorts the input data based on the key columns. The transform performs sorting to detect batch key changes.

The transform sends out each batch when it reaches the maximum batch size. If you also set the batch key columns, the transform sends out each batch when there are batch key changes.

3.6.11.2.1 XML_Map transform batch options

Use the XML_Map transform editor in batch mode to specify the *Schema In*, *Schema Out*, and batch mode options.

Use batch mode to accumulate blocks of rows before you send those rows to the next transform. Accumulating blocks of rows in this manner with SAP functions may improve performance.

Group the data by *Batch size* and optionally by the *Batch key columns* in batch mode.

Batch mode option descriptions

Options	Description
Batch size	Specifies the maximum size of second-level rows for each batch. Enter either a positive number, a global variable, or a substitution parameter. The default value of batch size is 10000.
Batch key columns	Specifies the input columns on which a given batch is constructed. Batch key columns become root-level columns of the output schema. Populate the *Batch key columns* pane by dragging columns from the *Schema In* pane to the *Batch key columns* pane. After you add an input column, the transform enables the checkbox option *Input already sorted by batch key columns*.
Input already sorted by batch key columns	Indicates that input data is sorted. The transform does not need to sort the input data. **i Note** If the *Batch key column* is not specified, the *Batch size* value defines the size of the block of rows.

3.6.11.2.2 Using Batch mode in XML_Map transform

Set up the XML_Map transform in batch mode using both *Batch size* and optionally, *Batch key columns* settings.

> **i Note**
>
> The transform allows only one input source for the XML_Map transform in batch mode.

1. In the XML_Map transform editor, select *Batch Mode*.

> **i Note**
>
> The XML_Map transform icon changes to this image  to indicate that the XML_Map transform is in batch mode.

2. Select the maximum *Batch size* value of second level rows for each batch.

> **i Note**
>
> Default value is *10000*. Value can be a positive number, a global variable or a substitution parameter.

3. Optional. For *Batch key columns*, drag the selected column name from the *Schema In* pane to the *Batch key columns* list box.

> **i Note**
>
> - The whole input schema moves one level down in the output schema, making it nested.
> - Batch key columns become part of the output schema at the top-level. However, if you want to modify the output schema, then add downstream transforms.
> - If you do not select the *Batch key columns*, the transform does not perform a sort operation. The transform constructs each batch based on the *Batch Size* setting. There are no first-level columns.

4. Select the checkbox *Input already sorted by batch key columns* if you need to indicate that the input is sorted.

The transform sends the batch when it reaches the maximum batch size, and optionally if the batch key changes.

3.6.11.3 XML_Map transform normal mode operation

In normal mode, the transform processes data on a row by row basis before sending it to the next object in the data flow.

The mapping configuration area of the XML_Map editor contains several tabs where you enter information to specify the data you want retrieved. The transform flags the tabs that contain entries with 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, the settings in the *Mapping* tab may be sufficient. For more complex mappings, complete the other tabs as appropriate.

XML_Map Editor tab descriptions

Tab	Description
Mapping	Specifies how the software maps or derives the selected output column. Contains a *Schema Remapping* button to replace an obsolete schema with a correct schema.

Tab	Description
Iteration Rule	Specifies how the transform creates instances for the current schema from instances of the source or sources. Create an iteration rule only for a repeatable target schema. Additionally, in most situations, a repeatable target schema has an iteration rule.

WHERE	Specifies conditions that determine the rows to output.
	
	Use the *Functions* and Smart Editor *Ellipses* buttons to build expressions.
	
	Enter the conditions in SQL syntax, like a `WHERE` clause in a SQL Select statement.
	
	Example
	
	`TABLE1.EMPNO = TABLE2.EMPNO`
	`AND TABLE1.EMPNO > 1000`
	`OR TABLE2.EMPNO < 9000`

DISTINCT	Specifies the list of distinct columns from the input or output schema, if required. **Restriction** You cannot mix input and output columns in the DISTINCT list.
GROUP BY	Specifies how the transform combines the output rows.
ORDER BY	Specifies how the transform sorts the output rows.
Advanced	Specifies whether the transform runs in a separate process, and specifies additional options for input schemas. Options that you set in the *Advanced* tab apply to the entire XML_Map transform.
Find	Uses search criteria to find specific words, terms, or other objects.

Note

Use the *WHERE* through *ORDER BY* tabs to specify additional constraints for the current schema, similar to SQL Select statement clauses.

XML_Map transform normal mode Mapping tab [page 1010]
Use the *Mapping* tab to create valid expressions that specify how the transform derives or maps the selected output column.

XML_Map transform normal mode Iteration Rule tab [page 1012]
Use the *Iteration Rule* tab to define how the transform calculates the output data set for the selected output schema.

XML_Map transform WHERE tab [page 1016]
Create a `WHERE` clause for any target schema in the output structure.

XML_Map transform DISTINCT tab [page 1016]
Use the *DISTINCT* tab to specify the input or output schema columns that the transform uses to determine whether a row is distinct.

XML_Map transform GROUP BY tab [page 1017]
Use the *GROUP BY* tab to specify a list of columns for which you want to combine output.

XML_Map transform ORDER BY tab [page 1018]
Use the *ORDER BY* tab to specify the columns you want the transform to use to sort the output data set.

XML_Map transform Advanced tab [page 1019]
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 or schemas.

XML_Map transform Find tab [page 1020]
Use the Find tab to search for a specific word or term in the input schema or the output schema.

3.6.11.3.1 XML_Map transform normal mode Mapping tab

Use the *Mapping* tab to create valid expressions that specify how the transform derives or maps the selected output column.

Mapping expressions contain table columns and functions among other elements. Form mapping expressions using the following methods:

- Type input column names in the work area.
- Drag columns from the *Schema In* pane and drop them in the work area.
- Type functions in the work area.
- Use the smart editor or the function wizard.

Remapping

The transform can remap the input schema if you change the source in your data flow.

> **Example**
>
> Your data flow contains a source object connected to the XML_Map transform and a target object. The software automatically maps the input source. Then you change the data flow. You place a Validation transform between the source and the XML_Map transform so that only data with a specific format passes to the XML_Map transform.

When you change the data flow so the source data changes, the current input schema mapping is no longer correct. However, the software checks the existing top-level mappings, and remaps the input schema when it finds changes.

> **Example**
>
> The new input contains a column named Row_ID in the New_Source table. The current schema has the same column named Row_ID, but it is in the Old_Source table. The software remaps the input schema by replacing the name Old_Source with the name New_Source, and keeps the column named Row_ID under New_Source.

Specifically, the software 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, either remove the column or remap it from the original source.

The software does not automatically remap the input schema under the following situations:

- When you connect a new source to the XML_Map transform before you disconnect the old source. In this situation, remap by clicking the *Schema Remapping* button in the *Mapping* tab. The software updates 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 change the source to a similar nested schema, or add or delete a transform before the XML_Map transform. In this situation, remap by clicking the *Schema Remapping* button in the *Mapping* tab to update the mapping input schema name.

Schema to schema mapping

Map a source schema to a target schema by making the target schema current and entering the source schema path in the *Mapping* tab. The software assumes that 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. However, 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 transform enables the *Merge To* operation. 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, the software displays only the original target schema.

Using XML_Map Schema Remapping [page 1012]

If you did not remap the input schema in the XML_Map transform, use the Schema Remapping feature to remap the input schema.

3.6.11.3.1.1 Using XML_Map Schema Remapping

If you did not remap the input schema in the XML_Map transform, use the Schema Remapping feature to remap the input schema.

1. In the *Mapping* tab, click the *Schema Remapping* button. The *Replace Obsolete Schema window* opens.
2. In the *Specify obsolete schema* dropdown list, choose the source schema that you disconnected from the XML_Map transform.
   This list displays only the top-level input schema. For an obsolete nested schema, enter the name of the top-level schema.
3. In the *Choose correct schema* list, choose the output schema of the transform that you added between the source and XML_Map transform.
4. Click *Remap*.
   A message displays the number of columns that were remapped; for example:

   ```
   Schema "ODS_SALESORDER" was replaced by schema "Validation_Pass" in 11 column
   names.
   ```

3.6.11.3.2 XML_Map transform normal mode Iteration Rule tab

Use the *Iteration Rule* tab to define how the transform calculates the output data set for the selected output schema.

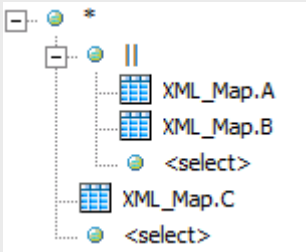
Associate an iteration rule with a repeatable target node only. An iteration rule defines how to construct the instances of the target schema from the source data. Use an iteration rule as a mechanism to specify the input data sets and how the software joins them to create the target data set.

Data Services supports the following types of joins in the iteration rule:

- Inner join
- Left outer join
- Cross join
- Parallel join (not a standard SQL join)

The *Iteration Rule* tab contains a hierachical tree that represents a logical combination of operations and input schemas that form a rule. The tab displays each operation in the rule as a node, which may contain other operations or input schemas as children.

> **Example**
>
> A rule that performs a parallel operation on example tables A and B, and then combines that output set with table C by using a cross operation might logically look like this: `(A || B) * C`
>
> In the iteration rule tab, this same rule might look like this:



XML_Map transform iteration rule construction [page 1013]
Construct iteration rules in the *Iteration Rule* tab for each repeatable schema in your output.

XML_Map transform automatic rule generation [page 1015]
Have the software automatically generate iteration rules.

3.6.11.3.2.1 XML_Map transform iteration rule construction

Construct iteration rules in the *Iteration Rule* tab for each repeatable schema in your output.

Add a new element to the rule by clicking the *<select>* place holder under an operation node and choosing the new operation or input schema from the dropdown list.

Remove an element from the rule by clicking the element and choosing *<delete>* from the dropdown list. If you remove an operation node, the transform removes any child operations or schemas from the rule. To change an operation type, click the operation node and choose the new operation from the dropdown list.

> **i Note**
>
> There is no limit to the number of sources that may be used in an iteration rule.

Iteration rule elements

Elements to add to rules	Notes
Inner join	Performs a SQL Inner join on sources. Create the expression for the join condition in the *On* area of the rule editor. When you create the expression, 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. Requirements: • For source column, the column path to the source schema cannot contain repeatable schemas. • For target column, the column must be scalar and descend from the parent schema of the schema in which you define the iteration rule. In addition, the column path to the target schema cannot contain 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, 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. Requirements: • For source columns, the path from the source column to the source schema cannot contain repeatable schemas. • For target columns, the column must be scalar and descend from the parent schema of the schema in which you define the iteration rule. In addition, the column path to the target schema cannot contain repeatable schemas.

Elements to add to rules	Notes
* - 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	Not a standard SQL operation. 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 are 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 contains the same number of rows of the largest source. For extra rows in the output set that contain data from only one source, the transform considers the additional columns that would contain data from the other sources as empty.
Available input schemas	

3.6.11.3.2.2 XML_Map transform automatic rule generation

Have the software automatically generate iteration rules.

Before you can have the software generate an iteration rule, create 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 automatically generated iteration rule matches your requirements. Modify the rule as needed, and add the `ON` condition expression when appropriate.

Also, propose rules recursively by clicking *Propose rule recursively* in the *Iteration Rule* tab. The software recursively moves through the target tree under the current target schema, finds all repeatable schemas, and generates the iteration rule for each repeatable schema based on the mappings under the schema.

> **i Note**
>
> Automatic rule generation is a best-guess function. For example, the software cannot know the `ON` condition, or whether to use inner join or left outer join. Use the automatic rule generation as a guide and always verify that the iteration rule that it creates fits your needs.

3.6.11.3.3 XML_Map transform WHERE tab

Create a `WHERE` clause for any target schema in the output structure.

Use the *WHERE* tab in the XML_Map transform to set conditions that determine the rows the transform outputs. Enter the conditions in SQL syntax, as you would in a `WHERE` clause in a SQL Select statement. The *WHERE* tab settings apply to the current output schema.

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.
- Click the *Functions* button to select functions.
- Use the `pushdown_sql` function to have Data Services create `WHERE` clauses dynamically based on data rather than pre specifying the clause.

> **i Note**
>
> Use the `pushdown_sql` function when the immediate input to the XML_Map transform is the table source where you want to push the `WHERE` clause.

Use source and target columns in the `WHERE` expression.

> **! Restriction**
>
> - For source column, the column path to the source schema cannot contain repeatable schemas.
> - For target column, the column must be scalar and descend from the parent schema of the schema in which you define the iteration rule. In addition, the column path to the target schema cannot contain repeatable schemas.

> **i Note**
>
> If your expression contains varchar comparisons, Data Services ignores trailing blanks in the data. For Oracle data, use the `rtrim` or `rpad` functions if the number of trailing blanks might differ on either side of the comparison.

3.6.11.3.4 XML_Map transform DISTINCT tab

Use the *DISTINCT* tab to specify the input or output schema columns that the transform uses 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 *Distinct columns* pane. The transform 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.

> **i Note**
>
> You cannot specify both source and target columns at the same time in the *DISTINCT* tab.

Requirements:

- Source columns descend from the source schemas in the current iteration rule.
- For source columns, the path from the source schema to the column cannot contain repeatable schemas.
- Target columns descend from the source schemas in the current iteration rule.
- For target columns, the source schema path to the column cannot contain repeatable schemas.

3.6.11.3.5 XML_Map transform 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, the transform 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 transform adds the column to the bottom of the list.

The transform uses the first column listed for primary grouping, the second column listed 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.

> **i Note**
>
> You can specify either source or target columns in the GROUP BY column list.
>
> When you use source columns, they must descend from the source schemas in the current iteration rule. In addtion, the path from the source schema to the column cannot have repeatable nodes.
>
> When you use target columns, they must descend from the current target schema. In addition, the path from the current target schema to the column cannot have 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.

  > **i Note**
  >
  > All columns in the output schema must be either part of the group by list or mapped to an aggregate function such as avg, count, max, min, or sum.
- Instance aggregation
  The XML_Map transform evaluates the aggregation functions for each of the items in the output data set when the following conditions are met:
  - Aggregation functions are defined under the current target schema.
  - Columns used in the aggregation functions descend from the sources in the current iteration rule.
  - Paths from the sources to the columns being used contain repeatable nodes.

  The XML_Map transform also evaluates the aggregation functions for each of the items in the output data set when the following conditions are met:
  - Aggregation functions are defined under the current target schema.
  - Columns used in the aggregation functions descend from the current target schema.
  - Paths from the current target schema to the columns being used contain repeatable nodes.

> **! Restriction**
>
> You cannot use both group and instance aggregation at the same time.

3.6.11.3.6 XML_Map transform ORDER BY tab

Use the *ORDER BY* tab to specify the columns you want the transform to use 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 transform uses the first column listed for primary sorting, the second column listed for secondary sorting, and so on. 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.

Specify either source or target columns in the *ORDER BY* tab.

When you specify source columns, descend them from the source schemas in the current iteration rule. In addition, do not include repeatable nodes in the path from the source schemas to the column.

When you use target columns, descend them from the current target schema. In addition, do not include repeatable nodes in the path from the current target schema to the column.

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 dropdown arrow.

3.6.11.3.7 XML_Map transform 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 or schemas.

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 or schemas that you use in the transform:

Option	Description
Cache column	Indicates whether the software reads the required data from the source and loads it into memory or pageable cache. Because the transform reads an inner source of a join for each row of an outer source, you might want to cache a source when it is used as an inner source in a join. Available values: • *Automatic* (default) • *Yes* • *No* Cache specified in the *Advanced* tab of the XML_Map transform editor overrides any cache specified in a source.

3.6.11.3.8 XML_Map transform Find tab

Use the Find tab to search for a specific word or term in the input schema or the output schema.

Find tab option descriptions

Option	Description
Find what	Indicates the action to perform on the term in the next text box. Enter a term or select a previous term from the drop-down list: • *Find what*: Select to find the term indicated in the next text box. • *Replace*: Select to replace the term indicated in the next text box.
Find	Click to start the search after you have set all of the applicable search criteria.
Schemas	Indicates the schema to search. Options are *Input*, *Output*, or *Both*.
Elements	Indicates the types of mappings to search. Options include such elements as primary keys, schemas, columns, all types, and so on.
Where	Indicates the properties to search within.
Match case	Indicates to search for the term as you typed it, matching the casing.

Searching with XML_Map Find tab [page 1020]
To search for a word or phrase in the input or output schema, use the options in the *Find* tab of the XML_Map transform editor.

3.6.11.3.8.1 Searching with XML_Map Find tab

To search for a word or phrase in the input or output schema, use the options in the *Find* tab of the XML_Map transform editor.

1. Open the *Find* tab in the XML_Map transform editor.
2. Enter the search term in the *Find what* text box, or select a term from a previous search from the dropdown list.
3. Choose the schema or schemas in which to search from the *Schemas* list.
4. Choose the mapping type in which to search from the *Elements* list.
5. Choose the property in which to search from the *Where* list.

> **i Note**
>
> Select to search either in one or all properties. SAP Data Services doesn't allow you to choose from two or three specific properties at a time.

6. Select the *Match case* checkbox to constrain your search to the capitalization you used in the *Find what* text box.
7. Select *Find*.
   Data Services searches the transform configuration for the term you specified within the constraints you defined.

   > **i Note**
   >
   > Data Services searches for columns loaded into memory. If columns aren't loaded into memory, expand the schema to load the columns into memory before clicking *Find* and searching for the columns.

The transform displays all matches in the table below the find constraints. When you select a table or column name, Data Services highlights the input schema, output schema, table, or column in the corresponding input or output schema area.

Initially, Data Services lists the matching columns in the order that they appear within the schemas. If you're searching both schemas, Data Services lists the matches as follows:

- **First listed:** First match found in the input schema.
- **Last listed:** Last match found in the output schema.

Resort results in ascending or descending order by selecting the applicable column heading.

3.6.11.4 Example: Nesting data with the XML_Map transform

Use the XML_Map transform to take an input schema and produce a nested output.

To illustrate how to use the XML_Map transform to nest output data, we use an example. In this example, assume that 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 department contains a list of employees and a new column that contains the number of employees in the department.

Source	Target

`Employee`
`  |-departmentID`
`  |-departmentName`
`  |-employeeName`
`  |-employedDate`
	`Company`
	`  |-department (*)`
	`  |-departmentID`
	`  |-departmentName`
	`  |-employee (*)`
	`  |-employeeName`
	`  |-employedDate`
	`  |-totalEmployees`

1. Create iteration rules for the `department` and `employee` schemas.

   Both schemas require information from the `Employee` source table, so they must iterate on the source table; 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 the transform encounters a new value in that column, it creates a new `department` instance.

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, the transform selects only rows that have the matching department ID.

4. Aggregate the number of employees in the department.

   The `employee` instances have already been created, so you can use the `employee` instances to create a mapping expression for the `totalEmployees` column:

   ```
   count(Company.department.employee)
   ```

3.6.11.5 Example: Unnesting data with the XML_Map transform

Use the XML_Map transform to unnest a nested input schema and output to a flat file.

To illustrate how to use the XML_Map transform to unnest an input schema for output to a flat file, we use an example. 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

`purchaseOrders`
`  |-purchaseOrder (*)`
`  |-sellerParty`
`  |-sellerID`
`  |-sellerName`
`  |-buyerParty`
`  |-buyerID`
`  |-buyerName`
`  |-orderLine`
`  |-item (*)`
`  |-name`
`  |-quantity`
`  |-unitPrice`
`  |-currency`
	`item`
	`  |-sellerID`
	`  |-buyerID`
	`  |-itemName`
	`  |-totalSales`
	`  |-currency`

1. Create an iteration rule for the `item` output schema.

   Because the output schema requires data from columns in multiple nested schemas, use a cross (*) operation to flatten the data.

   ```
   *
   |-purchaseOrders.purchaseOrder
   ```

```
|-purchaseOrders.purchaseOrder.orderLine.item
```

The input ports are always assumed to be repeatable, which means that the software expects that multiple documents of `puchaseOrders` may come in.

2. Sort the output set by the total sales for each item.
   a. Calculate the total sales amount for each item.

      Use information from the source columns in an expression in the mapping for the `totalSales` column:

      ```
      orderLine.item.quantity * orderLine.item.unitPrice
      ```

   b. Order the output set.

      Include the `item.totalSales` target column in the ORDER BY tab for the `item` output schema.

3.6.11.6 Example: Transform hierarchical source to hierarchical target with XML_Map

Use the XML_Map transform to produce a hierarchical target that is different from the hierarchical source.

To help illustrate how to use the XML_Map transform for this transformation, we use an example. 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

`catalog`
`  |-book`
`  |-Name`
`  |-Price`
`  |-Quantity`
`  |-Author`
`  |-firstName`
`  |-lastName`
`  |-Author_nt_1`
`  |-street`
`  |-city`
`  |-state`
`  |-zip`
	`authors`
	`  |-Author`
	`  |-Name`
	`  |-firstName`
	`  |-lastName`
	`  |-Address`
	`  |-Author_nt_1`
	`  |-street`
	`  |-city`
	`  |-state`
	`  |-zip`
	`  |-book`
	`  |-Name`
	`  |-Price`
	`  |-Quantity`
	`  |-totalSales`

1. Create an iteration rule for the `Author` output schema.

   Because the output schema requires data from columns in multiple nested schemas, use a cross (*) operation to flatten the data.

   ```
   *
   |-catalog.book
   |-catalog.book.Author
   ```

2. Create a new Author instance for each individual author, based on the first and last name of the author.

   Include the `authors.Author.Name.firstName` and `authors.Author.Name.lastName` target columns in the DISTINCT tab for the `Author` target schema. Each time the transform encounters a new combination of the values in those columns, it creates a new `Author` instance.

3. Sort the Author instances by the name of each author.

   Include the `authors.Author.Name.firstName` and `authors.Author.Name.lastName` target columns in the ORDER BY tab for the `Author` target schema.

4. Because there may be more than one line for the street portion of the author address, create an iteration rule for the `Author_nt_1` output schema.

```
catalog.book.Author.Author_nt_1
```

5. Map the author information from the source schema to the target schema.

   Include the appropriate source column or schema in the Mapping tab for each output column or schema:

Target column or schema	Mapping expression
authors.Author.Name.firstName	`catalog.book.Author.firstName`
authors.Author.Name.lastName	`catalog.book.Author.lastName`
authors.Author.Address.Author_nt_1	`catalog.book.Author.Author_nt_1`
authors.Author.Address.Author_nt_1.street	`catalog.book.Author.Author_nt_1.street`
authors.Author.Address.city	`catalog.book.Author.city`
authors.Author.Address.state	`catalog.book.Author.state`
authors.Author.Address.zip	`catalog.book.Author.zip`

6. Create an iteration rule for the `book` output schema.

   Like the `Author` output schema, the `book` output schema requires a combination of the `catalog.book` and `catalog.book.Author` source schemas. Use a cross (*) operation to flatten the data.

```
*
|-catalog.book
|-catalog.book.Author
```

7. Identify and create a `book` instance for each book that belongs to the author.

   Because the first and last names of the author of the book are already known, you can use them in an expression in the WHERE tab for the `book` schema to include only the correct books:

```
catalog.book.Author.firstName = authors.Author.Name.firstName AND
catalog.book.Author.lastName = authors.Author.Name.lastName
```

8. Map book information from the source schema to the target schema.

   Include the appropriate source column in the Mapping tab for each output column:

Target column	Mapping expression
authors.Author.book.Name	`catalog.book.Name`
authors.Author.book.Price	`catalog.book.Price`

Target column	Mapping expression
authors.Author.book.Quantity	`catalog.book.Quantity`

9. Calculate the total sales for each book.

   Use information from the source columns in an expression in the mapping for the totalSales column:

   ```
   catalog.book.Price * catalog.book.Quantity
   ```

3.7 Text Data Processing transforms

Use the Text Data Processing transforms to help you extract specific information from text.

These transforms can parse large volumes of text. This parsing enables you to identify and extract entities and facts such as customers, products, locations, and financial information relevant to your organization.

Parent topic: Transforms [page 240]

Related Information

Transform reference [page 240]
Dynamic transform settings [page 244]
Embedded help for transform editors [page 249]
Data Integrator transforms [page 250]
Data Quality transforms [page 338]
Platform transforms [page 840]

3.7.1 Entity Extraction transform

The Entity Extraction transform extracts information from unstructured data and creates structured data that you can use with various business intelligence tools.

Entity Extraction transform information

Characteristic	Description
	Entity Extraction transform icon

Characteristic	Description
Uses	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. The transform can extract entities and facts from text. Use the Entity Extraction transform when you have text with specific information to extract and then use with downstream analytics and applications. **Feature** / **Benefit** Searches for: • Patterns • Activities • Events • Relationships between entities — Resulting data helps determine the type of information data contains. Use for business purposes including document and content management, data integration, business intelligence, and so on. Understands word semantics Discovers new entities — Goes beyond character matching for information retrieval. Use new entities to customize extraction by creating custom entity dictionary. Automates key information extraction — Reduces the number of records to review and manually tag. Provides insights about data that may otherwise remain hidden in the text. Use the Entity Extraction transform with other transforms in the same data flow to perform various analytics on unstructured information. **Example** Your goal is to: • Extract names and addresses embedded in unstructured text. • Validate the names and addresses before running analytics.

Characteristic	Description
	Connect the source to an Entity Extraction transform. The Entity Extraction transform processes text and extracts various entities. Pass results to a Case transform. Use the Case transform to identify which rows represent names and which rows represent addresses. Pass results to a Data Cleanse transform. Use the Data Cleanse transform to standardize the extracted names. Pass results to a Global Address Cleanse transform. Use the Global Address Cleanse transform to validate and correct the extracted address data.
Content objects	Sample transform

Entity Extraction transform options [page 1027]
Set options in the Entity Extraction transform for language, parsing rules, and custom dictionaries.

Entity Extraction transform filtering options [page 1031]
Filtering options enable you to limit extracted entities to specific entities defined in a dictionary, system file, rule, or a combination of all methods.

Entity Extraction transform input fields [page 1035]
Use Data Services recognized input fields when you map the input schema in the Entity Extraction transform.

Entity Extraction transform output fields [page 1036]
Use Data Services recognized output fields in the output schema set up to output information about extracted data.

Examples for using the Entity Extraction transform [page 1040]
To understand how to use the Entity Extraction transform, view the following examples.

Entity Extraction transform extraction dictionary [page 1042]
An extraction dictionary is a user-defined repository that stores customized information about the entities in your data.

Entity Extraction transform extraction rule [page 1043]
Use an extraction rule to define custom patterns to extract entities, relationships, events, and larger extractions such as facts.

3.7.1.1 Entity Extraction transform options

Set options in the Entity Extraction transform for language, parsing rules, and custom dictionaries.

For complete information about working with dictionary and rule files, and for custumizing data extraction, see the following guides:

- *Text Data Processing Extraction Customization Guide*
- *Text Data Processing Language Reference Guide*

Entity Extraction option descriptions

Option	Description
Languages	
Language	Specifies the language for processing your content. Options include Auto and an alphabetical list of countries. • *Auto*: If the transform cannot identify the language, it uses the setting of the Default Language option. *Auto* is the default setting. > **i Note** > If input content is in XML, and organized in sections, the transform identifies the language of each section. • Specific language: When you select a specific language from the dropdown list, the transform extracts only the entities defined for that language for the entire document. When you select Auto, the transform uses only the specified dictionaries and rule files that do not specify a language. However, the transform applies the dictionaries and rule files that do identify a language to input identified in that language. > **i Note** > You cannot run extraction unless you have a language directory that contains the files for at least one language. By default, the installer installs the language directory on a client as well as a server during installation. The default location for the language directory is: `<LINK_DIR>/TextAnalysis/languages`. Optionally use a substitution variable for this option for convenience.
Default Language	Specifies a default language to use when you set the *Language* option to *Auto* and the transform cannot identify a language. • *None*: Indicates that there is not default language. If the software cannot identify a language, the transform issues a processing error, but continues processing. • Specific language: Indicates the selected language to use as the default language. If the software cannot identify a language on first attempt, the transform extracts only the entities defined for the language you chose for *Language*. Optionally use a substitution variable for this option for convenience.

Option	Description
Filter By Entity Types	Optional. Specifies a list of entity types that the selected language supports to use for filtering the extraction output. Clicking the *Ellipses* button opens the *Ordered Options Window - [Filter By Entity Types - Option]* dialog. **i Note** When you set the *Language* option to *Auto*, the software populates the list of Available entity values in the *Ordered Options Window* dialog box with the combined entity types for all installed languages. After you add the applicable entity types to the selected values list, click *OK* to close the *Ordered Options Window* dialog box. **i Note** Available entities depend on the language that you select.
Processing Options	
Dictionary Only	Specifies whether the software limits the extraction process to entities that you define in the specified dictionaries. • *Yes*: Limits the extraction process to the entities that you define in the specified dictionaries. If you select *Yes*, also specify a dictionary in the *Dictionaries* group. • *No*: Does not limit the extraction process to the entities that you define in the specified dictionaries. **i Note** If you select *Yes*, the extraction output does not include any predefined entities. If you select *Yes* and you also set options in the *Rules* group, the transform extracts entities and facts defined in the rules along with entities from the specified dictionaries.
Linguistic Markup	Specifies what the transform returns. • *ENTITY_EXTRACTION*: Returns entities only. *ENTITY_EXTRACTION* is the default. • *POS_AND_STEM*: Returns parts of speech and stems. • *ENTITY_POS_AND_STEM*: Returns entities, parts of speech, and stems. For more information about parts of speech tagging and stemming, see the *Text Data Processing Language Reference Guide*.
Advanced Parsing	Specifies whether the transform produces advanced parsing information during extraction. Applicable for English language only. • *Yes*: Enables advanced parsing. Transform issues an error if you do not also select *English* for *Language*. • *No*: Disables advanced parsing. Advanced parsing enriches linguistic processing including richer noun phrase structure, noun phrase coordination, and syntactic function attributes that can be leveraged in custom rules.

Option	Description
Processing Timeout	Specifies the amount of time before the transform stops processing the file content. • *60*: Transform stops processing file content after 60 seconds. *60* is the default setting. • Other value: Enter a number for the number of seconds. Enter -1 to disable processing timeout.
Document Properties	Specifies whether the transform extracts document properties of a binary document when present. • *Yes*: If there are binary documents present, extracts document properties. • *No*: If there are binary documents present, does not extract document properties. *No* is the default. Document properties are name-value pairs. The Entity Extraction transform extracts only the following document properties for binary documents: • APP_NAME: The name of the software that created the document • APP_VERSION: The version of the software that created the document • AUTHOR: The name of the person who created the document • COMPANY: The name of the company that owns the document • TITLE: The title of the document • DATE_CREATED: The date on which the document was created If document properties are available, the transform extracts them as entities. Define the following output fields to use document properties (DOC_PROPERTY): • `<ID>`: The entity ID of the document property • `<SOURCE>`: DOC_PROPERTY • `<TYPE>`: The name of the document property • `<STANDARD_FORM>` or `<SOURCE_FORM>`: The value of the document property • `<CONVERTED_TEXT>`: The textual content of the binary document The other output columns are not applicable to `<DOC_PROPERTY>` extraction rows, and have a value of `-1`.
Dictionaries	
Dictionary	Group of options to specify a dictionary file the transform uses during extraction. Duplicate the *Dictionary* group for each dictionary file to use.
Dictionary file	Specifies a valid, compiled dictionary file to use for data extraction. **i Note** The Job Serve must be able to access the dictionary files. For dictionary files on remote computers, ensure that the job server can resolve the path. For convenience, use the substitution variable for this option. However, if you select the entity types through the *Filter by Entity Types* option, the transform populates the Available entity list in the *Ordered Options Window* dialog box as if you did not specify a dictionary file.

Option	Description
Filter By Entity Types	Specifies a list of entity types defined in the selected dictionary file to use for filtering the extraction output. Click the *Ellipses* button to launch the *Ordered Options Window* dialog box. Use the *Add* or *Remove* buttons to construct a list of entity types for the filter.
Rules	
Rule	Group of options to specify a rule file the transform uses during extraction. Duplicate the group for each rule file to use.
Rule File	Specifies the valid, compiled rule file to use for extraction. **i Note** The Job Server must be able to access the Rule files. For rule files on remote computers, ensure that the job server can resolve the path. **i Note** A rule file may contain multiple rules. Use the *Filter by Rule Names* option to select a specific rule in the rule file. For convenience, use a substitution variable for this option. However, if you select rules using the *Filter by Rule Names* option, the transform populates the Available value list in the *Ordered Options Window* dialog box as if you did not specify a rule file.
Filter By Rule Names	Specifies a list of rule names defined in the selected rule file to use for filtering the extraction output. Click the *Ellipses* button to launch the *Ordered Options Window* dialog box. Use the *Add* or *Remove* buttons to construct a list of rule names for the filter.

3.7.1.2 Entity Extraction transform filtering options

Filtering options enable you to limit extracted entities to specific entities defined in a dictionary, system file, rule, or a combination of all methods.

The Entity Extraction transform provides filtering options that you use individually or in combination so that you control the output that is generated by the transform.

The transform editor provides filtering options in three categories as described in the following table.

Filter option	Option group	Description
Filter By Entity Types	*Languages*	Limits extraction output to include only selected entities for this language.
Filter By Entity Types	*Dictionary*	Limits extraction output to include only entities defined in a dictionary.

Filter option	Option group	Description
Filter By Rule Names	*Rules*	Limits extraction output to include only entities and facts returned by the specific rules.

The following table describes information contained in the extraction output based on the combination of the filter options.

Languages Entity types	Dictionaries Entity types	Rules Rule names	Extraction output content	Notes
Yes	No	No	Entities, extracted using the entity types selected in the filter.	N/A
No	Yes	No	• Entities extracted using the entity types defined in the selected language • Entity types selected from the filter in the *Dictionaries* group	The transform returns entities of the specified type from all dictionaries if: • you specify multiple dictionaries • the dictionaries all contain the same entity type • you select this entity type as a filter for only one dictionary
Yes	Yes	No	• Entities extracted using the entity types • Entities defined in the filters for the selected language • Any specified dictionaries	N/A

Languages Entity types	Dictionaries Entity types	Rules Rule names	Extraction output content	Notes
No	No	Yes	• Entities extracted using the entity types • Entities defined in the selected language • Entities defined by any rule names selected in the filter from any specified rule files	The transform returns entities and facts from all rule files if: • you specify multiple rule files • the rule files all contain the same rule name • you select the rule name as a filter for only one rule
No	Yes	Yes	• Entities extracted using entity types • Entities defined in the selected language • Entity types selected from the dictionaries filter • Rule names selected in the filter from any specified rule files	N/A
Yes	No	Yes	• Entities extracted using entity types • Entities defined in the filters for the selected language • Rule names selected in the filter from any specified rule files	N/A

Languages Entity types	Dictionaries Entity types	Rules Rule names	Extraction output content	Notes
Yes	Yes	Yes	• Entities extracted using entity types • Entities defined in the filters for the selected language • Entity types selected from the dictionaries filter • Rule names selected in the filter from any specified rule files	The transform extracts the output using filters. The filter is based on the union of extracted entities or facts for the selected language, dictionaries, and rule files.

Special circumstances

If you change your selection for the language, dictionaries, or rules, clear existing filters that are associated with the option by clicking the *Filter by...* option. Select new filtering choices based on the changed selection.

If you set up your job in the following manner, the transform uses a filter to extract output:

- You include multiple dictionaries and rule files in the job setup
- You set filtering options only for some of the dictionary or rule files, but not all of them

The transform uses a filter of the joined data of all entity types and rule names that you defined for each file in the job setup.

If you select the *Dictionary Only* option under the *Processing Options* group, with a valid dictionary file, the transform does not include the entity types defined for the language in the extraction output, but the transform does includes any extracted rule file entities and facts in the extraction output.

3.7.1.3 Entity Extraction transform input fields

Use Data Services recognized input fields when you map the input schema in the Entity Extraction transform.

Input field descriptions

Input field name	Data type	Description
<LANGUAGE>	varchar	The language of the extraction. Use as a dynamic input field to set this field at runtime. The following table describes the transform behavior for various conditions of the input field <LANGUAGE>: **Value of input field LANGUAGE** — **Runtime transform behavior** Auto — Overrides the set value of the transform option *Language*. Valid and supported — Overrides the set value of the transform option *Language*. Valid but not supported — Transform issues error and skips processing input text. Invalid — Transform issues error and skips processing input text Not specified — Ignored
TEXT	Long, blob, or varchar	Required. 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 TXT, HTML, XML, or certain binary formats such as PDF.
TEXT_ID	Long, int, or varchar	Optional. Unique identifier used for tracing the content in case of an error. **i Note** • The transform ignores an unsupported data type during runtime. Instead, if the transform reads from an unstructured text file format, the transform uses either the file name or the string in the <TEXT> input field as the content identifier. • When a varchar or long column is mapped to the TEXT_ID input field and a value used to construct an error message contains more than 1KB, the transform truncates the value to 1KB.

3.7.1.4 Entity Extraction transform output fields

Use Data Services recognized output fields in the output schema set up to output information about extracted data.

Output field descriptions

Output field name	Data Type	Description
CONVERTED_TEXT	long	The content text representation in UTF-16 encoding of the input text. **i Note** When you include the CONVERTED_TEXT output field, the first entity-fact output row for any input document contains the UTF-16 textual representation of the input document. Subsequent entity-fact output rows for the input document do not contain textual representation of the input document.
ID	int	The parent-child relationship between entities and/or facts. This value is unique within the scope of the processed input text. **i Note** If you process two different input documents in the same data flow and store the output to a database, you should not use this field as a primary key.
LENGTH	int	The character length of an entity or a fact in the CONVERTED_TEXT field.
OFFSET	int	The character offset of an entity or a fact in the CONVERTED_TEXT field.
PARAGRAPH_ID	int	A unique identifier of the paragraph in the CONVERTED_TEXT field containing the entity or fact.
PARENT_ID	int	Represents a parent-child relationship between entities and/or facts. If there is a parent-child relationship, provides a link to a parent ID value. If there is not a parent-child relationship, value is -1 to indicate there is no relationship.
SENTENCE_ID	int	A unique identifier of the sentence in the CONVERTED_TEXT field containing the entity or fact.
SOURCE	varchar (10)	The origin of an entity or fact indicating how the transform determined the match: • `SYSTEM`: Used system files to make the match. • `DICTIONARY` - Used a dictionary to make the match. • `RULE` - Used an extraction rule file to make the match. • `LINGUISTX` - Used the setting in the *Linguistic Markup* option of the *Processing Options* group to make the match.
SOURCE_FORM	varchar (2000)	• The name of an entity, fact, or subfact as mentioned in the input text. • For linguistic output, the original word in the input.

Output field name	Data Type	Description
STANDARD_FORM	varchar (2000)	• The standard form of an entity, fact, or subfact. Generally it is the longest, most precise or official name associated with the value of the corresponding `TYPE` column. **i Note** The standard form and the source form for an entity are often the same. • For linguistic output, the stem of the original word.
TYPE	varchar (255)	• The type of entity or fact. If applicable, also could be subtypes or subfacts. **Example** For example, "Mr. Jones" is identified as a `PERSON` entity and "car" as a `COMMON_VEHICLE/LAND` entity subtype. **i Note** The transform uses the pipe symbol (`/`) as a separator between subtypes. • For linguistic output, the part of speech of the original word.

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

ID	PA-RENT_ID	STANDARD_FORM	TYPE	SOURCE_FORM	SOURCE
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
11	-1	He states that Green offered him $1250.00 but his car is worth anywhere from $2500 to $4500.	Sentiment	He states that Green offered him $1250.00 but his car is worth anywhere from $2500 to $4500.	RULE
12	11	car	Topic	car	RULE
13	11	worth anywhere from $2500 to $4500	WeakPositiveSentiment	worth anywhere from $2500 to $4500	RULE
14	-1	Green	PROP_MISC	Green	SYSTEM
15	-1	$1250.00	CURRENCY	$1250.00	SYSTEM
16	-1	car	COMMON_VEHICLE/LAND	car	SYSTEM
17	-1	from $2500 to $4500	CURRENCY	from $2500 to $4500	SYSTEM
18	-1	Mr. Jones	PERSON	Mr. Jones	SYSTEM
19	-1	Mr. Jones would like Green's comprehensive coverage to be in line with other competitors.	Request	Mr. Jones would like Green's comprehensive coverage to be in line with other competitors.	RULE
20	19	Mr. Jones	Topic	Mr. Jones	RULE
21	19	would like	GeneralRequest	would like	RULE
22	19	Green's comprehensive coverage	Topic	Green's comprehensive coverage	RULE
23	-1	Green	PROP_MISC	Green	SYSTEM
24	-1	other competitors	COMMON_PERSON /GROUP	other competitors	SYSTEM

ID	PA-RENT_ID	STANDARD_FORM	TYPE	SOURCE_FORM	SOURCE
25	-1	other competitors	COMMON_ ORGAN-IZATION/COMMER-CIAL	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.

Example: Extraction of linguistic information

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 when you request POS_AND_STEM information:

- ID
- SOURCE_FORM
- STANDARD_FORM
- TYPE
- SOURCE

Sample input text: "The excited children boarded the airplane and flew to California, where they will visit their grandparents and many cousins."

ID	SOURCE_FORM	STANDARD_FORM	TYPE	SOURCE
1	The	the	Det-Def	LINGUISTX
2	excited	excited	Adj	LINGUISTX
3	children	child	Nn-Pl	LINGUISTX
4	boarded	board	V-Past	LINGUISTX
5	the	the	Det-Def	LINGUISTX
6	airplane	airplane	NN-Sg	LINGUISTX
7	and	and	Conj-Coord	LINGUISTX
8	flew	fly	V-Past	LINGUISTX
9	to	to	Prep	LINGUISTX
10	California	California	Prop	LINGUISTX
11	,	,	Punct-Comma	LINGUISTX

ID	SOURCE_FORM	STANDARD_FORM	TYPE	SOURCE
12	where	where	Conj-Sub	LINGUISTX
13	they	they	Pron	LINGUISTX
14	will	will	Aux	LINGUISTX
15	visit	visit	V-Pres	LINGUISTX
16	their	their	Det-Poss	LINGUISTX
17	grandparents	grandparent	Nn-Pl	LINGUISTX
18	and	and	Conj-Coord	LINGUISTX
19	many	many	Det-Pl	LINGUISTX
20	cousins	cousin	Nn-pl	LINGUISTX
21	.	.	Punct-Sent	LINGUISTX

In the above example, row 3 shows ID as 3 (unique identifier). The SOURCE_FORM shows the original word "children" as it appeared in the sentence. The STANDARD_FORM shows "child" (the stem of the source form). The TYPE identifies the word's part of speech as a plural noun. The SOURCE shows this processing was handled by the LINGUISTX analyzer.

For more information about the linguistic extraction features, see the *Text Data Processing Language Reference Guide* under "Stemming" and "Part-of-Speech Tagging".

Related Information

XML extraction and parsing for columns

3.7.1.5 Examples for using the Entity Extraction transform

To understand how to use the Entity Extraction transform, view the following examples.

The following examples use the same scenario to obtain different types of data.

Example

Scenario:

A human resources department wants to analyze resumes that they have received in various formats:

- A text file as an attachment to an e-mail
- A text resume pasted into a field on the company Web site

- Updates to resume content that the department wants to process in real time

Text file e-mail attachment

Example

The human resources department frequently receives resumes as attachments to e-mails from candidates. They store these attachments in a separate directory on a server.

To analyze and process data from these text files, they perform the following steps:

1. Configure an *Unstructured text* file format that points to the directory that contains the resumes.
2. Build a data flow that contains the following objects: The unstructured text file format as the source, an Entity Extraction transform, and a target.
3. Configure the Entity Extraction transform to process and analyze the text.

Text resume pasted into a field on a Web site

Example

The human resources department has an online job application form that includes a field into which applicants can paste their resumes. They then capture the field in a database table column.

To analyze and process data from the database:

1. Configure a connection to the database by creating a database datastore.
2. Build a data flow that contains the following objects: The database table as the source, an Entity Extraction transform, and a target.
3. Configure the Entity Extraction transform to process and analyze the text.

Updated content to be processed in real time

Example

The human resources department is seeking a particular qualification in an applicant. When an applicant updates their resume on the company Web-based form with the desired qualification, the HR manager wants to be immediately notified. Use a real-time job to enable this functionality.

To analyze and process the data in real time:

1. Add a real-time job to the data flow that includes begin and end markers.
2. Add the following objects to the data flow: A message source, an Entity Extraction transform, and a message target.

3. Configure the transform to process and analyze the text.

Related Information

Database datastores
Real-time Jobs
Unstructured file formats

3.7.1.6 Entity Extraction transform extraction dictionary

An extraction dictionary is a user-defined repository that stores customized information about the entities in your data.

For example, the extraction dictionary can store the following:

- Entities composed of common words that have special meaning in your domain
- Alphanumeric sequences such as specialized vocabulary or part numbers

Use an entity dictionary to store name variations in a structured way. The transform accesses the entity dictionary through the extraction process. A dictionary structure can help standardize references to an entity.

A language-independent dictionary doesn't have a language specified in the dictionary name. Use a language-independent dictionary to store all your entities and patterns. When you use the dictionary for data of different languages, you apply the same entity and patterns to each data set for each language. For example, use a language-independent dictionary for the following purposes:

Purpose	Example
Name variation management	Such as recognizing "Bimmer" as a colloquial reference to the car make BMW.
Disambiguation of unknown entities	Such as extracting DF-21D, DF 21D, DF21D, and df-21d as alternative forms of the Dong Feng 21D rocket.
Control over entity recognition	Such as forcing Text Analysis to extract Dong Feng 21D as an entity of type WEAPON instead of extracting Dong Feng as a PERSON.

For more information about dictionaries and how to create them, see the *Text Data Processing Extraction Customization Guide*.

3.7.1.7 Entity Extraction transform extraction rule

Use an extraction rule to define custom patterns to extract entities, relationships, events, and larger extractions such as facts.

You write custom extraction rules using a pattern-based language. A pattern-based language enables you to perform pattern matching using character or token-based regular expressions combined with linguistic attributes to define custom entity types. For example, create extraction rules for the following purposes:

- Extract complex facts based on relations between entities and predicates such as verbs or adjectives.
- Extract entities from new styles and formats of written communication.
- Associate entities such as times, dates, and locations, with other entities, otherwise known as entity-to-entity relations.
- Identify entities in unusual or industry-specific language. For example, use of the word crash in computer software versus insurance statistics.
- Capture facts expressed in new, popular vernacular. For example, recognizing sick, epic, and fly as slang terms meaning good.

For more information about using extraction rules, see the *Text Data Processing Extraction Customization Guide*.

4 Data Types

SAP Data Services data types are internal storage formats that store values.

A data type implies a default format for displaying and entering values. For example, you enter values in expressions, which are a combination of constants, operators, functions, and variables. Expressions evaluate to a value of a given data type.

Data Services supports specific data types. When you load source data to Data Services, the data types reflect the associated database management system. The DBMS data types may not match with Data Services data types. Therefore, to properly process data, Data Services converts incoming data types to its internal data types. It can then recognize the data types and know how to process the data. When a data flow outputs generated data to a target, Data Services converts the data types back into the applicable target DBMS data types.

- Data Services converts data read from sources to the appropriate Data Services data types.
- Data Services converts data loaded to targets from SAP Data Services data types to types appropriate for the target.

All data types that Data Services recognizes allow Null values unless otherwise noted.

Related Information

Data type conversion

4.1 blob

The blob data type stores binary data.

SAP Data Services uses blob data types for multimedia data such as images, audio, and video. The software stores blob columns in the file system during the data flow execution. You cannot use blob columns in comparisons, calculations, or data type conversions.

The following are Data Services limitations for blob data types:

- No data type conversions between blob and any other data type.
- Blob data cannot be stored in long data type.
- No support for the NULL indicator for blob data.
- Designer *Data Preview* pane cannot display blob data.
- Designer *View Data* utility shows data for a blob column as `<blob>`.

> **i Note**
>
> Blob and long data types share many of the same limitations.

Related Information

Conversion to or from internal data types
long [page 1053]

4.1.1 File format considerations for blob

Define blob data type columns in SAP Data Services fixed-width files, delimited files, XML files, and XML messages.

Define an unlimited number of blob columns in a file format. The blob columns may appear in any order in a file format.

In fixed-width file formats, the software sizes blob columns in bytes and not characters. The following minimum and maximum field sizes apply:

- 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. By "inline" we mean that the data itself appears at the location where a specific column is expected.

Example

Consider a fixed-width file format with the following column setup:

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.

EmployeeNo	LastName	FirstName	Password
7369	Washington	George	??????????

EmployeeNo	LastName	FirstName	Password
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.

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>>
```

Use blob data types in sources and targets and pass them through transforms. Data Services handles blob data based on the particular blob characteristic:

- When Data Services reads blob data from fixed-width flat files, it does not trim the leading or trailing 0x00 bytes.
- Data Services does not treat all 0x00s as Null, but stores them as is.
- If the size of the input blob data is not equal to the target blob field size, Data Services issues an error when it loads a blob into a fixed-width flat file.
- If the input blob consists of only 0x00s or is a NULL value, Data Services loads all 0x00s up to the field size of the target blob column.

Parent topic: blob [page 1044]

Related Information

Database considerations for blob [page 1046]
Limitations for long and blob data types [page 1047]

4.1.2 Database considerations for blob

Select the correct bulk loader when you use blob with certain databases.

SAP Data Services can load blob columns to the following databases:

- Databases that support parameterized loading.
- Databases with API-based bulk loaders and not file-based bulk loaders. The following table lists the database types with API-based bulk loaders.

API-based bulk loader database types

Database	Bulk loader
Oracle	API
DB2	CLI load
Microsoft SQL Server	Bulk load
SAP ASE	Bulk load

Parent topic: blob [page 1044]

Related Information

File format considerations for blob [page 1045]
Limitations for long and blob data types [page 1047]

4.1.3 Limitations for long and blob data types

Long and blob data types share many of the same limitations.

You cannot use long or blob data types in the following situations:

- In join, key, compare, or pivot columns
- In SQL functions such as `substr`
- In expressions and conditions
- In SELECT lists of queries containing GROUP BY clauses
- In SELECT lists of queries with the *Distinct Rows* option enabled
- In GROUP BY, ORDER BY, or WHERE clauses
- In input or output parameters or return types of functions such as lookup
- With variable data types
- In work flow and data flow input and output parameters
- With debug filters

The following table shows some of these limitations by transform.

Transform	Do not use long or blob column data in
Case	Case condition
Hierarchy_Flattening	Parent or child columns

Transform	Do not use long or blob column data in
History_Preserving	Compare columns
Pivot	Pivot columns
Table_Comparison	Primary key columns or compare column
Query	WHERE, GROUP BY, ORDER BY, or DISTINCT

Data Services 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 runtime to indicate that it has disabled the auto-correct load option.

> **i Note**
>
> To use large object data types with Informix datastores, first configure the Informix ODBC options. For more information, refer to the Informix datastore options.

Parent topic: blob [page 1044]

Related Information

File format considerations for blob [page 1045]
Database considerations for blob [page 1046]

4.2 date

The date data type defines calendar dates.

SAP Data Services automatically converts incoming date values to its internal data type, and converts outgoing data back to the external database data type. 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 using the codes and literal strings listed in the following table.

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

Date format code	Description	Example
MONTH	Full name of the month	The first month of the year: `JANUARY`
MON	Abbreviated, three-character name of month	The first month of the year: `JAN`
YY	2-digit year The software interprets a `YY` value less than 15 as being 20 `<yy>`. For example, the software interprets 10 as being the year 2010. The software interprets a `YY` value greater than or equal to 15 as being 19 `<yy>`. For example, the software interprets 35 interpreted as being the year 1935.	The year 1998: `98`
YYYY	4-digit year	The year 1999: `1999`

Changing value for determining 2-digit year dates [page 1049]
Change the value that the software uses to interpret 2-digit year dates.

Examples: Operations on date data types [page 1050]
Perform operations on dates such as add and subtract date, datetime, interval, and time values.

Related Information

Date arithmetic

4.2.1 Changing value for determining 2-digit year dates

Change the value that the software uses to interpret 2-digit year dates.

1. In Designer, select *Tools* > *Options*.
2. In the *Options* dialog box, expand Data and select General.
3. Enter a new value in *Century Change Year*.

   Ensure that the value you enter is a positive integer between 0 and 99.

Task overview: date [page 1048]

Related Information

Examples: Operations on date data types [page 1050]

4.2.2 Examples: Operations on date data types

Perform 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.

Example

In this example, the value of the variable `MyDate` is the first day of 1996.

> **Note**
>
> The case that you use when you set functions affects resulting values:
>
> - If you use lower case to type “month” or “mon”, the resulting value for `to_char` is lower case. For example, “january” or “jan”.
> - If you use upper case to type “MONTH” or “MON”, the resulting value for `to_char` is upper case. For example “JANUARY” or “JAN”.

Examples of date formats and functions

Example	Output
`to_char($MyDate, 'YYYY.MM.DD')`	1996.01.01
`to_char($MyDate, 'MONTH DD, YYYY')`	JANUARY 01, 1996
`to_char($MyDate, 'DD/MM/YY')`	01/01/96
`to_date('01/01/96','DD/MM/YY')`	1996.01.01 stored as a date
`to_date('01/01/19','DD/MM/YY')`	2019.01.01 stored as a date

Parent topic: date [page 1048]

Related Information

Changing value for determining 2-digit year dates [page 1049]

4.3 datetime

The datetime data type defines calendar dates and times.

SAP Data Services manages date operations in the format used by your database management system. 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 the software converts a date field to a datetime value, the default time added to the value is 00:00:00. If the software converts a time to a datetime value, the default date added to the value is 0000.01.01.

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

When you convert datetime values to strings, choose the sections of the value not to convert by excluding them from the format description.

> **Example**
>
> To convert a datetime value to a string that contains only the time, specify the function parameters as follows:
>
> ```
> to_char($MyDateTime, 'hh24:mi:ss.ff')
> ```

Oracle has a default datetime format that includes only date values. Therefore, if you load datetime data from SAP Data Services into a char field in an Oracle table, Oracle puts the data in its default datetime format, and excludes the time value.

Related Information

Date arithmetic

4.4 decimal

The decimal data type defines exact decimal numbers.

When you specify a decimal data type in Data Services, you indicate the following characteristics of the type:

- Precision: Total number of digits in the value.
- Scale: Number of digits to the right of the decimal point.

In the software, the following relations must hold for precision and scale:

1 <= precision <= 96
0 <= scale <= precision

The decimal value can have a plus or minus sign indicating a positive or negative value. The sign can appear before or after the value with any number of blanks between the value and the sign. When there is no plus or minus sign, the software considers the decimal value as positive. The sign does not count as part of precision.

The software permits leading zeros in the integer digit, and trailing zeros in the fraction part.

When input is more precise than the data type of the column or variable in which it is stored, the software rounds the input. The software issues a runtime error when the input is out of range. For example, an absolute value is too large.

> **i Note**
>
> The software uses a maximum of 28 precision. It does not enforce precision, meaning that there is no error when you have a larger number. Instead, the software rounds any number more than 28.

The software considers the decimal data type and the numeric data type as identical.

> **i Note**
>
> When you import a table from an Oracle datastore and the native column data type is NUMBER (without any specific precision and scale) the software imports the column as Decimal (28,7) by default. You can override the default Precision and Scale values for an Oracle table at the database level by entering the values in the *Advanced* section of the New Datastore editor.

Related Information

Oracle datastore

4.5 double

The double data type defines an 8-byte floating point value.

The 8-byte floating point value has radix, exponent range, and precision of the platform on which SAP Data Services is running.

4.6 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. The software considers values without a plus or minus sign as positive values.

4.7 interval

The interval data type defines differences between dates or times.

The value is in days unless you specify another unit, such as in a conversion function.

SAP Data Services provides conversion functions to make interval values available: `interval_to_char` and `num_to_interval`.

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

Related Information

Date arithmetic

4.8 long

The long data type stores character data of variable length up to 2 gigabytes in length.

SAP Data Services uses long to represent character-based large objects (clob). Data Services converts several other database-specific large object types, such as longvarchar and text, to the long data type. The Data Services long data type supports mapping from all databases.

The software stores long columns either in memory or in the file system during the data flow execution depending on the size of the long value.

You cannot use long columns in comparisons, calculations, or data type conversions, except for `long_to_varchar` and `varchar_to_long`.

Long and blob data types share some of the same limitations.

Limitations for long [page 1054]
There are limitations that you should be aware of when you use long data types.

File format considerations for long [page 1054]
Define long data type columns in SAP Data Services delimited files, XML files, and XML messages.

Related Information

Limitations for long and blob data types [page 1047]

4.8.1 Limitations for long

There are limitations that you should be aware of when you use long data types.

The following limitations exist for long data types:

- The software does not convert between long and any other data types except varchar. You can convert only 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. Therefore, 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.

Parent topic: long [page 1053]

Related Information

File format considerations for long [page 1054]
Conversion to or from internal data types

4.8.2 File format considerations for long

Define long data type columns in SAP Data Services delimited files, XML files, and XML messages.

Long data can be in the file or it can reference an external file. The notation for this external file is `<<filename>>`. The software automatically generates the file name.

Example

Consider a comma-delimited file format that contains the following database columns:

Column name	Data type
ProductNo	integer
Description	long
Picture	blob

In Example 1, the long data appears in the file, but in Example 2, it references a file. The file name designates a path relative to the original input/output file or an absolute path.

Example 1:

```
7369,WidgetA transforms questionable data into trusted sources through a
single
environment.,<<pictures\WidgetA.jpg>>
```

Example 2:

```
7499,<<descriptions\WidgetC_descr.txt>>,<<C:\Widgets\pictures\WidgetC.jpg>>
```

Parent topic: long [page 1053]

Related Information

Limitations for long [page 1054]

4.9 numeric

The numeric data type is identical to the decimal data type in SAP Data Services.

Just like decimal, the numeric data type defines exact decimal numbers. When you specify a numeric data type in Data Services, indicate precision and scale. See the topic for decimal data type for more information.

4.10 real

The real data type defines a 4-byte floating point value, with radix, exponent range, and precision of the platform on which SAP Data Services is running.

The real value can have a plus or minus sign indicating a positive or negative value. The sign can appear before or after the value with any number of blanks between the value and the sign. Unsigned values are considered to be nonnegative.

Databases store real values as a 32-bit approximation of the number. Because of this approximation, comparison results are unpredictable when a real value is used in an equality or inequality comparison.

Therefore, SAP recommends 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:

- In 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.

4.11 time

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

SAP Data Services manages time operations in the format used by your database manager. Conversion operations to or from times require you to specify the format of the time value. To specify a time format, generate a string from the following codes and other literal strings or punctuation.

Time format code	Description
HH24	2-digit hour of the day (0-23)
MI	2-digit minute (0-59)
SS	2-digit second (0-59)

Example

The following examples show the use of time formats with functions. The value of the variable `MyTime` is 25 minutes after 8 in the evening.

Example	Output
`to_char($MyTime, 'HH24:MI:SS.FF')`	`20:25:00`
`to_char($MyTime, 'HH24:MI')`	`20:25`

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

Related Information

Date arithmetic

4.12 timestamp

The timestamp data type supports the Oracle 9i timestamp data type with no zone information.

The timestamp data type incorporates up to a 9-digit sub-second.

You can add or subtract timestamp values. The resulting data type from addition or subtraction operations depend on the operation and data types involved.

Related Information

Date arithmetic

4.12.1 Conversion between timestamp and character strings

Convert between timestamp and character values by using the `to_date` and `to_char` functions.

The `to_date` and `to_char` functions have a format called FF, which indicates the sub-second digits.

Example

The following example code shows valid function calls.

```
to_date ('2002.02.26 01234004 09:03:25','yyy.mm.dd ff hh24:mi:ss')
to_char (timestamp_column, 'yyyy.mm.dd hh24:mi:ss.ff')
```

Use the FF format for datetime columns to access sub-seconds. For example, a DB2 timestamp column is mapped to datetime in SAP Data Services. This column contains micro-second. You can access these sub-seconds using the FF format.

Parent topic: timestamp [page 1056]

Related Information

timestamp limitations [page 1057]

4.12.2 timestamp 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:

1. Convert the timestamp column in the select list of the SQL transform to a character format using the `to_char` function.
2. Convert it back to timestamp using the `to_date` function.

To use a timestamp column in an Oracle stored procedure:

1. Convert input and output timestamp parameters in the stored procedure to char, using the `to_char` function
2. Convert the output parameter back to timestamp 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.

Parent topic: timestamp [page 1056]

Related Information

Conversion between timestamp and character strings [page 1057]

4.13 varchar

A data type that can hold letters and numbers.

When specifying a varchar data type, indicate the following characteristic of the type:

Characteristic	Description
Length	Number of characters that the variable or column can hold. Length must be greater than zero. There is no maximum allowable value for the length.

Character strings longer than the number of characters defined for the column or variable are truncated on the right to the length of the data type. Only the required number of characters is used to store strings shorter than length.

SAP Data Services provides functions to convert values to and from strings; to join strings together, use the concatenation operator (`||`). This data type allows NULL values.

The software conforms to the ANSI SQL-92 varchar standard and treats varchar data as follows:

- Keeps trailing blanks in character values that you insert into varchar columns.
- Keeps trailing blanks when you read from sources with string data types. If you want to remove trailing blanks from your input data, you must use the rtrim or rtrim_blanks function.
- Ignores trailing blanks when you compare varchar data in transforms (Case, Query, and Table_Comparison) and functions (decode, ifthenelse, lookup, lookup_ext, lookup_seq).

> **i 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.

  > **i Note**
  >
  > The software treats an empty string differently, depending on the source type. For example, Oracle treats an empty string as a NULL value, but DB2 and Microsoft SQL Server treat an empty string as a zero-length varchar value. For the most current information on the treatment of empty strings, refer to the documentation for your specific database server.

- When using the Equal (=) or Not Equal (<>) operators to compare a value with a NULL constant, the comparison always evaluates to FALSE. Use the IS NULL and IS NOT NULL operators to test for NULL values in the WHERE clause of the Query transform, the lookup_ext function, and the SAP Data Services scripting language.

If you currently run the software pre-version 11.5.0 scripts and data flows, it is recommended that you migrate them to use the ANSI varchar behavior because the previous varchar behavior will not be supported in a future version.

The software supports reading, transforming, and loading National Language Supported (NLS) data from different language locales using the varchar data type. The software supports national character-set data types in the following databases:

Database	Version	National character-set data type
DB2	7.0 and higher	graphic, vargraphic
MS SQL Server	7.0 and higher	nchar, nvarchar
Oracle	9i and higher	nchar, nvarchar2

When the software encounters a national character-set data type in an expression, it binds the column with the data type recommended by the database.

The engine reads and loads national character-set data types seamlessly without the need for you to configure a locale for a database client and its datastore for the columns that use these data types.

Related Information

NULL values and empty strings

Processing with and without UTF-16 Unicode [page 1449]

5 Data type conversion

SAP Data Services converts data types during certain processes so that the data types from different sources use the same data types for processing.

Data Services converts data types under the following situations:

- Performs date arithmetic
- Converts to or from internal data types
- Processes expressions
- Works with number data types
- Converts between explicit data types
- Converts between native data types

Data Services reads, loads, and calls stored procedures with unknown data types under certain circumstances. Data Services can work with unknown data types provided your database servers can do the following:

- Convert from VARCHAR to the native (unknown) data type
- Convert from the native (unknown) data type to VARCHAR

For complete information about how Data Services converts data types for each database type, see the *Designer Guide*.

Related Information

Data type conversion

6 Smart Editor and the function wizard

SAP Data Services provides the Smart Editor and the function wizard as tools to help you create scripts, expressions, and custom functions.

The Smart Editor and function wizard allow you to create scripts and custom functions without having to manually type column, function, and variable names.

Open the Smart Editor and/or the function wizard from the following locations:

- Query transform editor:
  - *Mapping* tab
  - *From* tab
  - *Where* tab
- Script editor
- Conditional editor
- While Loop editor
- Function wizard, *Define Input Parameter(s)* page
- SQL transform editor
- Case transform editor

Smart Editor library pane [page 1062]
To create functions in the Smart Editor, use the library pane, which provides access to a list of function types and variables.

Smart Editor pane and syntax coloring [page 1063]
To build functions, expressions, or scripts in the Smart Editor, use a combination of elements from the library pane and manual entry.

6.1 Smart Editor library pane

To create functions in the Smart Editor, use the library pane, which provides access to a list of function types and variables.

Library pane tab descriptions

Tab	Description
Functions	Displays existing functions including built-in, custom, and imported. Scroll through the list of existing functions. Expand function groups to find the exact function to use. For example, expand Aggregate and find avg, count, min, max, and so on. Use the Find node to search for a function. A small text box at the bottom of the pane describes the item and provides example syntax.
Variables	Displays variables, parameters, and data type formats. Use the right-click menu options in the current context. For example, select an object from the Patterns, Substitution Parameters, or other groups from the library pane. Right-click to open the context menu. The dropdown context menu options include Insert, delete, and define.
Data	Displays schemas for the sources of the current data flow, including nested schemas for the current context. For example, when you create a Where clause in a Query transform editor, open the Smart Editor in the *Where* tab. The *Data* tab lists schemas from the connected sources. The *Data* tab does not appear when you open the Smart Editor from a script object. For example, when you open Smart Editor to create a new function from the *Functions* tab in object library.

Parent topic: Smart Editor and the function wizard [page 1061]

Related Information

Smart Editor pane and syntax coloring [page 1063]

6.2 Smart Editor pane and syntax coloring

To build functions, expressions, or scripts in the Smart Editor, use a combination of elements from the library pane and manual entry.

As you add functions and variables to your script, expression, or custom function, the editor pane shows the text. The editor changes the text color to indicate the type of language element it represents as follows:

- Pink: Quoted strings
- Blue: Keywords
- Green: Comments
- Black: Functions, operators, and variables

Enable the selection list and tool tips from the right-click popup menu in the editor pane. Then use the popup selection list and tool tips as a guide when entering text in the edit pane. With the popup editing tools, you can even close the library pane as you develop your string.

The popup selection list shows the same items as in the library pane. But the popup selection displays the items in alphabetical order instead of grouping them into the categories. 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 also open it from the tool bar so it always stays open.

Parent topic: Smart Editor and the function wizard [page 1061]

Related Information

Smart Editor library pane [page 1062]

7 Data Services Functions and Procedures

Functions and procedures are alike because they both take input values. However, they differ in what they do with the values.

- 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 include the following:

- Parameters passed into a data flow
- Values from a column of data
- Variables defined inside a script

7.1 Data Services functions

Functions in SAP Data Services take input values and produce a return value.

Some functions produce the same or similar values as transforms. However, functions operate on single values and transforms operate on data sets. Use functions alone or include functions in expressions. Data Services provides many built-in functions for you to use. You can also create custom functions by using Data Services scripting language and the Smart Editor.

Data Services can perform implicit data type conversions on date, time, datetime, and interval values.

Find complete details about working with functions in the *Designer Guide*.

Related Information

Smart Editor and the function wizard [page 1061]
Data Services scripting language [page 1307]

7.2 Data Services function types

The function type determines where you can use a function.

The following table describes each type of function and where you can call it from.

Function types

Type	Description
Aggregate	Generates a single value from a set of values. Aggregate functions, such as max, min, and count, use the data set specified by the expression in the *Group By* tab of a query. Call an aggregate function only from within a Query transform. You cannot call an aggregate function from custom functions or scripts.
Iterative	Maintains state information from one invocation to another. An iterative function, such as the lookup function, contains state information that lasts only until you execute the query in which you use the function. Call an iterative function only from within a Query transform. You cannot call an iterative function from other functions or scripts.
Stateless	Does not maintain state information from one invocation to the next. Use stateless functions, such as to_char or month, anywhere you can use expressions.

7.3 Data Services built-in functions

SAP Data Services provides built-in functions to use with specific types of transforms, transactions, and special processes.

Data Services enables you to create custom functions. It also provides several built-in functions that you can use throughout Data Services.

The following table lists each built-in function available in Data Services. It includes a description of each function as well as the function category. Find the function listed under the related category in the function wizard and smart editor.

Function	Category	Description
abs	Math	Returns the absolute value of an input number. abs [page 1082]
add_months	Date	Adds a given number of months to a date. add_months [page 1083]
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, the software returns *-1*. ascii [page 1084]
avg	Aggregate	Calculates the average of a given set of values. avg [page 1085]
base64_decode	Miscellaneous	Returns the source data after decoding the base64-encoded input. base64_decode [page 1086]
base64_encode	Miscellaneous	Returns the base64-encoded data in the engine locale character set. base64_encode [page 1087]
before_image	Miscellaneous	Returns the before image value of a row. This function is available for the Map_Operation transform only and is applicable to UPDATE rows. before_image [page 1087]
begin_delta_load	Miscellaneous	Marks the beginning of a delta load for a Replication Server real-time change data capture (CDC) job. begin_delta_load [page 1088]
begin_initial_load	Miscellaneous	Marks the beginning of the initial load for a Replication Server real-time CDC job. begin_initial_load [page 1089]
cast	Conversion	Returns a value in the cast data type. cast [page 1090]
ceil	Math	Returns the smallest integer value greater than or equal to an input number. ceil [page 1092]

Function	Category	Description
chr	String	Gets character representation of provided ASCII value. chr [page 1093]
concat_date_time	Date	Returns a datetime from separate date and time inputs. concat_date_time [page 1094]
copy_from_remote_system	Miscellaneous	Associates with a file location object that contains file transfer protocol information and local—remote server information. Transfers a file copy from a remote server to a local server before reading and processing the file in a work flow. You may use wildcard characters (* or ?) to select a group of files. copy_from_remote_system [page 1094]
copy_to_remote_system	Miscellaneous	Associates with a file location object that contains file transfer protocol information and local—remote server information. Transfers the generated output file, which is output to the local server, to a remote server. You may use wildcard characters (* or ?) to select a group of files. copy_to_remote_system [page 1096]
count	Aggregate	Counts the number of values in a table column. count [page 1099]
count_distinct	Aggregate	Counts the number of distinct non-null values in a table column. count_distinct [page 1100]
current_configuration	Miscellaneous	Returns the name of the datastore configuration in use at runtime. current_configuration [page 1102]

Function	Category	Description
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. current_system_configuration [page 1102]
dataflow_name	Miscellaneous	Returns the data flow name in which this call exists. If the call is not in a data flow, returns NULL. dataflow_name [page 1103]
datastore_field_value	Miscellaneous	Retrieves the value of a specified datastore field. datastore_field_value [page 1103]
date_diff	Date	Returns the difference between two dates or times. date_diff [page 1104]
date_part	Date	Extracts a component of a given date. date_part [page 1106]
day_in_month	Date	Determines the day in the month on which the given date falls. day_in_month [page 1107]
day_in_week	Date	Determines the day in the week on which the given date falls. day_in_week [page 1108]
day_in_year	Date	Determines the day in the year on which the given date falls. day_in_year [page 1109]
db_type	Miscellaneous	Returns the database type of the datastore configuration in use at runtime. db_type [page 1109]
db_version	Miscellaneous	Returns the database version of the datastore configuration in use at runtime. db_version [page 1111]
db_database_name	Miscellaneous	Returns the database name of the datastore configuration in use at runtime. db_database_name [page 1112]

Function	Category	Description
db_owner	Miscellaneous	Returns the real owner name for the datastore configuration that is in use at runtime. db_owner [page 1113]
decode	Miscellaneous	Returns an expression based on the first condition in the specified list that evaluates to TRUE. decode [page 1114]
decrypt_aes	Cryptographic	Decrypts the input string using the user-specified passphrase and key length using the AES algorithm. decrypt_aes [page 1116]
decrypt_aes_ext	Cryptographic	Decrypts cipher text using the AES key generated using the specified passphrase and salt. Ensure that the passphrase and salt are the same as the passphrase and salt you used to encrypt the data. decrypt_aes_ext [page 1117]
double_metaphone	String	Encodes the input string using the Double Metaphone algorithm and returns a string. double_metaphone [page 1118]
encrypt_aes	Cryptographic	Encrypts the input string using the user-specified passphrase and key length using the AES algorithm. encrypt_aes [page 1120]
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. encrypt_aes_ext [page 1121]
error_context	Miscellaneous (Can only be found when creating a script)	Returns the context of the caught exception. For example, "|Session datapreview_job|data flow debug_ DataFlow|Transform Debug" error_context [page 1122]

Function	Category	Description
error_message	Miscellaneous (Can only be found when creating a script)	Returns the error message of the caught exception. error_message [page 1123]
error_number	Miscellaneous (Can only be found when creating a script)	Returns the error number of the caught exception. error_number [page 1123]
error_timestamp	Miscellaneous (Can only be found when creating a script)	Returns the timestamp of the caught exception. error_timestamp [page 1124]
exec	System	Sends a command to the operating system for execution. exec [page 1124]
extract_from_json	Conversion	Extracts JSON data directly from a single column in a database table, and converts it into its internal nested relational data model (NRDM). To access this function, open the function wizard from within a new function call. extract_from_json [page 1129]
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, open the function wizard from within a new function call. extract_from_xml [page 1130]
file_copy	Miscellaneous	Copies an existing file to a different location using the same file name or a different file name. Copies a group of files indicated by a wildcard to a different existing directory. Overwrites any existing target files when overwrite flag is set to *1*. The original file still exists in the original location after `file_copy`. file_copy [page 1131]
file_delete	Miscellaneous	Deletes an existing file, or deletes a group of files indicated by a wildcard. file_delete [page 1133]

Function	Category	Description
file_exists	Miscellaneous	Checks to see if a given file or directory exists. file_exists [page 1134]
file_move	Miscellaneous	Moves an existing file to a different location using the same file name or a different file name. Moves a group of files indicated by a wildcard to a different existing directory. Overwrites any existing target files when overwrite flag is set to 1. Original file does not exist in the original location after `file_move`. file_move [page 1135]
fiscal_day	Date	Converts a given date into an integer value representing a day in a fiscal year. fiscal_day [page 1137]
floor	Math	Returns the largest integer value less than or equal to an input number. floor [page 1138]
gbq2file	Conversion	Optimizes performance when you export large-volume data from Google BigQuery to a local file via Google Cloud Storage. gbq2file [page 1139]
gen_row_num_by_group	Miscellaneous	Returns group row number of the record. gen_row_num_by_group [page 1141]
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_row_num [page 1142]
gen_uuid	Miscellaneous	Returns a unique varchar string identifier. gen_uuid [page 1143]
get_domain_description	Miscellaneous	Returns the description of a value when given the domain name and the value. get_domain_description [page 1144]

Function	Category	Description
get_env	Environment	Returns a value for the specified environmental variable. get_env [page 1145]
get_error_filename	Environment	Returns the full path and file name for the error log. get_error_filename [page 1145]
get_file_attribute	Miscellaneous	Returns date created, date modified, or size (in bytes) of a physical file. get_file_attribute [page 1146]
get_monitor_filename	Environment	Returns the full path and file name for the monitor log. get_monitor_filename [page 1147]
get_trace_filename	Environment	Returns the full path and file name for the trace log. get_trace_filename [page 1148]
greatest	Miscellaneous	Returns greatest of the list of one or more expressions. greatest [page 1149]
host_name	Miscellaneous	Returns the name of the computer on which the job is executing. host_name [page 1151]
ifthenelse	Miscellaneous	Allows conditional logic in mapping and selection operations. ifthenelse [page 1151]
index	String	Returns the index of a given word in a string. index [page 1152]
init_cap	String	Changes the characters in a string to title case. init_cap [page 1154]
interval_to_char	Conversion	Returns a string representation of the interval. interval_to_char [page 1155]
is_group_changed	Miscellaneous	Returns 1 if the group is changed, 0 otherwise. is_group_changed [page 1156]

Function	Category	Description
is_set_env	Environment	Verifies if the specified environment variable is set. is_set_env [page 1157]
is_valid_date	Validation	Indicates if an expression can be converted into a valid date value. is_valid_date [page 1158]
is_valid_datetime	Validation	Indicates if an expression can be converted into a valid datetime value. is_valid_datetime [page 1160]
is_valid_decimal	Validation	Indicates if an expression can be converted into a valid decimal value. is_valid_decimal [page 1161]
is_valid_double	Validation	Indicates if an expression can be converted into a valid double value. is_valid_double [page 1162]
is_valid_int	Validation	Indicates if an expression can be converted into a valid integer value. is_valid_int [page 1163]
is_valid_real	Validation	Indicates if an expression can be converted into a valid real value. is_valid_real [page 1164]
is_valid_time	Validation	Indicates if an expression can be converted into a valid time value. is_valid_time [page 1165]
isempty	Miscellaneous	Indicates if a nested table contains data. isempty [page 1166]
isweekend	Date	Indicates that a date corresponds to Saturday or Sunday. isweekend [page 1167]
job_name	Miscellaneous	Returns the name of the job in which the call to this function exists. job_name [page 1170]
job_run_id	Miscellaneous	Retrieves the job run ID for the current job execution. Job_Run_ID [page 1171]

Function	Category	Description
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 [page 1171]
julian_to_date	Conversion	Converts a Julian value to a date. julian_to_date [page 1172]
key_generation	Database	Generates keys for the specified table, after determining the appropriate starting value. key_generation [page 1173]
last_date	Date	Returns the last date of the month for a given date. last_date [page 1174]
least	Miscellaneous	Returns the least in a list of one or more expressions. least [page 1175]
length	String	Returns the number of characters in a given string. length [page 1177]
literal	String	Returns an input constant expression without interpolation. Allows you to assign a pattern to a variable without interpolation. literal [page 1178]
ln	Math	Returns the natural logarithm of the given numeric expression. ln [page 1179]
load_from_gcs_to_gbq	Conversion	Loads data from Google Cloud Storage into Google BiqQuery tables. load_from_gcs_to_gbq [page 1180]
load_from_s3_to_redshift	Conversion	Uses the Redshift COPY command to copy data files from an Amazon Simple Storage Service (S3) bucket to a Redshift table. load_from_s3_to_redshift [page 1182]

Function	Category	Description
load_to_xml	Conversion	Generates XML text from NRDM and loads it into a single database column. (Assumes that the database supports XML text in its columns.) load_to_xml [page 1185]
local_to_utc	Date	Converts the input datetime of any time zone to Coordinated Universal Time (UTC). local_to_utc [page 1187]
log	Math	Returns the base-10 logarithm of the given numeric expression. log [page 1188]
long_to_varchar	Conversion	Converts a data type from long to varchar. long_to_varchar [page 1189]
lookup	Lookup	Finds a value in one table or file based on values in a second table or file. lookup [page 1190]
lookup_ext	Lookup	Finds data from a database table, flat file, or memory datastore table. lookup_ext [page 1194]
lookup_seq	Lookup	Finds a value in one table based on values in a second table or file, and ensures that the sequence matches. lookup_seq [page 1204]
lower	String	Changes the characters in a string to lowercase. lower [page 1208]
lpad	String	Pads a string with characters from a specified pattern. lpad [page 1209]
lpad_ext	String	Pads a string with logical characters from a specified pattern. lpad_ext [page 1210]
ltrim	String	Removes specified characters from the start of a string. ltrim [page 1211]

Function	Category	Description
ltrim_blanks	String	Removes blank characters from the start of a string. ltrim_blanks [page 1213]
ltrim_blanks_ext	String	Removes blank and control characters from the start of a string. ltrim_blanks_ext [page 1213]
mail_to	System	Sends the specified e-mail message. mail_to [page 1214]
match_pattern	String	Matches whole input strings to simple patterns supported by Data Services. This function does not match substrings. match_pattern [page 1216]
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. match_regex [page 1219]
match_simple	String	Matches a whole input string to simple patterns supported by the software for this function. Does not match substrings. match_simple [page 1224]
max	Aggregate	Returns the maximum value from a list. max [page 1226]
min	Aggregate	Returns the minimum value from a list. min [page 1227]
mod	Math	Returns the remainder when one number is divided by another. mod [page 1228]
month	Date	Determines the month in which the given date falls. month [page 1229]
num_to_interval	Conversion	Converts a numeric value to an interval. num_to_interval [page 1230]
nvl	Miscellaneous	Replaces NULL values. nvl [page 1231]

Function	Category	Description
power	Math	Returns the value of the give expression to the specified power. power [page 1232]
previous_row_value	Miscellaneous	Returns the column value of previous row. previous_row_value [page 1233]
print	String	Prints the given string to the trace log. print [page 1234]
pushdown_sql	Miscellaneous	Allows you to create dynamic WHERE clauses. pushdown_sql [page 1235]
quarter	Date	Determines the quarter in which the given date falls. quarter [page 1237]
raise_exception	Miscellaneous	Causes the software to generate an exception. raise_exception [page 1238]
raise_exception_ext	Miscellaneous	Same as raise_exception, but takes a second parameter for an exit code. raise_exception_ext [page 1238]
rand	Math	Returns a random number between 0 and 1. rand [page 1239]
rand_ext	Math	Returns a random number between 0 and 1. rand_ext [page 1240]
regex_replace	String	Matches the whole input string to the pattern that is specified with regular expressions. The regular expressions are based on the POSIX standard. Then flags and replaces the matching part of the input string with the replacement string provided. regex_replace [page 1241]
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 [page 1242]

Function	Category	Description
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. replace_substr_ext [page 1243]
repository_name	Miscellaneous	Returns a database connection string and owner name. The connection string and owner name are the ID for the repository from which the job is run. For example: `beq-local.DBUser`. repository_name [page 1245]
restore_repserver_cdb_backlogged_transactions	Miscellaneous	Restores backlogged transactions saved in the Replication Server real-time CDC datastore. restore_repserver_cdb_backlogged_transactions [page 1246]
round	Math	Rounds a given number to the specified precision. round [page 1247]
rpad	String	Pads a string with characters from a given pattern. rpad [page 1248]
rpad_ext	String	Pads a string with logical characters from a given pattern. rpad_ext [page 1249]
rtrim	String	Removes given characters from the end of a string. rtrim [page 1250]
rtrim_blanks	String	Removes blank characters from the end of a string. rtrim_blanks [page 1251]
rtrim_blanks_ext	String	Removes blank and control characters from the end of a string. rtrim_blanks_ext [page 1252]

Function	Category	Description
sap_openhub_processchain_execute	SAP	Starts the process chain that extracts data from an SAP NetWeaver Business Warehouse (BW) and loads the extracted data into an Open Hub Destination table. sap_openhub_processchain_execute [page 1253]
sap_openhub_set_read_status	SAP	Sends the read status for the Open Hub table to SAP NetWeaver BW. sap_openhub_set_read_status [page 1256]
search_replace	String	Searches input parameters and replaces by matching criteria and values specified by search table. search_replace [page 1258]
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_cdc_checkpoint [page 1261]
set_env	Environment	Sets an environmental variable temporarily to a specified value. set_env [page 1262]
sleep	Miscellaneous	Suspends the execution of the data flow or work flow from where it is called. sleep [page 1263]
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. soundex [page 1264]
sql	Database	Runs a SQL operation in the specified database. sql [page 1265]
sqrt	Math	Returns the square root of the given expression. sqrt [page 1267]
smtp_to	System	Sends the specified e-mail message using the SMTP protocol. smtp_to [page 1267]

Function	Category	Description
string_to_number	String	Returns the integer sum of all characters from the input string. string_to_number [page 1269]
substr	String	Returns a specific portion of a string starting at a given point in the string. substr [page 1270]
sum	Aggregate	Calculates the sum of a given set of values. sum [page 1271]
sysdate	Date	Returns the current date as listed by the Job Server operating system. sysdate [page 1272]
system_user_name	Miscellaneous	Returns the user name used to log into the Job Server operating system. system_user_name [page 1273]
systime	Time	Returns the current time as listed by the operating system. systime [page 1274]
table_attribute	Miscellaneous	Retrieves the value of a specified table attribute. table_attribute [page 1276]
to_char	Conversion	Converts a date or numeric type to a string. to_char [page 1277]
to_date	Conversion	Converts a string to a date. to_date [page 1280]
to_decimal	Conversion	Converts a varchar to a decimal. to_decimal [page 1281]
to_decimal_ext	Conversion	Converts a varchar to a decimal, including precision as a parameter. to_decimal_ext [page 1282]
to_varchar	Conversion	Converts a date or numeric expression to varchar based on a given format. to_varchar [page 1283]

Function	Category	Description
to_WKT_point	Conversion	Converts latitude and longitude to a geometry point in Well Known Text (WKT) format. to_WKT_point [page 1288]
total_rows	Database	Returns the number of rows in a particular table in a datastore. total_rows [page 1290]
translate	String	Takes the input string and translates each character to its corresponding mapping to return the output string. translate [page 1291]
trunc	Math	Truncates a given number to the specified precision. trunc [page 1292]
truncate_table	Miscellaneous	Allows you to explicitly expunge data from a memory table. truncate_table [page 1293]
upper	String	Changes the characters in a string to uppercase. upper [page 1294]
utc_to_local	Date	Converts the input in Coordinated Universal Time (UTC) to the desired timezone value. utc_to_local [page 1295]
varchar_to_long	Conversion	Converts a data type from varchar to long. varchar_to_long [page 1296]
wait_for_file	Miscellaneous	Returns the existing files that match the input file pattern. wait_for_file [page 1297]
week_in_month	Date	Determines the week in the month in which the given date falls. week_in_month [page 1298]
week_in_year	Date	Determines the week in the year in which the given date falls. week_in_year [page 1299]
WL_GetKeyValue	String	Returns the value of a given keyword in Web log search strings. WL_GetKeyValue [page 1301]

Function	Category	Description
word	String	Returns one word out of a string. word [page 1301]
word_ext	String	Returns the word identified by its position in a delimited string. word_ext [page 1302]
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. workflow_name [page 1304]
year	Date	Determines the year in which the given date falls. year [page 1304]

7.4 Descriptions of Data Services built-in functions

Each built-in function has specific syntax and uses specific SAP Data Services objects to achieve the stated objective.

For details about the specific syntax and related objectes for each function, see the individual function topics.

> **i Note**
>
> For information about operators, functions, and transforms that you can use as push-down functions with Data Services, see SAP Note 2212730.

7.4.1 abs

Use the abs function to return the absolute value of a number. The absolute value (sometimes known as the modulus) of a number is the value of a number without regard to its sign – it can also be thought of as the distance of a number from zero.

> **Syntax**
>
> ```
> abs(<num>)
> ```

Return value

decimal, double, int, or real

The absolute value of the given number, `<num>`. The type of the return value is the same as the type of the original number.

Where

`<num>`	The source number.

Example

Function	Results
`abs(12.12345)`	`12.12345`
`abs(-12.12345)`	`12.12345`

7.4.2 add_months

Use add_months to add a given number of months to a date.

Syntax

```
add_months(<original_date>,<months_to_add>)
```

Return value

date

Where

`<original_date>`	Specify the starting year.month.date.
`<months_to_add>`	Number of months to add to the original date.

Details

The `<months_to_add>` can be any integer. If `<original_date>` is the last day of the month or if the resulting month has fewer days than the day component of `<original_date>`, then the result is the last day of the resulting month. Otherwise, the result has the same day component as `<original_date>`.

Example

Function	Results
`add_months('1990.12.17', 1)`	`'1991.01.17'`
`add_months('2001.10.31', 4)`	`'2002.2.28'`

7.4.3 ascii

Use the ascii function to return a decimal value of an ASCII code of the first character in the input string.

Syntax

Syntax

```
ascii(<input_string>)
```

Return Value

Int

Where

`<input_string>`	The source string.

Details

Returns the decimal value of the ASCII code of the first character in the input string. Returns -1 if the first character is not a valid ASCII character.

Example

Function	Results
`ascii('AaC')`	65

7.4.4 avg

Use the avg function to calculate 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 function calculates the average to the same precision as the input value.

Where

`<value_list>`	The source values for which to calculate an average, such as values in 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.

7.4.5 base64_decode

Use the base64_decode function to return the source data after decoding the base64-encoded input.

Syntax

```
base64_decode(<base64-encoded input>, 'UTF-8')
```

Return Value

varchar or blob

If the base64-encoded input is NULL, or the size of the data is 0, Data Services returns NULL. Otherwise, it returns the base64-decoded data that conforms to RFC 2045.

Where

`<base64-encoded input >`	The base64-encoded input data. Supports varchar and blob data types.
UTF-8	The code page of the output data. **Note** UTF-8 is required for Data Integrator version 11.7.3. This parameter is not required when the input data type is blob.

Related Information

base64_encode [page 1087]

7.4.6 base64_encode

Use the base64_encode function to return the base64-encoded data in the engine locale character set.

Syntax

```
base64_encode(<input data>, 'UTF-8')
```

Return Value

varchar or blob

Returns base64-encoded data. If the input data is NULL or the size is 0, Data Services returns NULL. Otherwise, it returns the base64-encoded data that conforms to RFC 2045.

Where

`<input data>`	The input data that needs to be encoded to base64. Supports varchar and blob data types.
UTF-8	The code page of the input data. **Note** UTF-8 is required for Data Integrator version 11.7.3. This parameter is not required when the input data type is blob.

Related Information

base64_decode [page 1086]

7.4.7 before_image

Use the before_image function to retrieve the before image value of a row.

This function is available only for the Map_Operation transform and is applicable to UPDATE rows.

> **Note**
>
> You cannot use quoted strings in the parameter value for before_image. Because of this limitation, you cannot use a global variable. For example, `before_image(T1.Col1)` is valid, but `before_image('T1.Col1')` is not.

> **Syntax**
>
> ```
> before_image(<column_name>)
> ```

Return value

Inline

The return value is the same data type as the input column data type. For example, if the input column (column_name) is an integer, the software returns an integer.

Where

`<column_name>`	The name of the table column.

Example

```
before_image(Epml.SALARY)
```

Related Information

Map_Operation [page 952]

7.4.8 begin_delta_load

Use the begin_delta_load function to mark the beginning of a delta load for a real-time Replication Server change data capture (CDC) job.

> **Syntax**
>
> ```
> begin_delta_load()
> ```

Details

Use this optional function to configure CDC for databases that use the Replication Server. The function provides the source table CDC configuration in the Replication Server for the entire job. This function also prepares the Replication Server for making changed data available for the CDC reader to process. It also processes restore requests at the beginning of each iteration of the continuous work flow.

If the software does not call this function explicitly in a script in a continuous work flow (CWF), then it calls the function implicitly at the beginning of each iteration of the CWF. Because the software requires that the job calls all Replication Server CDC data flows inside one continuous work flow, begin_delta_load executes implicitly when the continuous work flow starts. But if you have other steps to perform before processing your CDC data flows, you can explicitly mark the beginning of a delta load inside the CWF.

The guidelines for the begin_delta_load function are as follows:

- If the job does not call the function begin_initial_load, then begin_delta_load validates the Replication Server configuration (create CDC queue, and replication definition, and subscription).
- If the job calls begin_initial_load, then the begin_delta_load function does the necessary configuration to transition from initial load to delta load.
- Ensure that the begin_delta_load function is executed inside a continuous work flow.
  For each iteration of the continuous work flow, the begin_delta_load function does the following:
  - Begins the transaction
  - Communicates with child data flow CDC readers whether there is CDC data available to process
  - After all the readers have successfully processed CDC data, commits the transaction
  - Repeats these steps in next iteration. Committing the transaction moves the pointer in the CDC queue in Replication Server to the next set of changed data.

7.4.9 begin_initial_load

Use the begin_initial_load function to mark the beginning of an initial load for a real-time Replication Server change data capture (CDC) job.

Syntax

```
begin_initial_load()
```

Details

If the work flow calls begin_initial_load, it must call it before the begin_delta_load function. When begin_initial_load executes, it provides the source table CDC configuration for the Replication Server real-time CDC data stores for the entire job. This function also prepares the Replication Server for capturing CDC transactions from the source system while the initial load is in progress.

When begin_delta_load executes in the CDC continuous work flow, it notifies the Replication Server to mark the endpoint of CDC transactions that were captured by the Replication Server during initial load. If you select

Enable Auto Correct in the CDC reader, the reader generates automatically corrected records for all the transactions that occurred during the initial load.

7.4.10 cast

Use the cast function to explicitly convert an expression of one data type to another.

Syntax

```
Cast('<expression>','<data_type>')
```

Return Value

Returns the same value as in data_type.

Where

`<expression>`	Input expression to be cast to target data type.
`<data_type>`	Target data type that is a built-in data type and specified as a constant string. For example, 'decimal `(28,7)`'.

Details

The cast function explicitly converts the value of the first parameter into the built-in data type that you specify in the second parameter. The following table 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

Cast type compatibility matrix

From / To	Date	Date time	Decimal	Double	Int	Interval	Real	Time	Time stamp	Varchar
Decimal			X	X	X	X	X			X
Double			X	X	X	X	X			X
Int			X	X	X	X	X			X
Interval			X	X	X	X	X			X
Real			X	X	X	X	X			X
Time		X						X	X	X
Time stamp	X	X						X	X	X
Varchar	X	X	X	X	X	X	X	X	X	X

The following table contains syntax for the given data_type.

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'
time	'time'
interval	'interval'

The following table shows the date&time format for the cast() function:

Timestamp	yyyy.mm.dd hh24:mi:ss.ff

Datetime	yyyy.mm.dd hh24:mi:ss
Date	yyyy.mm.dd
Time	hh24:mi:ss

Example

Input	Output
`cast('20.3','decimal(3,1)')`	20.3

7.4.11 ceil

Use the ceil function to return the smallest integer value greater than or equal to a number.

Syntax

```
ceil(<num>)
```

Return value

decimal, double, int, or real

The indicated integer, cast as the same type as the original number, `<num>`.

Where

`<num>`	The source number.

Example

Function	Results
`ceil(12.12345)`	`13.00000`

Function	Results
`ceil(12)`	`12`
`ceil(-12.223)`	`-12.000`

7.4.12 chr

Use the chr function to convert a decimal ASCII code to a character.

Syntax

```
chr (<integer_expression>)
```

Return Value

ASCII character

Where

`<integer_expression>`	Integer from 0 through 255. Returns NULL if the integer expression is not in this range.

Details

This function returns the character associated with the specified ASCII code decimal number. If you specify a value of less than 0 or greater than 255 for the integer_expression parameter, the software returns NULL. Use chr to insert control characters into character strings. For example, `chr(9)` can be used to insert <tab>.

Example

Function	Results
`chr(65)`	'A'

7.4.13 concat_date_time

Use the concat_date_time function to return a datetime from separate date and time inputs.

Syntax

```
concat_date_time(<date>,<time>)
```

Where

<date>	Date input value.
<time>	Time input value.

Return value

datetime

The datetime value obtained by combining the inputs.

Example

```
concat_date_time(MS40."date",MS40."time")
```

7.4.14 copy_from_remote_system

Use the `copy_from_remote_system` function to copy files from a remote server to your local server.

Syntax

```
copy_from_remote_system("<file_location_object name>", "<remote_file_name>",
"<remote_sub_directory>")
```

Return value

Int

Returns 1 if function is successful. Returns 0 if function is not successful.

Where

`<file_location_object name>`	Name of file location object.
`<remote_file_name>`	Name of the file to copy from the remote server.
`<remote_sub_directory>`	Optional. Subdirectory path to the remote location for the file specified in the `<file_name>` parameter. Complete this parameter when the file is located farther in the hierarchy than what is specified in the file location object. Data Services concatenates this information with the location specified in the file location object to form a complete directory path. **Example** You want to download the file named `nw_sales_q2` from the `Sub_directory1` folder in the following directory structure in your Amazon S3 cloud storage: **Directory structure** – Bucket – Parent folder + Child folder1 – Child folder2 Sub_directory1 Sub-directory2 The file location named "AmazonS3_bucket" for an Amazon S3 cloud storage has the *Remote directory* option set to `<bucket>/<Parent folder>/<Child folder2>`. However, the file is located in the `Sub_directory1` folder. Therefore, you use the function to add the additional subdirectory: `copy_from_remote_system('AmazonS3_bucket','nw_sales_q2','Sub_directory1')` The parameter is applicable to the following cloud storage types: • Amazon S3 • Azure blob storage • Azure Data Lake Store • Hadoop File System • Google Cloud storage

Details

Use this function in a workflow script to transfer a file from a remote server to a local server. Then use the file as a source in a data flow. Also set the workflow script to copy groups of files from a remote server to your local

server. Indicate groups of files with a wildcard in the file name using either an asterisk (*) or question mark (?) character.

The software uses the local directory, remote directory, remote path prefix, and transfer protocol information (FTP, SFTP, or SCP) from the named file location object. Then it concatenates the file location information with the information in the `copy_from_remote_system` function.

More details

- Use this function on regular files only. For example, you cannot use this function to copy directory or symbolic links.
- Directory information is included in the file location object in such a way that this function does not copy any directories.
- This function does not recursively copy files under any subdirectories.

Example

You want to copy a file named `prod.txt` from a remote server to a local server to use it in a data flow. Data flow includes a query transform. A file location object named `flo_ftp_1` contains the file transfer protocol. You set up a script object that contains the `copy_from_remote_system` function as follows:

Sample Code

```
copy_from_remote_system("flo_ftp_1", "prod.txt");
```

The function uses the FTP information in the file location object `flo_ftp_1` to copy the file named `prod.txt` that is located in a remote directory.

Related Information

Use built-in functions for file transfer protocol

7.4.15 copy_to_remote_system

Use the `copy_to_remote_system` function to copy generated files from your local server to a remote server after SAP Data Services processing.

Syntax

```
copy_to_remote_system("<file_location_object name>", "<local_file_name>",
"<remote_sub_directory>")
```

Return value

int

Returns 1 if function is successful. Returns 0 if function is not successful.

Where

`<file_location_object name>`	Name of file location object.
`<local_file_name>`	Name of the file to read from the local directory. Data Services concatenates with the local directory information specified in the file location object to form the full file name. **Example** Local directory is `C:\local<local_file_name>` is `file.j` Concatenated location and file name is `C:\local\file.j` Use wildcards in the file name to designate a group of files. Use either * or ? characters. For example, `C:\local\*.j`.

`<remote_sub_directory>` Optional. Subdirectory path to the remote location in the cloud for the file specified in the `<file_name>` parameter. Complete this parameter when the location in the remote directory is located farther in the hierarchy than what is specified in the file location object.

Data Services concatenates this information with the location specified in the file location object to form a complete directory path.

> **Example**
>
> You want to place the file generated from Data Services named `nw_sales_q2` from your local server to the `Sub_directory1` folder in the following directory structure in your Amazon S3 cloud storage:
>
> **Directory structure**
> – Bucket
> – Parent folder
> + Child folder1
> – Child folder2
> Sub_directory1
> Sub-directory2
>
> The file location named "AmazonS3_bucket" for an Amazon S3 cloud storage has the *Remote directory* option set to `<bucket>/<Parent folder>/<Child folder2>`. However, the file named nw_sales_q2 should be uploaded to the `Sub_directory1` folder. Therefore, you use the function to add the additional subdirectory:
>
> ```
> copy_to_remote_system('AmazonS3_bucket','nw_sales_q2','Sub_directory1')
> ```

The parameter is applicable to the following cloud storage types:

- Amazon S3
- Azure blob storage
- Azure Data Lake Store
- Hadoop File System
- Google Cloud storage

Details

Use the `copy_to_remote_system` function in a workflow script to transfer the generated output file, which is output to the local server, to a remote server. The software uses the attributes, such as local directory, remote directory, remote path prefix, and transfer protocol information (FTP, SFTP, or SCP) from the named file location object.

- Use this function on regular files only. For example, you cannot use this function to copy directory or symbolic links.
- Directory information is included in the file location object in such a way that this function does not copy any directories.
- This function does not recursively copy files under any subdirectories unless the subdirectory is included in the remote directory value.

Example

You want to move a file after Data Services processing from a local server to a remote server using file transfer protocol.

The file name is `prod_out.txt`. You set the file transfer protocol in a file location object named `flo_ftp_1`. To copy the file to the remote system, you create a script as follows:

Sample Code

```
copy_to_remote_system("flo_ftp_1", "prod_out.txt");
```

The script uses the FTP information in the file location object `flo_ftp_1` to copy the file named `prod_out.txt` that is located in the local directory.

Related Information

Use built-in functions for file transfer protocol

7.4.16 count

Use the count function to count the number of values in a group.

count

Syntax

```
count(<column>)
```

Return value

int

The number of rows in the column that have a non-NULL value.

Where

`<column>`	The column in the input table in which to count values.

Example

To determine the number of customers located in a specific sales region, use the count function with a filter defined in the *Where* tab of the query editor. The following WHERE clause selects the rows in the REGION column with the value `TX`:

```
REGION = "TX"
```

With the target column selected, enter the count function in the *Mapping* tab of the editor:

```
count(REGION)
```

Example

7.4.17 count_distinct

Use the count_distinct function to return 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 region is stored in a column named `REGION`. To count the number of distinct regions the customers come from, use the count_distinct function with a filter defined in the *Where* tab of the query editor. Enter the count_distinct function in the *Mapping* tab of the editor, as follows:

```
count_distinct(REGION)
```

Input

Name	Region	Country
Cust 1	East	US
Cust 2	East	US
Cust 3	West	US
Cust 4	East	France

Output

```
count_distinct(REGION) = 2
```

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

> **i Note**
>
> If you want to provide more resources to execute the `count_distinct` function, select *Run as a separate process*. This option creates a separate sub data flow process for the `count_distinct` function when Data Services executes the data flow.

Related Information

Distributed data flow execution

7.4.18 current_configuration

Use the current_configuration function to return the name of the datastore configuration that the software uses at runtime.

If the datastore does not support multiple configurations, for example, the datastore is a memory datastore, the function returns the name of the datastore instead.

Syntax

```
current_configuration(ds_name)
```

Return Value

varchar

Where

<ds_name>	The name you enter when you create the datastore.

Example

Create a job and add a script with, for example, the following line.

```
print('Datastore Configuration used at runtime: [current_configuration()]')
```

Returns, for example, the following to the trace log:

```
Datastore configuration used at runtime: Test_DS
```

7.4.19 current_system_configuration

Use the current_system_configuration function to return the name of the system configuration the software uses at runtime.

If there is no system configuration defined, the function 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
```

7.4.20 dataflow_name

Use the dataflow_name function to return the name of the data flow in which this call exists.

If the call is not in a data flow, the function returns NULL.

Syntax

```
dataflow_name()
```

Return Value

varchar

Example

```
print('Data Flow Name: [dataflow_name()]')
```

7.4.21 datastore_field_value

Use the datastore_field_value function to retrieve the value of a specified datastore field.

Syntax

```
datastore_field_value(<ds_name, field_name>)
```

Return Value

varchar

Where

`<ds_name>`	The name you enter when you create the datastore.
`<field_name>`	The name of the field.

Details

The `<field_name>` should match the name in the language of the datastore. To view the language of the datastore, open the datastore editor and click *Show ATL*. The ATL contains the valid field names. If the function cannot find a specified field or the datastore is invalid, the software returns NULL. If the `<field_name>` is 'password,' the software returns NULL..

Example

Function	Results
`datastore_field_value('mssql', 'sql_server_database')`	'DBUser'

7.4.22 date_diff

Use the date_diff function to return the difference between two dates or times.

Syntax

```
date_diff(<date1>,<date2>,'<fmt_str>')
```

Return Value

int

Where

`<date1, date2>`	The dates between which the function determines the difference.	
`<fmt_str>`	The string that describes the format of the dates. Choose from the following values:	
	D	Day
	H	Hours
	M	Minutes
	S	Seconds
	MM	Months
	YY	Years

Details

If date1 is smaller than date2, the date_diff function returns a positive value. To cause the function to return only a positive value, surround the function with the abs() function.

i Note

When you use the sysdate function with date_diff, be aware that the value the sysdate function returns is datetime. 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 uses only the date for calculations. It ignores the time data. If you change the data type to datetime, Data Services uses both a date and a time. If the data type is datetime and you don't want to use the time data, use the to_char function to truncate the timestamp from sysdate.

Example

Function	Results
`date_diff(start_date,sysdate(),'D')`	The number of days between the date in the column start_date and the current date.
`date_diff(start_time,systime(),'M')`	The number of minutes between the time in the column start_time and the current time.

7.4.23 date_part

Use the date_part function to extract a component of a given date.

Syntax

```
date_part(<in_date>,'<fmt_str>')
```

Return Value

int

Where

<in_date>	The input date.	
<fmt_str>	The string describing the format of the extracted part of the date. Choose from the following values:	
	YY	Year
	MM	Month
	DD	Day
	HH	Hours
	MI	Minutes
	SS	Seconds

Details

This function takes in a datetime and extracts the component requested as an integer.

Note

The function displays year as four digits, not two.

Example

Function	Results
`date_part('1990.12.31', 'YY')`	`1990`
`date_part('1991.01.17 23:44:30', 'SS')`	`30`

7.4.24 day_in_month

Use the day_in_month function to determine the day in the month on which the input date falls.

Syntax

```
day_in_month(<date1>)
```

Return value

int

The number from 1 to 31 that represents the day in the month that `<date1>` occurs.

Where

`<date1>`	The source date.

This function extracts the day component from the date value.

Example

Function	Results
`day_in_month(to_date('Jan 22, 1997','mon dd, yyyy'))`	`22`
`day_in_month(to_date('02/29/1996','mm/dd/yyyy'))`	`29`

Function	Results
`day_in_month(to_date('1996.12.31','yyyy.mm.dd'))`	31

7.4.25 day_in_week

Use the day_in_week function to determine the day in the week on which the input date falls.

Syntax

```
day_in_week(<date1>)
```

Return value

int

The number from 1 (Monday) to 7 (Sunday) that represents the day in the week that `<date1>` occurs.

Where

`<date1>`	The source date.

This function allows you to categorize dates according to the day of the week the date falls on. For example, all dates for which this function returns a "3" occur on Wednesday.

Example

Function	Results
`day_in_week(to_date('Jan 22, 1997','mon dd, yyyy'))`	3 (Wednesday)
`day_in_week(to_date('02/29/1996','mm/dd/yyyy'))`	4 (Thursday)
`day_in_week(to_date('1996.12.31','yyyy.mm.dd'))`	2 (Tuesday)

7.4.26 day_in_year

Use the day_in_year function to determine the day in the year on which the input date falls.

Syntax

```
day_in_year(<date1>)
```

Return value

int

The number from 1 to 366 that represents the day in the year that `<date1>` occurs.

Where

`<date1>`	The source date.

Example

Function	Results
`day_in_year(to_date('Jan 22, 1997','mon dd, yyyy'))`	22
`day_in_year(to_date('02/29/1996','mm/dd/yyyy'))`	60
`day_in_year(to_date('1996.12.31','yyyy.mm.dd'))`	366 (1996 was a leap year.)

7.4.27 db_type

Use the db_type function to return the database type of the datastore configuration in use at runtime.

Syntax

```
db_type(<ds_name>)
```

Return Value

varchar

Possible db_type() return values for datastore types are as follows:

Datastore Types	Possible db_type() Return Value
Adapter	Adapter
Database	Attunity_Connector, DB2, Informix, Memory, Microsoft_SQL_Server, ODBC, Oracle, SAP, SAP_BW, SQL_Anywhere, SAP Sybase (for SAP ASE), Sybase_IQ, Teradata
JDE One World	DB2, Microsoft_SQL_Server, ODBC, or Oracle
JDE World	ODBC
Oracle Applications	Oracle
PeopleSoft	Microsoft_SQL_Server, or Oracle
SAP Applications	SAP
SAP BW Source	SAP
SAP BW Target	SAP_BW
Siebel	DB2, Microsoft_SQL_Server, or Oracle

Where

`<ds_name>`	The datastore name you enter when you create the datastore.

Details

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. Using the function in a SQL statement allows the SQL statement to use the correct database type for each run no matter what datastore configuration is in use.

Example

If you have a SQL transform that performs a function that you have to write differently for database types, you can tell the system 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}');
```

7.4.28 db_version

Use the db_version function to return the database version of the datastore configuration in use at runtime.

Syntax

```
db_version(<ds_name>)
```

Return Value

varchar

Possible db_version() return values are:

Database type	Version
Oracle	Currently supported versions
Microsoft SQL Server	Currently supported versions
DB2 UDB	Currently supported versions
Informix IDS	Currently supported versions
SQL Anywhere	Currently supported versions
SAP ASE	Currently supported versions
SAP Sybase IQ	Currently supported versions
Teradata	Currently supported versions
""	An empty string is returned for any other database type

Where

<ds_name>	The datastore name you enter when you create the datastore.

Details

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. Using the function in a SQL statement allows the SQL statement to use the correct database version for each run no matter what datastore configuration is in use.

Example

If you have a SQL transform that performs a function that is written differently for different versions of Oracle, you can tell the system 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}');
```

7.4.29 db_database_name

Use the db_database_name function to return the database name of the datastore configuration in use at runtime.

Syntax

```
db_database_name(<ds_name>)
```

Return Value

varchar

Where

`<ds_name>`	The datastore name you enter when you create the datastore.

Details

This function is useful if your datastore has multiple configurations and is accessing an MS SQL Server or SAP ASE database. For a datastore configuration that is using either of these database types, you enter a database name, when you create a datastore. This function returns that database name.

For example, master is a database name that exists in every Microsoft SQL Server and SAP 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. Using the function in a SQL statement allows the SQL statement to use the correct database name for each run no matter what datastore configuration is in use.

This function returns an empty string for datastore configurations without MS SQL Server or SAP ASE as the Database Type.

Example

If you have a SQL transform that performs a function that is written differently for different versions of database types, you can tell the system 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}');
```

7.4.30 db_owner

Use the db_owner function to return the real owner name for the datastore configuration that is in use at runtime.

Syntax

```
db_owner(<ds_name>, <alias_name>)
```

Return Value

varchar

Where

`ds_name`	The datastore name that you entered when you created the datastore.
`alias_name`	The name of the alias that you created in the datastore, then mapped to the real owner name when you created a datastore configuration.

Details

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 run no matter what datastore configuration is in use.

Example

```
$real_owner = db_owner('sales_ds', 'sales_person');
```

7.4.31 decode

Use the decode function to return an expression based on the first condition in the specified list of conditions and expressions that evaluates to TRUE.

Syntax

```
decode(<condition_and_expression_list>,<default_expression>)
```

Return value

`<expression>` or `<default_expression>`

Returns the value associated with the first <condition> that evaluates to TRUE. The data type of the return value is the data type of the first <expression> in the <condition_and_expression_list>.

If the data type of any subsequent <expression> or the <default_expression> is not convertible to the data type of the first <expression>, Data Services produces an error at validation. If the data types are convertible but do not match, a warning appears at validation.

Where

<condition_and_expression_list>	A comma-separated list of one or more pairs that specify a variable number of conditions. Each pair contains one <condition> and one <expression> separated by a comma. Specify at least one <condition> and <expression> pair. The <condition> evaluates to TRUE or FALSE. If the <condition> evaluates to TRUE, the <expression> is the value that the function returns.
<default_expression>	An expression that the function returns if none of the conditions in <condition_and_expression_list> evaluate to TRUE. A <default_expression> is required.

Details

The decode function provides an easier way to write nested ifthenelse functions. In nested ifthenelse functions, you 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 scripting language.

If a condition compares a varchar value with trailing blanks, the decode function ignores the trailing blanks.

To compare a NULL value (NULL constant or variable that contains a NULL constant), use the IS NULL or IS NOT NULL operator. If you use the Equal (=) or Not equal to (<>) operator, the comparison against a NULL value always evaluates to FALSE.

Example

Function	Results
`decode((COUNTRY = 'FRANCE'), 'French', (COUNTRY = 'GERMANY'), 'German', (COUNTRY = 'ITALY'), 'Italian', (COUNTRY = 'USA'), 'America', (COUNTRY IS NULL), 'Unknown', 'Others')`	If the value in the COUNTRY column 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`.

7.4.32 decrypt_aes

Use the decrypt_aes function to decrypt the input string with the user-specified pass phrase and key length using the AES algorithm.

Note

The decrypt_aes function is intended to decrypt data that was encrypted by encrypt_aes function.

Syntax

```
decrypt_aes(<encrypted_input_string>,<passphrase>,<key_length_in_bits>)
```

Return value

Returns plain string as varchar.

In case of a failure, the function throws an exception of type execution error, which results in termination of the job. You can catch the exception by using try/catch handlers.

If the encrypted input string is empty, then the return value is an empty string.

If the encrypted input string is NULL, then the return value is NULL.

Where

<encryptedinput_string>	A varchar input string to be decrypted.
<passphrase>	A varchar character string.
<key_length_in_bits>	An int value of 128, 192, or 256.

Example

For security purposes, secure the passphrase in a database and read it using a sql() function into a local or global variable. Then you can pass the variable to the passphrase parameter.

```
 #read the passphrase
from a secured source such as a database
$G_passphrase = sql('PASSWORD_DATASTORE', 'select PASSPHRASE from PASSWORD');
encrypt_aes(SOURCE.SSN,
$G_passphrase, 128);
```

Similar to other string functions, this function can be called from a custom function, in the column mapping of a Query transform, or in a script in the work flow.

Related Information

encrypt_aes [page 1120]

7.4.33 decrypt_aes_ext

Use the decrypt_aes_ext function to decrypt the input string with the user-specified passphrase, salt, and key length using the AES algorithm.

Ensure that the passphrase and salt are the same as the passphrase and salt used to encrypt the data.

The function generates an AES key of the specified key length using the specified passphrase and the key generation algorithm PKCS5_PBKDF2_SHA256. This key is used for decrypting the encrypted input string.

Syntax

```
decrypt_aes_ext(<Varchar Encrypted_input_string>,<Varchar
Passphrase>,<Varchar Salt>,<Int Key_length_in_bits>)
```

Return value

Returns plain string as varchar.

In case of a failure, the function throws an exception of type execution error, which results in the termination of the job. You can catch the exception by using try/catch handlers.

If the encrypted input string is empty, then the return value is an empty string.

If the encrypted input string is NULL, then the return value is NULL.

If you fail to provide the same passphrase and key length used for encryption to this function, then the call does not fail but instead returns an incorrect output.

Where

`<Encrypted_input_string>`	A varchar input string to be decrypted.
`<Passphrase>`	A varchar character string with at least one character.
`<Salt>`	A varchar that must be exactly eight ASCII characters.
`<Key_length_in_bits>`	An int value of 128, 192, or 256.

Example

For security purposes, 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, call this function from a custom function, in the column mapping of a Query transform, or in a script in the work flow.

7.4.34 double_metaphone

Use the double_metaphone function to encode the input string using the Double Metaphone algorithm.

Syntax

```
double_metaphone(<input_str, alternate, return_if_empty>)
```

Return Value

varchar

Returns the string containing the double metaphone encoding of the input string. The length of the return string depends on the length of the input string, but it is always shorter than the input string.

Where

`<input_str>`	The source string to encode.
`<alternate>`	A flag to control how the function returns the encoded strings. • When 0, returns the primary encoding. When no primary encoding, returns NULL or the input string based on how you set `<return_if_empty>`. • When input is not 0, returns the alternate encoding. If no alternate encoding, returns NULL or the input string based on how you set `<return_if_empty>`. • When NULL or invalid, such as non numeric, defaults to 0.
`<return_if_empty>`	A flag to determine whether to return null or the input string when there is no encoding. When 0, return NULL. Otherwise, return the input string when there is no encoding. When the parameter is NULL or invalid, such as non numeric, defaults to 1. When input is empty, there is no primary or secondary encodings. When `<return_if_empty>` = 0, then returns NULL. When `<return_if_empty>` = 1, then returns the empty string.

Details

Only use this function for input strings in English. The function ignores non-English characters.

When input is NULL, the function returns NULL.

When the second or third parameter has an invalid value, defaults to 0 and 1, respectively.

Example

Function	Result
`Print(double_metaphone('Hello',0,0);`	Prints the double metaphone of the word “Hello.”

Function	Result
`double_metaphone($VAR, 1,1);`	If the string stored in $VAR does not have encoding available, then returns the original string.
`double_metaphone($VAR,'a','b');`	Returns the primary double metaphone encoding or the variable $VAR when the primary encoding does not exist.

7.4.35 encrypt_aes

Use the encrypt_aes function to encrypt the input string using the specified passphrase and key length with the AES algorithm.

> **Note**
>
> Do not decrypt data that you encrypted within Data Services using the encrypt_aes function outside of Data Services. Instead, use the decrypt_aes function to decrypt this data.

> **Syntax**

```
encrypt_aes(<input_string>,<passphrase>,<key_length_in_bits>)
```

Return value

Returns encrypted string as varchar. The size of the encrypted string is about twice as large as the size of plain text. Therefore, ensure that you have enough space to hold the encrypted string.

In case of a failure, the function throws an execution error and terminates the job. You can catch the exception by using try/catch handlers.

If the input string is empty, then the function returns an encrypted string. The encrypted string is different for multiple calls of the encrypt_aes() function with an empty input string.

If the input string is NULL, then the return value is NULL.

Where

`<input_string>`	A varchar input string to be encrypted.
`<passphrase>`	A varchar character string.

<key_length_in_bits>	An int value of 128, 192, or 256.

Details

For security purposes, 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.

Example

```
 #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);
```

Like other string functions, you can call the encrypt_aes function from a custom function, in the column mapping of a Query transform, or in a script in the work flow.

7.4.36 encrypt_aes_ext

Use the encrypt_aes_ext function to encrypt an input string using the specified passphrase, salt, and key length with the AES algorithm.

Syntax

```
encrypt_aes_ext(<Varchar Input_string>,<Varchar Passphrase>,<Varchar
salt>,<Int Key_length_in_bits>)
```

Return value

Returns encrypted string as base64 encoded string. The size of the encrypted string is 1.3 times larger than the size of plain text. Therefore you must have enough space to hold the encrypted string.

In case of a failure, the function throws an exception of type execution error, which results in the termination of the job. You can catch the exception by using try/catch handlers.

If the input string is empty, then the return value is empty.

If the input string is NULL, then the return value is NULL.

Where

`<Input_string>`	A varchar input string to be encrypted.
`<Passphrase>`	A varchar character string.
`<Salt>`	A varchar that must be exactly eight ASCII characters.
`<Key_length_in_bits>`	An int value of 128, 192, or 256.

Details

The function generates an AES key of specified key length using the specified passphrase, salt, and the key generation algorithm PKCS5_PBKDF2_SHA256. The function uses this key for encrypting the input string.

For security purposes, 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.

Example

```
#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);
```

Like other string functions, you can call the encrypt_aes_ext function from a custom function, from the column mapping of a Query transform, or from a script in the work flow.

7.4.37 error_context

Use the error_context function to return the context of a caught exception.

Syntax

```
error_context()
```

Return value

varchar 512

Example

The following shows an example of the caught exception.

"|Session datapreview_job|Dataflow debug_DataFlow|Transform Debug"

Related Information

Catch error functions [page 37]

7.4.38 error_message

Use the error_message function to return the error message from a caught exception.

Syntax

```
error_message()
```

Return value

varchar 512

Related Information

Catch error functions [page 37]

7.4.39 error_number

Use the error_number function to return the error number of a caught exception.

Syntax

```
error_number()
```

Return value

int

Related Information

Catch error functions [page 37]

7.4.40 error_timestamp

Use the error_timestamp function to return the timestamp of a caught exception.

Syntax

```
error_timestamp()
```

Return value

timestamp

Related Information

Catch error functions [page 37]

7.4.41 exec

Use the exec function to send a command to the operating system.

Sends a command to the operating system for execution. With this function, you can add a program to a Data Services job.

Syntax

```
exec(<command_file, parameter_list, flag>)
```

Return value

Varchar(1020)

Returns up to 1020 characters that depend on the value of `<flag>` .

Where

<command_file>	A string that indicates the location and file name to execute. This string is relative to the Job Server location. It can be an absolute or relative path. Ensure that the files and directories in the path are available from the Job Server computer. The <command_file> can be a Windows batch file, a UNIX shell script, or a binary executable. To run other interpreted scripts, ensure that the <command_file> is the name of the command interpreter, such as 'perl', and the script is the first parameter in the <parameter_list>.
<parameter_list>	A string that lists the values to pass as arguments to the command file. Separate parameters with spaces. When passing no parameters to an executable, enter an empty string (' ').
<flag>	An integer that specifies what information appears in the return value string and how to respond when <command_file> cannot be executed or exits with a nonzero operating system return code.

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: • The command never produces output • The calling job does not need output
2	Standard output	NULL string	Use this flag if you do not intend to track the status of the command other than the presence or absence of output.
3	NULL string	NULL string	
4	Standard output	Error message string	
5	NULL string	Error message string	
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).	

Flag	If successful, returns:	On error:	Notes:
256	NULL string	NULL string	Use this flag to run your program independently of Data Services. Unlike flags 0-8, if you use flag 256, Data Services does not wait until the command (executable program) completes before continuing with job processing. In this case, the command runs independently of Data Services and stdout, stderr, and return code cannot be returned. Raises an exception (`System function failure`) if the program cannot be launched (e.g., program file not found).

Details

- Ensure that the program that this function executes does not wait for any user input (such as 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 also hangs. For flag 256, Data Services continues if the program hangs for input.
- For flags 4 and 5, the return value format for an error message string is:

  ```
  'error-number: error-message-text'
  ```

  The first field is exactly 7 characters wide and the second character 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 SAP `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'
  ```

  The first field is exactly 7 characters wide and the second field begins at index 10. The program produces the return code. Zero indicates success. Consult your program documentation to determine the meaning of other codes.

  > **Example**
  >
  > 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, which is the return code from the executed command, as a string of digits. Data Services automatically converts the string of digits to an integer wherever necessary. The second field extracts as a regular string.

Example

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 the components of that column in a mapping or WHERE clause. For example:

```
substr(query.foo, 1, 7);
```

```
substr(query.foo, 10, 1020);
```

Details about exec: Use of remote shells [page 1127]

For the exec function, use a remote shell to run a command elsewhere on the network.

7.4.41.1 Details about exec: Use of remote shells

For the exec function, use a remote shell to run a command elsewhere on the network.

To use a remote shell, consider the following information:

- The `<command_file>` named in an exec call can be 'rsh' on either Windows or UNIX systems to call the remote shell facility. Use the 'rsh' as a means of running a command on a machine elsewhere on the network.

Example

For example:

- `exec('rsh', '<RemoteMachineName> <CommandToRunRemotely> <CmdArg1> <CmdArg2>', 0);`

- `exec('rsh', '<RemoteBox> -l<RemoteUser> <RemoteCommand> <CmdArg>', 3);`

Call the remote shell facility sparingly, because the remote connection setup, remote authentication, and increased message traffic reduce performance.

- For `<flag>` values 4, 5, and 8, Data Services receives the 'rsh' (or 'remsh') command type return code. For example, 0 if it successfully gets a remote connection and authorization, non zero if it does not get a remote connection and authorization. There is no relation between this return code and the return value of the remote command inherent in the remote shell mechanism on all the operating systems.
  To work around this behavior, wrap the remote command in a `.bat` file (Windows) or shell script (UNIX). Wrapping the remote command gets the command return code `%errorlevel%` if Windows or `$?` if UNIX, and prints it to stdout or stderr.

  Example

  For example:

  - `exec('rsh', '<RemoteMachineName> <remcmdWrapper>.bat <CmdArg1> <CmdArg2>', 8);`
  - `exec('rsh', '<RemoteBox> -l<RemoteUser> /usr/acta/<remcmdWrapper> <CmdArg>', 4);`

- The system administrator of the remote machine sets up access for the product user. The `.rhosts` and— or the `hosts.equiv` file has an entry allowing this access.
  - If the remote machine is Windows, ensure that the Remote Shell Service is running on it.
  - If the remote machine is UNIX, ensure that the Remote Shell daemon rshd is running on it.

  Consult your operating system documentation for more information.

Example

The following examples apply to Windows or UNIX. If you use 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. Use 'cmd' or its equivalent if either:

- The “command” is a built-in of the shell. For example, 'DIR' is not a program in Windows.
- Piping, a single '|' in an argument, occurs
- In either Windows or UNIX, the pipe symbol sends the output of one command to another command. Only use a pipe symbol inside quotes. In Data Services, the double pipe symbol (||) concatenates strings. Only use double pipes outside quotes.

Also, remember that the forward and backward slash symbols ('\' '/') are interchangeable in Windows. However use only the forward slash ('/') as a directory separator on UNIX.

```
exec('cmd', 'dir ' || $filename, 8);
exec('cmd', 'x:/bin/program1.exe | x:/bin/postprocess.bat', 4);
exec(SRC.PROGNAME, ARGTBL.ARGS || ' lastArg', 2);
exec('c:\Data Services\bin\clone_and_rename.bat', TBL.FNAME, 1);
exec('C:\Perl5\bin\perl.exe', 'C:\sandbox\stats.pl 20 50 3000', 0);
```

7.4.42 extract_from_json

Use the extract_from_json function to extract JSON data from a field or database table into the output schema in a Query transform using SAP Data Services nested relational data model (NRDM) structure.

Syntax

```
extract_from_json(<json_column_name>,<schema_json_name>,<enable_validation>)
```

Where

Value	Description
`<json_column_name>`	The name of the input column that contains the JSON data. Ensure that the column data type is varchar.
`< schema_json_name >`	The name of the JSON Schema format that describes the incoming JSON data. Import the metadata for this format into Data Services. Data Services displays the format names in the *Formats* tab of the object library. Ensure that the input value is a constant so that Data Services knows the output schema at design time.
`<enable_validation>`	Enable a comparison of the incoming JSON data against the format you specify for `<schema_json_name>`. The JSON data structure and the JSON format structure must match. If the incoming data is not valid, and you enable this option, the data flow throws an exception. Enter `1` to validate. Enter `0` to disable the validation option. Validation is useful during development. It provides a more precise error if a problem occurs with the incoming JSON string.

Details

The function supports varchar data types in the input column. Use the following methods to extract data from clob and long data types.

- Data Services converts a clob data type input to varchar when you select the *Import unsupported data types as VARCHAR of size* option. Select the 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.

The output schema of this function matches that of the JSON format that you specify. Select any of the top level columns or the two columns that Data Services generates in the NRDM for output:

- Select any number of the top level columns from the JSON schema. The return types of each column are the same as the type defined in the JSON schema.
- `AL_ERROR_NUM` - Generated column that returns an error code that indicates that the error occurred inside the function. A `0` indicates success and any non zero integer indicates an error.

- AL_ERROR_MSG - Generated column that returns an error message if AL_ERROR_NUM is not 0. Otherwise, returns NULL.

Example

```
extract_from_json(<cust_note_column>,<note_format>,1)
```

7.4.43 extract_from_xml

Use the extract_from_xml function to extract XML data from a field or database table into the output schema in a Query transform using SAP Data Services nested relational data model (NRDM) structure.

Syntax

```
extract_from_xml(<xml_column_name>,<schema_dtd_name>,<enable_validation>)
```

Value	Description
<xml_column_name>	The name of the input column that contains the XML text. Ensure that the column data type is varchar.
< schema_dtd_name >	The name of the DTD or XML Schema format that describes the incoming XML text. Import the metadata for this format into Data Services. Data Services displays the format names in the *Formats* tab of the object library. Ensure that the input value is a constant so that Data Services knows the output schema at design time.
<enable_validation>	Enable a comparison of the incoming XML data against the format you specify for <schema_dtd_name>. Ensure that the XML data structure and the XML format structure match. When you enable <enable_validation>, the data flow throws an exception if the incoming data is not valid. Enter 1 to validate. Enter 0 to disable the validation option. Validation is useful during development. It provides a more precise error if a problem occurs with the incoming XML.

Details

The software supports varchar data types in the input column. 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.

Once you provide parameter values, the output schema of this function matches that of the XML that you specified. Select any of the top-level columns or the two columns that Data Services generates in the NRDM for output:

- Select any number of the top-level columns from XML schema. The return type of each column matches the type that you define in the XML schema.
- `AL_ERROR_NUM` - Generated column that returns an integer that indicates if an error occurred inside the function. A `0` indicates success and any non-zero integer indicates an error.
- `AL_ERROR_MSG` - Generated column that returns an error message when `AL_ERROR_NUM` is not `0`. Otherwise, returns `NULL`.

Example

```
extract_from_xml(<cust_note_column>,<note_format>,1)
```

Related Information

XML extraction and parsing for columns

7.4.44 file_copy

Use the file_copy function to copy an existing file to a different location using the same file name or a different file name.

Syntax

```
file_copy(<source>,<target>,overwrite_if_exist)
```

Return Value

int

Returns 1 if the file is copied to the target location. Returns 0 if the file is not copied.

Where

`<source>`	The absolute path and name of the file to copy. Use a wildcard (*) in the file name to copy a group of files that match the wildcard criteria. Ensure that you have permission to access the source file location.

`<target>`	The absolute path for the location of the copied file. • To keep the same name as the source file, do not include a file name. • To rename the moved file, include a different file name. If you copy a group of files using a wildcard (*), enter the absolute path for the location of the copied files. Ensure that you have permission to access the target file and location.
`overwrite_if_exist`	Enter a 0 or 1. 0 = Do not overwrite any existing file. The software does not overwrite the file if it exists in the target location. **i Note** In this case, the software return value is 0, and the software issues a warning that no files were copied to the target location. 1 = Overwrite any existing file. The software automatically overwrites the file if it exists in the target location. **i Note** In this case, the software return value is 1, the software copies the source file to the target location, and it overwrites any existing file with the same name in the target location.

Details

The file_copy function overwrites any existing target file when you set the overwrite flag to 1. The source file still exists in the original location after `file_copy`.

Use `file_copy` on regular file types only. For example, you cannot use `file_copy` for directory file types or symbolic links.

Do not use the following characters in the source and target file name: `\ / : * ? " < > |` except when you use the asterisk (*) in a file name to indicate a wildcard.

Example

Function	Results
`file_copy('C:\temp\my_list.txt','D:\my_lists\list_a.txt',1)`	Copies a file and pastes it into a different location with a different file name. The function copies the `my_list.txt` file from the source location `C:\temp` and pastes it to the target location `D:\my_lists\` using a new name `list_a.txt`. The function automatically overwrites any existing file of the same name in the target location because the overwrite flag is set to 1.
`file_copy('C:\temp\my_*.txt','D:\my_lists',1)`	Copies a group of files from one location and pastes them into a different location. The function copies all files that match the wildcard file name `my_*.txt` from the source location `C:\temp` to the target location `D:\my_lists`. The function automatically overwrites any existing files of the same name in the target location because the overwrite flag is set to 1.

7.4.45 file_delete

Use the file_delete function to delete an existing file, or delete a group of files indicated by a wildcard (*).

Syntax

```
file_delete(<DelFileName>)
```

Return Value

int

Returns 1 if the stated file is deleted. Returns 0 if the stated file is not deleted.

Where

`<DelFileName>`	The absolute path and file name of an existing file to delete. Use a wildcard (*) in the file name to delete a group of files that match the wildcard criteria. Ensure that you have permission to the file and directory.

Details

Use `file_delete` on regular file types only. For example, you cannot use `file_delete` for directory file types or symbolic links.

You may not use the following characters in the deleted file name: `\ / : * ? " < > |` except when you use the asterisk (*) in a file name to indicate a wildcard.

Example

Function	Results
`file_delete('C:\users\myfile.txt')`	Deletes a file. The function deletes the file named `myfile.txt` from `C:\users`.
`file_delete('C:\users\my*.txt')`	Deletes a group of files matching a wildcard. The function deletes all files that match the wildcard file name `my*.txt` from the `C:\users` directory.

7.4.46 file_exists

Use the file_exists function to see if a 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 it is 0 bytes long. Returns a 0 if the file or directory is not present on the disk.

Where

`<file_path>`	The file name and path, relative to where the Job Server is running. It can be an absolute or relative path.

Details

Example

Examples:

Call sleep for 1 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')
```

7.4.47 file_move

Use the file_move function to move an existing file or group of files to a different location using the same file name or a different file name.

Syntax

```
file_move(<source>,<target>, overwrite_if_exist)
```

Return Value

int

Returns 1 if the file is moved to the target location. Returns 0 if the file is not moved.

Where

`<source>`	The absolute path and name of the file to move. Use a wildcard (*) in the file name to move a group of files that match the wildcard criteria. Ensure that you have permission to access the source file and location.
`<target>`	The absolute path for the location of the moved file (or files). Ensure that you have permission to access the target file and location.
`overwrite_if_exist`	Enter a 0 or 1. 0 = Do not overwrite any existing file. The software does not overwrite the file if it exists in the target location. **i Note** In this case, the function return value is 0, and the software issues a warning that no files were moved to the target location. 1 = Overwrite any existing file. The software automatically overwrites the file if it exists in the target location. **i Note** In this case, the function return value is 1, the software moves the source file to the target location, and any existing file with the same name in the target location is overwritten.

Details

Overwrites any existing target file when the overwrite flag is set to 1. The source file does not exist in the original location after `file_move`.

Use `file_move` on regular file types only. For example, you cannot use `file_move` for directory file types or symbolic links.

- The source file no longer exists in the original location after `file_move`.
- You may not use the following characters in the source and target file name: \ / : * ? " < > |

  However, you may use the asterisk character (*) in a file name to indicate a wildcard.
- You can also use the `file_move` function to rename a file.

Example

Function	Results
`file_move('C:\temp\my_list.txt','C:\users',1)`	Moves a file from one folder to a different folder in the same directory. The software moves the file named `my_list.txt` from the `C:\temp` folder to the `C:\users` folder. After `file_move`, the source file `my_list.txt` no longer exists in the source folder. The software automatically overwrites any existing file named `my_list.txt` in the target location because the overwrite flag is set to 1.
`file_move('C:\temp\my_list.txt','C:\temp\my_list2.txt',0)`	Renames a file. The software renames the file `my_list.txt` to `my_list2.txt`. After the function successfully executes, the source file `my_list.txt` no longer exists. In this example, the `overwrite_if_exist` flag is set to 0, which means: • If the file doesn't already exist in the target location, the function return value is 1, and the file is moved to the target location. • If the file already exists in the target location, the function return value is 0, and the software issues a warning that the file was not moved to the target location because the file already exists.
`file_move('C:\temp\my*.txt', 'C:\users', 1)`	Move a group of files using a wildcard. The software moves all files that match the wildcard file name `my*.txt` from the source `C:\temp` to the target `C:\users`. After `file_move`, the files that match the wild card file name no longer exist in the source folder. The software automatically overwrites any files that already exist in the target folder because the overwrite flag is set to 1.

7.4.48 fiscal_day

Use the fiscal_day function to convert a date into an integer value that represents a day in a fiscal year.

Syntax

```
fiscal_day('<start_year_date>',<in_date>)
```

Return Value

int

Where

`<start_year_date>`	The first month and day of a fiscal year. Use the format: 'mm.dd'.
`<in_date>`	The date you want to convert. Use any valid datetime.

Details

Example

Function	Results
`fiscal_day('03.01', '1999.04.20')`	`51`

7.4.49 floor

Use the floor function to return the largest integer value equal to or less than a number.

Syntax

```
floor(<num>)
```

Return value

decimal, double, int, or real

The indicated integer, cast as the same type as the original number, `<num>` .

Where

`<num>`	The source number.

Details

Example

Function	Results
`floor(12.12345)`	`12.00000`
`floor(12)`	`12`
`floor(-12.223)`	`-13.000`

7.4.50 gbq2file

Use the gbq2file function to optimize software performance when you export large-volume Google BigQuery results to a user-specified file on your local machine.

Syntax

```
gbq2file('<GBQ_datastore_name>','<any_query_in_GBQ>','<local_file_name>','<file_location_object>', '<field_delimiter>','/<numeric_row_delimiter>');
```

Return value

int

Returns 1 if function is successful. Returns 0 if function is not successful.

Where

`<GBQ_datastore_name>`	Name of the Google BigQuery application datastore in Data Services.
`<any_query_in_GBQ>`	Name of the query in Google BigQuery.
`<local_file_name>`	Local file location and file name in which to store the Google data. Should be the same location as your local server.
`<file_location_object>`	Name of the Google Cloud Storage file location object in Data Services.
`<field_delimiter>`	Optional. The field delimiter to use between fields in the exported data. The default is a comma.
`<numeric row delimiter>`	Numeric value for the row delimiter For example, /013 **i Note** Default is 10, hex 0A.

Details

The software uses information in the associated Google cloud storage (GCS) file location object to identify your GCS connection information, bucket name, and compression information.

How the function works:

1. The function saves your Google BigQuery results to a temporary table in Google.
2. The function uses `export job` to export data from the temporary table to GCS.

   > **i Note**
   >
   > If the data is larger than 1 GB, Google exports the data in multiple files.

3. The function transfers the data from your Google Cloud Storage to the local file that you specified.
4. After the transfer is complete, the function deletes the temporary table and any files from Google Cloud Storage.

For details about creating a Google BigQuery application datastore, see the *Supplement for Google BigQuery*.

Related Information

Google Cloud Storage file location

7.4.51 gen_row_num_by_group

Use the gen_row_num_by_group function to generate a column of row identification numbers for each ID group in the specified column.

Syntax

```
gen_row_num_by_group(<expression_list>)
```

Return Value

Integer

Where

`<expression_list>`	A list of one or more comma-separated expressions.

Details

This function groups the rows in a table based on the values in the specified expression_list in the natural order. It returns a row ID beginning with 1, then increments it sequentially by 1 for each row in the group. When the group changes, the function restarts numbering at 1.

Example

For example, you have a table that lists record contracts by record number and contract ID. Values in Contract ID column are not unique.

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

Input

Record	Contract_ID	Revised_by	Revision_date
record 4	2	Joe	2/24/05
record 5	2	Sue	2/30/05

When you apply `gen_row_num_by_group` function to the Contract_ID column, the software adds a new column to the output table that contains row numbers by group.

```
A version = gen_row_num_by_group (Contract_ID)
```

There are two groups in the Contract_ID column: Three rows for Contract_ID 1 and two rows for Contract_ID 2. The following table shows the output with the additional Version_Num column.

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

If the `<expression_list>` value corresponds to a column in a table, like in the preceding example, the column must not be a nested relational data model (NRDM) or have the data type long. Also, do not use this function with any group by clauses or aggregate functions.

7.4.52 gen_row_num

Use the gen_row_num function to return an integer value beginning with 1, then incremented sequentially by 1 for each additional call.

Syntax

```
gen_row_num()
```

Return Value

int

Details

Each occurrence, or call, of the function in a data flow is a unique instance, resulting in a unique sequence. Two instances return values independent of each other. The first time the software calls an instance of this function, the function returns a value of 1. Subsequent calls of the same instance return the previous value incremented by 1, such as 2, 3, 4.

Each time the software calls the data flow, the software reinitializes all instances, and starts incrementing from 1.

Example

Function

```
gen_row_num(Col1)
```

Col1
0
0
0
0
0
0
0
0
0
0

Results

Col1
1
2
3
4
5
6
7
8
9
10

7.4.53 gen_uuid

Use the gen_uuid function to generate a unique identifier.

Syntax

```
gen_uuid()
```

Return value

Varchar

Returns a unique 32-character varchar string. For example, 550e8400e29b41d4a716446655440000.

7.4.54 get_domain_description

Use the get_domain_description function to return the description of a value when provided with the domain name and the value.

Syntax

```
get_domain_description(<domain_name>,<domain_value>)
```

Return value

varchar

The description is returned as a quoted string. If the value is not in the domain, then a NULL is returned.

Where

`<domain_name>`	Fully qualified domain name, including the database owner if required. For example:
	`<datastorename> . <owner.domain>`
	`<datastorename> .. <domain>`

`<domain_value>`	The constant value for which you want to return a description.

Details

Example

Function	Results
`get_domain_description('psoft..ACTION','ADL')`	"Additional"

7.4.55 get_env

Use the get_env function to return 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. Therefore, you can use the is_set_env function to determine whether a variable is set.

Where

`<variable_name>`	The name of the environment variable. Ensure that you surround the name with single quotes.

Details

Example

Function	Results
`getenv('TMP')`	`C:\Temp`

7.4.56 get_error_filename

Use the get_error_filename to return the full path and file name for the error log file that SAP Data Services generates after a job run.

Syntax

```
get_error_filename()
```

Return Value

varchar

Details

Data Services generates log files in the `<DS_COMMON_DIR>\log\<Job Server>\<repository>` directory. This log file name starts with 'error_'.

Data Services generates a different set of log files for each batch job run. It generates a different set of log files for 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.

Example

Create a job and add a script with, for example, the following lines.

```
print('Error File Name:');
print(get_error_filename());
```

Returns 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
```

7.4.57 get_file_attribute

Use the get_file_attribute function to return the value for a specified attribute of a physical file.

Syntax

```
get_file_attribute(<filename>, <attribute>)
```

Return value

double if the attribute is `size`.

datetime if the attribute is either `date_created` or `date_modified`.

Where

`<filename>`	The file name and path relative to the current Job Server. Enter a file name as a relative path or an absolute path.
`<attribute>`	One of the following attributes, entered with single quotes: date_created, date_modified, or size.

Details

Example

Function	Results
`get_file_attribute('//order', 'date_created')`	'2004:09:15:20:25:00' The format in this example is YYYY:MM:DD:HH24:MM:SS.
`get_file_ attribute(c:\database \order,'size')`	'63281' Displays file size in bytes.

Limitations

- The function cannot be pushed down.
- The function cannot be used in an ABAP data flow.
- For MS Window systems, the 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.

7.4.58 get_monitor_filename

Use the get_monitor_filename function to return the full path and file name for the monitor log (trace) file generated during a job run.

Syntax

```
get_monitor_filename()
```

Return Value

varchar

Details

Data Services generates log files in the `<DS_COMMON_DIR>\log\<Job Server>\<repository>` directory. This log file name 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.

Example

Create a job and add a script with the following lines.

```
print('Monitor File Name');
print(get_monitor_filename());
```

Returns, for example, the following to the trace log:

```
Monitor File Name
d://DI11/log/JS_Ora/repo__riv/
monitor_12_1_2004_12_06_41_12__8507da25_0b33_4ac1_9b53_fcf1e004c968.txt
```

7.4.59 get_trace_filename

Use the get_trace_filename function to return the full path and file name for the trace log that is generated during a job run.

Syntax

```
get_trace_filename()
```

Return Value

varchar

Details

Data Services generates log files in the `<DS_COMMON_DIR>\log\<Job Server>\<repository>` directory. This log file name 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.

Example

Create a job and add a script with the following lines.

```
print('Trace File Name');
print(get_trace_filename());
```

Returns, for example, the following to the trace log:

```
Trace File Name
d://DI11_97/log/JS_Ora/o920i1__riv/
trace_12_10_2004_12_06_41_12__8507da25_0b33_4ac1_9b53_fcf1e004c968.txt
```

7.4.60 greatest

Use the greatest function to return 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, Data Services converts the result into the return data type.

Where

`<expression_list>`	A list of one or more comma-separated expressions.

Details

Greatest returns the greatest of the list of one or more expressions. Data Services implicitly converts expression in the list to a normalized data type before comparison.

The software uses the following rules to determine the normalized data type.

- If the return data type is varchar, then the software implicitly normalizes all expressions to varchar before comparison.
- If the return data type is one of the date data types, then the software implicitly normalizes all the expressions in the list to that data type before comparison.

  > **Example**
  >
  > For example, if the return data type is date, and another data type is 'datetime', then the software normalizes the 'datetime' data type to 'date' before comparison.

- If the return data type is numeric, then the software implicitly normalizes all the expressions to the highest precedence numeric expression in the list.

  > **Example**
  >
  > 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). The normalized data type is decimal.

  The software converts all the expressions in the list to decimal data type before comparison. If the normalized data type is decimal, then the precision is the highest precision among all decimal data type expressions. The software preserves the scale for decimal data type expressions during implicit conversion. When the software converts an integer data type expression 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'

Output	
MAX_GRADE=greatest `(GRADE_Q1,GRADE_Q2,GRADE_Q3,GRADE_Q4)`	
ID	**MAX_GRADE**
2	'F'
3	NULL

7.4.61 host_name

Use the host_name function to return the name of the computer on which the job executes.

Syntax

```
host_name()
```

Return Value

varchar

Example

```
print('Host Name: [host_name()]');
```

7.4.62 ifthenelse

Use the ifthenelse function to enable 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.

Where

<condition>	An expression that evaluates to TRUE or FALSE.
<true_branch>	An expression that the function returns if <condition> evaluates to TRUE.
<false_branch>	An expression that the function returns if <condition> evaluates to FALSE.

Details

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.

To improve performance, Data Services pushes this function to the database. Thus, the database evaluates the IFTHENELSE logic rather than the 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 scripting language.

Example

Function	Results
`ifthenelse (ZIP < 94000, 'SOUTH', 'NORTH')`	If the value in the column ZIP is less than 94000, the value returned is SOUTH. If ZIP is greater than 94000, then the value returned is NORTH.

7.4.63 index

Use the index function to return the index of a give character sequence in a string.

Syntax

```
index(<input_string>, <index_string>, <start>)
```

Return value

int

Specifies the first location of the indicated character sequence.

Where

<input_string>	The source string.
<index_string>	The character sequence sought in <input_string>.
<start>	The position where the function starts searching in <input_string> for the character sequence contained in <index_string>. <Start> should be a positive number between 1 and the length of <input_string>.

Details

The function searches for the <index_string> beginning at the <start> position in the input_string.

- If <start> is 0, the function resets it 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 NULL.

Ensure that the characters in <index_string> exactly match the sequence of characters in <input_string>.

Example

The search is case-sensitive.

Function	Results
`index('Accounting Department', 'DEPARTMENT', 1)`	`NULL` The index_string does not match the case of the input_string so the function returns NULL.

Function	Results
`index('Accounting Department', 'Department', 1)`	`12` The string "Department" starts at position 12 when you begin counting at position 1, which is the "A" of Accounting.

7.4.64 init_cap

Use the init_cap function to convert the first letter of each word in a string to upper case and the rest of the value to lowercase. The function ignores all characters that are not alphabetic.

Syntax

```
init_cap(<value>,'<locale>')
```

Return value

varchar

The title case string. Words are delimited by white space or characters that are not alphanumeric.

Where

`<value>`	The string to be modified.
`<locale>`	Optional parameter that converts the string to the specified locale. **Note** The function supports ISO 639 language code and ISO 3166 country code formats.

Details

Example

Function	Results
`init_cap('Data Services')`	'Data Services'
`init_cap(StreetAddress)`	Writes the value, for example '1234 west washington school road', in column StreetAddress as '1234 West Washington School Road'.
`Print(Init_cap('have a nice day -hypen +plus _underscore \slash $dollar *star @at tab mIXedWORd UPPER lower !punctations 1234digits'));`	Have A Nice Day -Hypen +Plus _Underscore \Slash $Dollar *Star @At Tab Mixedword Upper Lower !Punctuations 1234digits
`init_cap(LastName,'tr')`	Converts the value in column`LastName` to have the first letter capitalized. If there are more than one last name in this column, the first letter of each word is capitalized. Converts the values in LastName to the Turkish locale, using the ISO 639 language code.

Limitations

- The function can be pushed down to Oracle databases only.
- You cannot use this function in an ABAP data flow.

7.4.65 interval_to_char

Use the interval_to_char function to convert an interval value to a string.

Syntax

```
interval_to_char(<input_interval, interval_type>)
```

Return value

varchar

The converted string.

Where

`<input_interval>`	The value of type `interval` to convert.	
`<interval_type>`	A string describing the format of the interval. Choose from the following values:	
	D	Days
	H	Hours
	M	Minutes
	S	Seconds

Details

Example

Function	Results
`interval_to_char(start_date - sysdate(), 'd')`	The number of days between the date in the column `start_date` and the current date.
`interval_to_char(start_time - systime(), 'm')`	The number of minutes between the time in the column `start_time` and the current time.

7.4.66 is_group_changed

Use the is_group_changed function to return an integer, which indicates if the current occurrence of a group of values has changed from the previous occurrence.

Syntax

```
is_group_changed(<expression>)
```

Return Value

Integer

1 = Group has changed

0 = Group has not changed

Where

`<expression>`	One or more valid input expressions separated by commas.

Details

This function groups records based on the equal value of the input expressions in parameter1 in the natural order of the input record stream. It returns 1 when the group is changed, 0 otherwise.

Example

In the following example, the results show that four of the input groups have changed.

Function	Results
`is_group_changed(state,city)`	`1,0,1,0,0,1,1`

Group ID	State	City	Group change
1	California	Los Angeles	1
2	California	Los Angeles	0
3	California	San Francisco	1
4	California	San Francisco	0
5	California	San Francisco	0
6	Nevada	Reno	1
7	Colorado	Reno	1

7.4.67 is_set_env

Use the is_set_env function to verify that the specified system environment variable is set.

Syntax

```
is_set_env(<variable_name>)
```

Return value

int

Returns 1 if the environment variable is set, otherwise, returns 0.

Where

`<variable_name>`	The name of the environment variable. Ensure that you surround the variable name by single quotes.

Details

Example

Function	Results
`is_set_env('MODE')`	Returns 1 if the MODE variable has already been set; returns 0 if the MODE variable has not been set.

7.4.68 is_valid_date

Use the is_valid_date function to indicate whether an expression can be converted into a valid calendar date value.

Syntax

```
is_valid_date(<input_expression>,'<date_format>')
```

Return value

int

- 1 = The expression is not NULL and is valid.
- 0 = The expression is not NULL and is invalid.
- NULL = The expression is NULL.

Where

<input_expression>	The expression to be validated. If the expression does not resolve to a value of data type `varchar`, the software issues a warning that the value has been converted to a `varchar`.	
<date_format>	The string identifying the date format of the input string. Construct the date format using the following codes and other literal strings or punctuation:	
	DD	2-digit day of the month
	MM	2-digit month
	MONTH	Full name of month
	MON	3-character name of month
	YY	2-digit year
	YYYY	4-digit year

Details

Example

For example the following expression returns 0 because there is no such date as January 34th.

```
is_valid_date ('01/34/2002', 'mm/dd/yyyy')
```

Example

Function	Results
`is_valid_date (Orders.SubmitDate,'mm/dd/yyyy')`	Tests whether the string `Orders.SubmitDate` can be converted to a calendar date with the `mm/dd/yyyy` date format.

7.4.69 is_valid_datetime

Use the is_valid_datetime to indicate whether an expression can be converted into valid calendar date and time values.

Syntax

```
is_valid_datetime(<input_expression>,'<datetime_format>')
```

Return value

int

- 1 = The expression is not NULL and is valid.
- 0 = The expression is not NULL and is invalid.
- NULL = The expression is NULL.

Where

<input_expression>	The expression to be validated.	
<datetime_format>	The string identifying the datetime format of the input expression. Construct the datetime format using the following codes and other literal strings or punctuation:	
	DD	2-digit day of the month
	MM	2-digit month
	MONTH	Full name of month
	MON	3-character name of month
	YY	2-digit year
	YYYY	4-digit year
	HH24	2-digit hour of the day (00-23)
	MI	2-digit minute (00-59)
	SS	2-digit second (00-59)

Details

Example

For example the following expression returns 0 because there is no such hour as 26:

```
is_valid_datetime ('01/14/2002 26:56:09', 'mm/dd/yyyy hh24:mi:ss')
```

Example

Function	Results
`is_valid_datetime (Orders.Received,'mm/dd/yyyy hh24:mi:ss')`	Tests whether the string `Orders.Received` can be converted to the `mm/dd/yyyy hh24:mi:ss` datetime format.

7.4.70 is_valid_decimal

Use the is_valid_decimal to indicate whether an expression can be converted into a valid decimal value.

Syntax

```
is_valid_decimal(<input_expression>,'<decimal_format>')
```

Return value

int

- 1 = The expression is not NULL and is valid.
- 0 = The expression is not NULL and is invalid.
- NULL = The expression is NULL.

Where

`<input_expression>`	The expression to be validated.

`<decimal_format>`	A string indicating the decimal format of the input expression. Use pound characters (#) to indicate digits and a decimal indicator. If necessary, include commas as thousands indicators. For example, to specify a decimal format for numbers smaller than 1 million with 2 decimal digits, use the following string: `'#,###,###.##'`. To indicate a negative decimal number, add a minus "-" sign at the beginning or end of this value. For example, to test if the stock price difference can be converted to decimal format, use the following function: `is_valid_decimal (Stocks.Price_difference, '-###.##')`

Details

Example

Function	Results
`is_valid_decimal (Orders.Price,'###,###.##')`	Tests whether the string `Orders.Price` can be converted to decimal format.

7.4.71 is_valid_double

Use the is_valid_double function to indicate whether an expression can be converted into a valid double value.

Syntax

```
is_valid_double(<input_expression>,'<double_format>')
```

Return value

int

- 1 = The expression is not NULL and is valid.
- 0 = The expression is not NULL and is invalid.
- NULL = The expression is NULL.

Where

`<input_expression>`	The expression to be validated.
`<double_format>`	A string indicating the double format of the input expression. Use pound characters (#) to indicate digits and a decimal indicator. If necessary, include commas as thousands indicators. For example, to specify a double format for numbers smaller than 1 million with 2 decimal digits, use the following string: `#,###,###.##'`

Details

Example

Function	Results
`is_valid_double (Product.Weight,'###.###')`	Tests whether the string `Product.Weight` can be converted to double format.

7.4.72 is_valid_int

Use the is_valid_int function to indicate whether an expression can be converted into a valid integer value.

Syntax

```
is_valid_int(<input_expression>,'<int_format>')
```

Return value

int

- 1 = The expression is not NULL and is valid.
- 0 = The expression is not NULL and is invalid.
- NULL = The expression is NULL.

Where

`<input_expression>`	The expression to be validated.
`<int_format>`	The format specifying the thousands separator of the input expression. For example, to specify an integer format, use the following string: #.###.###'. Valid separators include the period (.) and the comma (,). However, you can use only one valid separator type in a format. Separator defaults to the comma (,) when none is specified.

Details

Example

Function	Results
`is_valid_int (QuarterResults.Volume,'###.###')`	Tests whether the string `QuarterResults.Volume` can be converted to the `###.###` integer format.

7.4.73 is_valid_real

Use the is_valid_real function to indicate whether an expression can be converted into a valid real value.

Syntax

```
is_valid_real(<input_expression>,'<real_format>')
```

Return value

int

- 1 = The expression is not NULL and is valid.
- 0 = The expression is not NULL and is invalid.
- NULL = The expression is NULL.

Where

`<input_expression>`	The expression to be validated.
`<real_format>`	A string indicating the real format of the input expression. Use pound characters (#) to indicate digits and a decimal indicator. For example, to specify a real format for numbers smaller than 1 million with 2 decimal digits, use the following string: `'#,###,###.##'`.

Details

Example

Function	Results
`is_valid_real (QuarterResults.Mean,'#,###.#####')`	Tests whether the string `QuarterResults.Mean` can be converted to real format.

7.4.74 is_valid_time

Use the is_valid_time function to indicate whether an expression can be converted into a valid time value.

Syntax

```
is_valid_time(<input_expression>,'<time_format>')
```

Return value

int

- 1 = The expression is not NULL and is valid.
- 0 = The expression is not NULL and is invalid.
- NULL = The expression is NULL.

Where

`<input_expression>`	The expression to be validated.	
`<time_format>`	The string identifying the time format of the input expression. Construct the time format using the following codes and other literal strings or punctuation:	
	HH24	2-digit hour of the day (00-23)
	MI	2-digit minute (00-59)
	SS	2-digit second (00-59)

Details

Example

Function	Results
`is_valid_time (Orders.ReceivedTime,'hh24:mi:ss')`	Tests whether the string `Orders.ReceivedTime` can be converted to the `hh24:mi:ss` datetime format.

7.4.75 isempty

Use the isempty function to indicate whether a nested table contains data.

Syntax

```
isempty(<table_name>)
```

Return value

int

The result of the content test:

- Returns 1 when the table does not contain data.
- Returns 0 when the table contains data.

Where

`<table_name>`	The fully qualified name of the nested table to test. A fully qualified name contains the parent table names up to the top level of the table in the current context. If you specify only 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.

Details

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 contains data, then Data Services considers the first nested table as not empty.

Example

Use the isempty function to determine if there are line items associated with a sales order. 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)
```

7.4.76 isweekend

Use the isweekend function to indicate whether a date corresponds to Saturday or Sunday.

Syntax

```
isweekend(<date1>)
```

Return value

int

The result of the date test:

- Returns 1 when the date is a Saturday or Sunday.
- Returns 0 when the date is not a Saturday or Sunday.

Where

`<date1>`	The value of type `date` or `datetime` to test.

Details

Example

Function	Results
`isweekend(hire_date)`	Tests whether the date in `hire_date` is a Saturday or Sunday.
`isweekend(SYSDATE)`	Tests whether the current date is a Saturday or Sunday.

7.4.77 jde_date

Converts an input Julian date stored in a J.D. Edwards database to the equivalent SAP Data Services date data type.

Syntax

```
jde_date(<jde_julian_date>)
```

Return Value

date

Where

`<jde_julian_date>`	The integer column in the input table that contains a J.D. Edwards Julian date.

Details

Use `<jde_date>` when Data Services does not automatically translate dates.

Example

Function	Results
`jde_date (099049)`	`2/18/1999`

Enter the function in the *Mapping* tab or the *Where* tab of the Query transform editor.

Note

Because this function takes an integer as input, map the incoming column to an integer column before applying the function.

Related Information

Date data type conversion

7.4.78 jde_time

Converts an input J.D. Edwards (JDE) number that represents time into an SAP Data Services character value that represents the twenty-four hour time equivalent.

Syntax

```
jde_time(<time_integer>)
```

Return value

char (8)

Where

`<time_integer>`	The JDE number that represents a time.

Details

Given a number representing time in J.D. Edwards, `jde_time` returns a Data Services character value that represents the equivalent twenty-four-hour time in `<HH24:MI:SS>` format:

- `<HH>` = hours
- `<MI>` = minutes
- `<SS>` = seconds

Enter the JDE time function in the *Where* tab or the *Mapping* tab of the Query transform.

Example

Function	Results
`jde_time(92513)`	`09:25:13`

Note

Because this function takes an integer as input, map the incoming column to an integer column before applying the function.

7.4.79 job_name

Use the job_name function to return the name of the object, such as a job, in which the call to this function exists.

Returns the name of the job in which the call to this function exists.

Syntax

```
job_name()
```

Return Value

varchar

Details

Example

```
print('Starting execution of Job: [job_name()] as user: [system_user_name()]');
```

7.4.80 Job_Run_ID

Use the job_run_id function to retrieve the job run ID for the current job execution.

Syntax

Retrieves the job run ID for the current job execution.

```
job_run_id()
```

Return value

Varchar (20)

7.4.81 julian

Use the julian function to convert a date to the integer julian value. The Julian value is the number of days between the start of the Julian calendar and the given date.

Syntax

```
julian(<date1>)
```

Return value

int

The Julian representation of the date.

Where

`<date1>`	The source value of type `date` or `datetime`.

Details

Example

The following example uses the to_date function to convert the string to a date using the stated format. Then, the julian function converts the date to the Julian representation of the date.

Function	Results
`julian(to_date('Apr 19, 1997', 'mon dd, yyyy'))`	729436

7.4.82 julian_to_date

Use the julian_to_date function to convert a given Julian value to a date.

Syntax

```
julian_to_date(<input_julian>)
```

Return value

date

The date that corresponds to the input Julian value.

Where

`<input_julian>`	An integer representing the Julian value to be converted.

Details

Example

Function	Results
`julian_to_date(Julian_Date)`	Converts the number indicated by `Julian_Date` to its date value.

7.4.83 key_generation

Use the key_generation function to generate the next value in a series, after first determining the last value in the series.

Syntax

```
key_generation (<table>, <key_column>, <key_increment>)
```

Return value

int

The column value found to meet the function requirements.

Where

`<table>`	The full path name of the file or full database specification of the table in which the `<key_column>` is located. Enclose this value in single quotation marks. Use a fully qualified table name that includes the datastore, owner, and table name. For example: oracle_ds.TIGER.sales. For Netezza 7.x multischema, include the datastore, owner, schema, and table name. For example: NZ7_ds.DSDEV.SCHEMA1.TABLE1. If you do not specify the schema name SCHEMA1, the function uses the default schema. For example: NZ7_ds.DSDEV.TABLE1.

`<key_column>`	A column with existing keys from which this function determines the largest existing key value. Enclose this value in single quotation marks.
`<key_increment>`	The integer increment used between key values that this function generates.

Details

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.

> **i Note**
>
> Data Services does not support the key_generation function when you use SAP tables as readers.

> **Example**

Function	Results
`key_generation('target_ds.dbo.sales','order_number',1)`	Looks for the last key value in the `order_number` column in the sales database and returns the largest value plus one.

7.4.84 last_date

Use the last_date function to return the last date of the month for a given date.

> **Syntax**

```
last_date(<in_date>)
```

Return Value

date

Where

`<in_date>`	The date for which the last date of the month is to be calculated.

Details

Example

Function	Returns
`last_date('1990.10.01')`	'1990.10.31'

7.4.85 least

Use the least function to return 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, Data Services converts the result into the return data type.

Where

`<expression_list>`	A list of one or more comma-separated expressions.

Details

Data Services implicitly converts expressions in the list to a normalized data type before comparison.

The software uses the following rules to determine the normalized data type:

1. If the return data type is varchar, then implicitly normalizes all expressions to varchar before comparison.
2. If the return data type is one of the date data types, then implicitly normalizes all expressions in the list to that data type before comparison.

   > **Example**
   >
   > 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 implicitly normalizes all the expressions to the highest precedence numeric expression in the list.

   > **Example**
   >
   > 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). The normalized data type is decimal.

   The software converts all the expressions in the list to decimal data types before comparison. If the normalized data type is decimal, then the precision is the lowest precision among all decimal data type expressions. The decimal data type expressions preserve their scale during implicit conversion. When the software converts an integer data type expression to a decimal data type, the scale is 0. When the software converts float, double, and varchar data types into decimal data types, the 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

7.4.86 length

Use the length function to return the number of characters in a given string.

Syntax

```
length(<value>)
```

Return value

integer

The number of characters in `<value>`.

Where

`<value>`	A string indicating the column name, variable, or other element whose length is calculated.

Details

Example

In the *Mapping* box of a query, use the length function to return the number of characters in each row of a column. With the OUTPUT field selected in the target schema of a query, enter the following statement in the *Mapping* box:

```
length(dal_emp.ename)
```

The software produces the following results:

Source column (dal_emp.ename)	Target column (output)
jones	5
nguyen	6
tanaka	6

7.4.87 literal

Use the literal function to return an input constant expression without interpolation.

Syntax

```
literal(<input>)
```

Return value

Same value as the value given for the input parameter but without interpolation.

Where

`<input>`	A constant expression of any data type.

Details

Data Services 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.

Replacing special characters is an issue with the match_pattern and match_regex functions because they require these special characters. If your `pattern_string` or `regular_expresion_pattern` parameter in these functions is a constant, you may want to disable interpolation. If so, use the literal function.

Example

For example, you want to match `$my_pattern` with the pattern `'PART[123]'`.

If you code it as `$my_pattern = 'PART[123]'; match_pattern(product,$my_pattern);`, the interpolation changes the pattern being matched to 'PART123'. However, if you code it as `$my_pattern = literal ('PART[123]'); match_pattern(product, $my_pattern);`, the return value is 1 because the pattern remains 'PART[123]'.

Alternatively, if you do not want to use a variable, you can code it as `match_pattern (product,'PART[123]');`. Then the software does not interpolate on the constant 'PART[123]'.

There is no runtime cost for the literal function. Data Services substitutes the constant expression at compile time.

Example

To match only PART1 or PART2 or PART3 using the match_pattern function, assign a pattern to a variable without interpolation. Use the literal function in the following type of expression:

```
$pattern = literal('PART[123]');
```

If you do not use the literal function, the value assigned to `$my_pattern` in the following sample is 'PART123'. That is 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 expected result.

```
$my_pattern=LITERAL('PART[123]');
print($my_pattern);
if (match_pattern('PART1',$my_pattern) <> 0)
print('Matched');
else
print('Not Matched');
```

7.4.88 ln

Use the ln function to return the natural logarithm of the given numeric expression.

Syntax

```
ln(<numeric_expression>)
```

Return Value

Float

If input is negative, return value is NULL.

Where

`<numeric_expression>`	Any numeric expression.

Details

Returns the natural logarithm of the given numeric expression.

Example

Function	Results
`ln(5.436563656918)`	1.693147

7.4.89 load_from_gcs_to_gbq

Use the `load_from_gcs_to_gbq` function to transfer data from a Google Cloud Storage into a Google BigQuery table.

Syntax

```
load_from_gcs_to_gbq("<datastore_name>", "<remote_file_name>",
"<table_name>", "<write_mode>", "<file_format>")
```

Before you can use this function, ensure that the Google BigQuery table exists. The table doesn't have to be in your repository, however.

Return value

int

Returns 1 if function is successful. Returns 0 if function isn't successful.

Where

`<datastore_name>`	Name of the Google BigQuery datastore.
`<remote_file_name>`	Name of the file to copy from the remote server in the format `gs://bucket/filename`. You can use wildcards.
`<table_name>`	Name of the Google BigQuery table in the format `dataset.table`.

`<write_mode>`	Optional. The write mode values are: • Append • Truncated Append is the default.
`<file_format>`	The format of the data files using one of the following values: • CSV: (Default value) • DATASTORE_BACKUP • NEWLINE_DELIMITED_JSON • AVRO • PARQUET • ORC **i Note** For information about additional requirements for file formats, see the Google Cloud documentation at https://cloud.google.com/bigquery/docs . Search for "Introduction to loading data from Cloud Storage".

Details

Data Services uses the local and remote paths and Google Cloud Storage protocol information from the file location object. The file location object is specified in the named datastore. After the table is in a Google BigQuery table, use it as a source in a data flow.

Example

You want to copy a JSON file in a Google BigQuery datastore on a remote server to a Google BigQuery table on your local server. Here are the specifics:

- **Google BigQuery datastore name:** `DS_GBQ1`
- **Remote file name: (JSON file name)** `gs://test-bucket_1229/from_gbq/json08_from_gbq.json`
- **Google BigQuery table name:** `test.json08`
- **Write mode:** Append
- **File Format:** NEWLINE_DELIMITED_JSON

You set up the following script:

```
load_from_gcs_to_gbq('DS_GBQ1', 'gs://test-bucket_1229/from_gbq/
json08_from_gbq.json', 'gbq_table.json08', 'append',
'NEWLINE_DELIMITED_JSON');
```

The following example contains a sample expression for each type of supported file format.

Example

CSV:

```
load_from_gcs_to_gbq('DS_CSV_GBQ', 'gs://bucket_1230/from_gbq/
csv_from_gbq.csv', 'gbq_csv.table', 'append', 'CSV');
```

DATASTORE_BACKUP:

```
load_from_gcs_to_gbq('DS_DATASTRORE_GBQ', 'gs://bucket_1231/from_gbq/
datastore_from_gbq.export_metadata', 'gbq_datastore.table', 'append',
'DATASTORE_BACKUP');
```

AVRO:

```
load_from_gcs_to_gbq('DS_AVRO_GBQ', 'gs://bucket_1232/from_gbq/
avro_from_gbq.avro', 'gbq_avro.table', 'append', 'AVRO');
```

PARQUET:

```
load_from_gcs_to_gbq('DS_PARQUET_GBQ', 'gs://bucket_1233/from_gbq/
parquet_from_gbq.parquet', 'gbq_parquet.table', 'append', 'PARQUET');
```

ORC:

```
load_from_gcs_to_gbq('DS_ORC_GBQ', 'gs://bucket_1234/from_gbq/
orc_from_gbq.orc', 'gbq_orc.table', 'append', 'ORC);
```

7.4.90 load_from_s3_to_redshift

Use the load_from_s3_to_redshift function to use the Redshift `COPY` command to copy data files from an Amazon Simple Storage Service (S3) bucket to a Redshift table.

Syntax

```
load_from_s3_to_redshift("<datastore name>", "<table name>", "<file location
name>", "<file name>", "<file options>")
```

Where

`<datastore name>`	Name of the Redshift datastore.

`<table name>`	Name of the target table. You can also specify the following: • `<table name>` = `<table name>` • `<table name>` = `<schema name>.<table name>` • `<table name>` = `<Redshift datastore name>.<schema name>.<table name>` • `<table name>` = `<Redshift datastore name>.<Alias name used in datastore>.<table name>`
`<file location name>`	Location of the S3 file.
`<file name>`	Fully qualified name of the Amazon S3 file to copy to the Redshift table. Wild cards are allowed.
`<file options>`	Optional. Use the following file options as applicable when copying a file: • *acceptanydate*: Accepts any date, even those with invalid formats, without throwing an error. • *acceptinvchars*: Replaces invalid UTF-8 characters. • *blankasnull*: Inserts null if the input data is blank. • *dateformat*: Defines the date format. For example, \'YYYY-MM-DD\'. • *delimiter*: Defines the column delimiter. For example, \'|\'. • *emptyasnull*: Inserts null if input data is empty. • *encoding*: Defines the data file encoding type. Valid values include utf8 (default), utf16, utf16le, and utf16be. • *encrypted*: Loads encrypted data files from S3. • *escape*: Removes escape (\) character. For example,`a\\b\\c` would be `a\b\c`. • *explicit_ids*: Data values must match the Identity format and Identity columns. • *fillrecord*: Fills null if any record is missed. • *ignoreblanklines*: Ignores blank lines. • *ignoreheader*: Skips the specified number rows as a file header. The default is 0. • *manifest*: Loads manifest data files from S3. • *maxerror*: Defines the maximum number of errors allowed. The default is 0. • *null as*: Defines the special null string • *removequotes*: Removes quotes from the data file. • *roundec*: Rounds up numeric values when the input value is greater than the scale defined for the column. • *timeformat*: Defines the timestamp format. For example, `\'YYYY-MM-DD HH:MI:SS\'`. • *trimblanks*: Removes whitespace characters. Only applies to the varchar data type. • *truncatecolumns*: Truncates data in columns when the input value is greater than the column defined. Applies to varchar or char data types and rows 4MB or less in size. • *gzip*: Loads compressed data files from S3. • *lzop*: Loads compressed data files from S3. • *bzip2*: Loads compressed data files from S3.

Details

Sample Code

```
CREATE __AL_REPO_FUNCTION load_from_s3_to_redshift("Datastore"  __FUNC_CHAR
IN, "Table name" __FUNC_CHAR  IN, "File location name"  __FUNC_CHAR  IN,
"File name"  __FUNC_CHAR  IN, "File options"  __FUNC_CHAR  IN )
SET(database_type = 'ACTA',
function_type = 'Miscellaneous_Function',
DB_FunctionName = 'load_from_s3_to_redshift',
Description = 'This function loads Amazon S3 data file(s) to a Amazon
Redshift table',
Parallelizable = 'NO',
External_name = 'load_from_s3_to_redshift',
return_param_dep = 'null',
return_datatype = '5',
return_datatype_size = '4',
param0 = 'Name of the Amazon Redshift datastore.',
param1 = 'Name of the target table.',
param2 = 'Name of the Amazon S3 File location.' ,
param3 = 'Fully qualified name of the Amazon S3 file(s). Wild cards are
allowed.' ,
param4 = 'File options that can be applied when copying the file. For
example, \'delimiter \',\' encoding \'utf8\'\'.'
)
```

Example

To copy a data file inside `<bucket name>/<sub directory>` on S3 to a Redshift table, define the following in the S3 datastore:

- Bucket = `<bucket name>`
- Remote Directory=`<sub directory>`

Then enter the following:

```
load_from_s3_to_redshift('redshift_ft', 'customer ', 'S3_to_Redshift',
'customer.dat', 'delimiter \',\' ');
```

Example

To generate an AES256 key, enter the following:

```
encrypt_aes('<plain password>', '<passphrase>', 256)
```

You can then use the key to upload data from the Redshift table to the S3 bucket.

```
unload ('select * from <redshift table>')
to 's3://<bucket name>/<sub directory>/'
credentials 'aws_access_key_id=<access key>;aws_secret_access_key=<secret
access key>;master_symmetric_key=<AES256 key> '
delimiter '|' encrypted bzip2;
```

To copy the encrypted data files on S3 back to a Redshift table, enter the following:

```
load_from_s3_to_redshift('redshift_ft', 'public.t31_household',
'S3_to_Redshift_3', 't31_encrypted', 'master_symmetric_key \'<AES256 key> \'
encrypted bzip2 delimiter \'|\'');
```

Example

To copy JSON data from S3 to a Redshift table, with a JSON path, enter the following:

```
load_from_s3_to_redshift('redshift_ft', 'public.t32_category',
'S3_to_Redshift_3', 't33_category.json', 'json \'s3://dsqa-redshift-bkt3/
t33_category_jsonpath.json\'');
```

Example

To copy CSV data from S3 to a Redshift table, enter the following:

```
load_from_s3_to_redshift('redshift_ft', 'public.t32_category',
'S3_to_Redshift_3', 't34_category_csv.txt', 'csv quote as \'%\'');
```

Example

To copy fixed-width data from S3 to a Redshift table, enter the following:

```
load_from_s3_to_redshift('redshift_ft', 'public.t35_fixed_width',
'S3_to_Redshift_3', 't35', 'fixedwidth \'catid:5,catgroup:10,catname:
9,catdesc:40\'');
```

Related Information

Amazon S3 file location protocol options

7.4.91 load_to_xml

Use the load_to_xml function to convert a nested relational data model (NRDM) table into XML and place the XML in a single column during a data load.

Syntax

```
load_to_xml (nested_table_name, schema_dtd_name, enable_validaton,
xml_header, replace_string_nulls, is_top_level_element, max_size)
```

Return value

If the function fails due to an error when trying to produce the XML output, Data Services returns NULL for the scalar columns that you select for output and returns an empty nested table for NRDM columns.

Example

```
load_to_xml(nested_table_name, billing_address_schema, 0, '<?xml version="1.0"
encoding="UTF-8"?>', NULL, 1, 4000)
```

Where

`nested_table_name`	The name of the NRDM table that you want to convert into an XML format.
`schema_dtd_name`	The name of the DTD or XML Schema that you want Data Services to use to format the XML text in the output column. Be sure to match the structure of the nested table to the structure provided by the DTD or XML Schema. Or match the structure of its only child. Otherwise, Data Services does not produce output. See the `is_top_level_element` for an example of specifying an only child. **i Note** Use Data Services to generate an XML Schema or DTD from an NRDM schema.
`enable_validation`	Validates that Data Services generates XML that matches the XML format you specify for schema_dtd_name. • Enter 1 to enable validation. • Enter 0 to disable validation.
`xml_header`	If the value specified is NULL, the header of the output XML has UTF-8 in the header. The default header that the software generates is: `<?xml version="1.0" encoding="UTF-8"?>` If this value is not null, Data Services replaces the default XML header with a header that you provide. Make sure the XML_header that you provide matches the encoding of the target datastore where you store the XML data.
`replace_null_string`	If this value is not null, Data Services replaces NULL values with the specified string.

`is_top_level_element`	• Enter `1` if the input column representing the NRDM table matches the root element of the given DTD or XML Schema. • Enter `0`, if the input column should match the only child of the root element of the DTD or XML Schema. **Example** 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` . When you select the input column name `PO` as the `nested_table_name`, it matches the root element of the schema, so set the value of `is_top_level_element` to `1`. **Example** Imagine that a database table called `employees` has a row for each employee. Also, you use an XML Schema called `employees.xsd`. The `employees.xsd` 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 `nested_table_name`, set the value for `is_top_level_element` to 0 because it does not match the root element of the schema `allEmployees`. The function returns data with `allEmployees` as the root element and the number of employee elements under it to match the number of rows in the input employees table.
`max_size`	The expected maximum size of the generated XML.

Related Information

XML extraction and parsing for columns
Generating DTDs and XML Schemas from Query mapping

7.4.92 local_to_utc

Use the local_to_utc function to convert the input datetime of any time zone to Coordinated Universal Time (UTC).

Syntax

```
local_to_utc(<input datetime>, <timezone of the input with UTC offset>)
```

Return Value

datetime

Details

The second parameter, UTC offset, is a constant value. If you do not provide the UTC offset value, then Data Services uses the time zone of the Job Server host to calculate the UTC offset.

Details

Example

Function	Results
`local_to_utc('2014.02.01 00:00:00', 'UTC+08:30')`	'2014.01.31 15:30:00'

7.4.93 log

Use the log function to return the base-10 logarithm of the given numeric expression.

Syntax

```
log(<num>)
```

Return Value

Float

If the input is negative, the return value is NULL.

Where

`<num>`	The number for which you want a base- 10 logarithm returned.

Details

Example

Function	Results
`log(100.000)`	2.000000

7.4.94 long_to_varchar

Use the long_to_varchar function to convert a data type value of a given column from long to varchar.

Syntax

```
long_to_varchar(<column_name>, <max_size, start_position>)
```

Return value

varchar

Where

`<column_name>`	The name of the column containing the long data type.
`<max_size>`	The maximum size for the converted data in the table column.
`<start_position>`	Optional. Starting character position from which data is converted. The default start position is 1. A negative number indicates that Data Services starts counting backwards from the last character.

Details

Example

```
long_to_varchar(content_column, 4000)

long_to_varchar(content_column, 4000, -5000)
```

Related Information

XML extraction and parsing for columns

7.4.95 lookup

Use the lookup function to retrieve a value in a table or file based on the values in a different source table or file.

Syntax

```
lookup (<lookup_table>, <result_column>, <default_value>, <cache_spec>,
<compare_column>, <expression>)
```

Return value

Any type

The value in the `<lookup_table>` that meets the lookup requirements. The return type is the same as `<result_column>`.

Where

<lookup_table>	The table or file that contains the result or value you are looking up (<result_column>). The Use a fully qualified table name that includes the datastore, owner, and table name. For example: oracle_ds.TIGER.sales. <compare_column> is also located in this table. For Netezza 7.x multischema, include the datastore, owner, schema, and table name. For example: NZ7_ds.DSDEV.SCHEMA1.TABLE1. If you do not specify the schema name, the software uses the default schema. For example: NZ7_ds.DSDEV.TABLE1. You might need to put the owner in quotes, particularly if you use lower case letters.
<result_column>	The column containing the values you want to retrieve. This column is in the table or file listed for <lookup_table>.
<default_value>	The value returned when there is no matching row in the table or file listed for <lookup_table>Use a fully qualified table name that includes the datastore, owner, and table name. For example:.
<cache_spec>	The caching method that the lookup operation uses. Enclose with single quotes. There are three possible settings: • NO_CACHE: Reads values from the <lookup_table> for every row without caching values. • PRE_LOAD_CACHE: Loads the <result_column> and <compare_column> into memory after applying filters but before executing the function. Select this option if the number of rows in the table is small or you expect to access a high percentage of the table values. • DEMAND_LOAD_CACHE: Loads <result_column> and <compare_column> values into memory as the function identifies them. Select this option if the number of rows in the table is large and you expect to access a low percentage of the table values frequently. Select this option when you use the table in multiple lookups and the compare conditions are highly selective, resulting in a small subset of data.
<compare_column>	The column in the <lookup_table> that the function uses to find a matching row. When the function reads a varchar column in the <lookup_table>, it does not trim trailing blanks.

<expression>	The value that the function searches for in the <compare_column>. The value can be a simple column reference, such as a column found in both a source and the <lookup_table>. The value can also be a complex expression given in terms of constants and input column references. When <expression> refers to a unique source column, you do not need to include a table name qualifier. If <expression> is from another table or is not unique among the source columns, you need a table name qualifier. If <expression> is an empty string, the function searches for a zero-length varchar value in the <compare_column>. The function ignores trailing blanks in comparisons of <expression > and values in <compare_column>.

> **i Note**
>
> You can specify more than one <compare_column> and <expression> pair. To specify more than one pair, add additional pairs at the end of the function statement. Ensure that the values match for all specified pairs in order for the lookup function to find a matching row.

Details

> **i Note**
>
> You cannot use this function with a J. D. Edwards datastore. Use the lookup_ext function instead.

The lookup function uses a value that you provide in <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 software provides only fields for one pair; add extra <compare_column> and <expression> pairs to the output.

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 that the lookup

function returns 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 finds the matching row with the maximum value in the `<result_column>` and returns that value.
- For SAP application tables or flat files, the lookup function randomly selects a matching row and returns the value in the `<result_column>` for that row.

> **i Note**
>
> To avoid random row selection when the `<lookup_table>` is an SAP application table or a flat file, we recommend that you use the lookup_seq() function.

To enhance performance, 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 that you have a source table containing a numerical identifier, such as an employee number, and you want to use the employee 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:

1. Select the column that you want to look up in the target schema.
2. Click the *Functions* button located over the *Mapping* text box. The function wizard opens.
3. Select *Miscellaneous_function* under *Function* categories.
4. Select *lookup* under *Function* name.
5. Click *Next*. The *Define Input Parameters* dialog box opens.
6. 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 following 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 lookup function returns the value from the GBSTA column in:

- the VBUP table that corresponds to the VBELN value in the VBAK table
- the POSNR value in the VBAP table

When the function does not fine a corresponding value, it returns "none."

7.4.96 lookup_ext

Use the lookup_ext function to retrieve data from a lookup table based on user-defined lookup conditions that match input data to the lookup table data.

Syntax

The following syntax includes line breaks for clarity.

> **Note**
>
> This function has a graphical editor.

```
lookup_ext(
[<lookup_table,cache_spec,return_policy>],
[<return_column_list>],
[<default_value_list>],
[<condition_list>],
[<orderby_column_list>],
[<output_variable_list>],
[<sql_override>]
set ("run_as_separate_process"='yes'),
("output_cols_info"='<?xml version="1.0"
encoding="UTF-8"?>
<output_cols_info>
<col index="1" expression="yes"/>
</output_cols_info>')
)
```

Return value

any type

The return type is the first lookup column in `<return_column_list>`.

Where

`<lookup_table>`	The table, file, or memory datastore that contains the results or values you are looking up (`<result_column_list>`). **Database table:** If the `<lookup_table>` is a database table, enter the value using the `<datastore.owner.table>` format. **Example** For example: ERP_ds.OWNER.EMPLOYEES For Netezza 7.x multischema, include the datastore, owner, schema, and table name in the value. **Example** For example: NZ7_ds.DSDEV.SCHEMA1.TABLE1. If you do not specify the schema name, SCHEMA1 in the example, the function uses the default schema. **Example** For example: NZ7_ds.DSDEV.TABLE1. **Flat file:** If the `<lookup_table>` is a flat file, use the `<file_ds.filename>` format when you enter the value. **Example** For example: delim."c:/temp/employees" To substitute a variable for a file name, replace the data inside the double quotes. **Example** For example `delim."$employees"`. The variable used to store a file name can be a local or global variable or a parameter passed to a work flow or a data flow. If the cache specification is NO_CACHE, SAP Data Services passes in a different file name each time it calls lookup_ext. **Example** For example, call lookup_ext in a WHILE loop and assign a different file name to the variable passed as the lookup file. If the cache specification is PRE_LOAD_CACHE or

	DEMAND_LOAD_CACHE, the software uses only the first file name that is passed. The software ignores all file names passed during subsequent calls. **Memory datastore table:** If the `<lookup_table>` is a memory datastore table, enter the value using the `<memory_ds.table>` format. **Example** For example: mem_ds.employees.

`cache_spec`	The caching method the lookup_ext function uses. You can select one of the following cache specifications:
	• **NO_CACHE:** Reads values from the `<lookup_table>` for every row without caching values.
	• **PRE_LOAD_CACHE:** Loads the `<return_column_list>`, `<compare_column>` (see `<condition_list>`), and `<orderby_column_list>` into memory after applying constant filters and before executing the function.
	Select this option if the number of rows in the table is small or you expect to access a high percentage of the table values.
	• **DEMAND_LOAD_CACHE:** Loads `<return_column_list>`, `<compare_column>` (see `<condition_list>`), and `<orderby_column_list>` into memory as the function identifies them.
	Select this option if:
	  • The number of rows in the table is large and you expect to frequently access a low percentage of table values.
	  • You use the table in multiple lookups and the compare conditions are highly selective, resulting in a small subset of data.

`return_policy`	Optional. Specify whether the software obtains the return columns from the smallest or the largest row based on values in the order by columns. Use `<return_policy>` when you expect multiple rows and want output data from one of the selected rows. The value can be: • MAX (default) • MAX-NS • MIN • MIN-NS MAX-NS and MIN-NS allow the lookup_ext function to treat NULL as the smallest value instead of the largest value.
`return_column_list`	Specify the names of output columns in the `<lookup_table>` in a comma-separated list. For a given output column in the lookup table, select the *Expression?* checkbox in the lookup_ext editor when some of the data is in the form of expressions. If you select the *Expression?* checkbox and the data begins with an equal sign (=), the software evaluates the data as an expression and returns the result. Otherwise, it returns the column value.
`<default_value_list>`	A comma-separated list containing the default expressions for the output columns. When no rows match the lookup condition, the software returns the default values for the output column. Ensure that each default expression type is compatible with the corresponding output column type such that, if the types are not exactly the same, automatic conversion is still possible. If `<default_value_list>` is empty or has fewer expressions than the number of output columns, the software returns NULL by default. You cannot have more default expressions than the number of output columns.

<condition_list>

A list of triplets that specify lookup conditions. Each set in a triplet contains <compare_column>, <compare_operator>, and <compare_expression>. The value for <compare_operator> can be:

- <
- <=
- >
- >=
- =
- !=
- IS
- IS NOT
- ~

The <compare_column> is a column from the <lookup_table>. The function compares the column against the value in <compare_expression> to compute the output row.

Write the <compare_expression> in terms of constants, variables, and columns in the calling data flow or scripts. While <compare_expression> cannot contain column reference from the <lookup_table>, it can be a simple constant, variable, column reference, or a complex expression involving arithmetic operations and function calls.

Use compare operators IS and IS NOT to examine <compare_column> against the NULL constant. When you use IS or IS NOT as the compare operator, ensure that the <compare_expression> contains the NULL constant. When you use other compare operators against a <compare_expression> that contains a NULL, the lookup condition return value is always FALSE.

Use the tilde (~) compare operator to indicate that the column from the lookup table contains a pattern. The required pattern tags in the lookup table are as follows:

- mp (<pattern>) : Indicates the match_pattern type of pattern syntax
- mr (<pattern>) : Indicates the match_regex type of pattern syntax
- ms (<pattern>) : Indicates the match_simple type of pattern syntax

If <compare_expression> is an empty string, the function looks up a zero length varchar value in the lookup table. The

function ignores trailing blanks in the `<compare_expression>`.

If you create more than one set of triplets, the function implicitly joins all triplets with the AND operator to compute the final lookup condition.

Example

```
[c1,
            '=',
10,c2,'<',query.a,c3,'>=',lower(que
ry.name)]
```

`<orderby_column_list>`

Optional. A list of column names from the `<lookup_table>` separated by commas.

Working with `<return_policy>`, the function uses the `<orderby_column_list>` to determine what row to return the output when more than one row satisfies the lookup condition. When multiple rows occur, the function sorts the list of rows based on columns from the `<orderby_column_list>` and choosing a row to return using the MIN/MAX `<return_policy>`.

The `<orderby_column_list>` is optional. If you leave it blank, the orderby columns match the output columns.

Example

With [c1,c3,c4], the function sorts the rows using column values in c1, c3, c4.

With [ ], the function interprets as an empty list. The function uses it as a placeholder for specifying subsequent parameters.

`<output_variable_list>`	Optional. A comma-separated list of output variables. When you specify more than one output column in the function call, the function uses the output variables to receive output returns. Variables and output columns are matched by position. This parameter is optional unless more than one output column appears in the `<return_column_list>`. In the case of more than one output column, ensure that output variables are equal in number to output columns. To enable conversion, ensure that the variable data type is compatible with the corresponding output column. You do not need to specify output variables if the function is called using the function wizard to map output columns in the query window. **Example** [$a,$b,$c]

<sql_override>

This parameter is available as a button called *Custom SQL* in the function wizard. Enter 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 setting in this parameter replaces the SQL SELECT statement generated internally by the function for populating the cache. Ensure that the SELECT statement selects at least the columns contained in <return_column_list>, <condition_list>, and <orderby_column_list>.

The function allows any valid SQL SELECT statement and you can also include references to other tables besides the <lookup_table> to specify inner and outer joins. <sql_override> can be specified only when the <lookup_table> is a database table.

The value you choose for <cache_spec> affects the way the software processes the function:

- **NO_CACHE:** The software executes the sql_override query each time the function is called.
- **NO_CACHE:** When you pass in a dynamic SQL statement in the form of a variable, the software dynamically executes the SQL.
- **PRE_LOAD_CACHE:** The software executes only the first sql_override query to populate the lookup cache. The software ignores all subsequent SQL statements after it builds the lookup cache.
- **DEMAND_LOAD_CACHE:** The software converts the caching mode to PRE_LOAD_CACHE and behaves accordingly.

Example

```
['select out1,

out2,compare1,compare2,orderby1,ord
erby2 from lookuptbl,othertbl
where c1=10
                        and
lookuptbl.c2=othertbl.c2']
```

The following table contains the SET options:

run_as_separate_process	• **Yes:** Runs each operation s a separate process that uses separate resources such as memory and computer, to improve performance and throughput. • **No:** Disables this option. No is the default setting. **Example** `SET ("run_as_separate_process"='yes')`
output_cols_info	• **Yes:** An output column contains an expression (when the *Expression* checkbox is selected). • **No:** An output column does not contain an expression (when the *Expression* checkbox is not selected). No is the default setting. **Example** Use the following syntax. Notice that the first column begins with index value of 1. `SET ("output_cols_info"='<?xml version="1.0" encoding="UTF-8"?> <output_cols_info> <col index="1" expression="yes"/> </output_cols_info>')`

Related Information

Lookup tables and the lookup_ext function
match_pattern [page 1216]
match_regex [page 1219]
match_simple [page 1224]
Run as separate process

7.4.96.1 Optimize lookup_ext using pushdown

Optimize the performance of the lookup_ext by pushing down the function processing to the database.

For best performance, the lookup_ext function can be pushed down to the database under the following conditions:

- Use the lookup_ext function 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 as the reader or from a linked datastore to the reader.
- You set the `<cache_spec>` value to NO_CACHE.
- You set the `<return_policy>` value to either MAX or MIN.
- All conditions that you use in `<condition_list>` are database expressions.
- You use only the equals operator (=) in the lookup `<condition_list>`.
- You set the `<run_as_separate_process>` SET option to No.
- You ensure that for lookups with multiple-result column values, the database supports the rank, or equivalent function.

When the database is SAP HANA, MySQL, SAP ASE, and Informix, you can utilize pushdown with restrictions. Because there is no analytic function support for these databases, use pushdown only for single-result columns, or for multiple-result columns for primary keys.

7.4.96.2 lookup_ext limitations

There are limitations that you should know about when you use the lookup_ext function.

- You cannot call the lookup_ext function from an ABAP data flow. Instead, use the lookup function.
- The caching mode is always NO_CACHE when you call the lookup_ext function in script objects, jobs, or work flows. Selecting NO_CACHE is necessary because the software cannot determine when to release the cache after executing the function.
- The following parameters in the lookup_ext function do not support long, blob, and NRDM data types:
  - `<default_value_list>`
  - `<orderby_column_list>`
  - `<condition_list>`

  Therefore, you cannot look up a long column or specify a column or expression of the long data type.
- If you do not set a value for an optional parameter, ensure that you add empty square brackets as an empty placeholder [ ] for the optional parameter when it is followed by another optional parameter for which you specify a value.
- Under certain circumstances, for best performance, use the equals operator to specify the lookup condition.

  > Example
  >
  > - When you set the caching mode to NO_CACHE, and the lookup table is a database table, the underlying DBMS uses fast access methods, such as index, to retrieve data based on an indexed key.
  > - When you set the caching mode to PRE_LOAD_CACHE, the equals comparison results in more efficient memory lookup than other comparison operators.
  > - When you set the caching method to PRE_LOAD_CACHE, the function pushes down any lookup condition that uses a constant expression to the database. The pushdown results in smaller lookup caches than the current lookup.
- When you set the lookup_ext function for pattern evaluation, your data flow could run out of memory. That is because pattern evaluation uses virtual memory and is not included as part of pageable cache when the

lookup table is cached. When your lookup table has many patterns, select the *Run as a separate process* option so the process runs separately from the rest of the function processes. Also consider using *Run as a separate process* when you use expressions in output columns because of the same caching issue.

7.4.97 lookup_seq

Use the lookup_seq function to retrieve a value in a table or file based on the values in a different source table or file and a sequence value.

Syntax

```
lookup_seq (<lookup_table>, <result_column>, <default_value>,
<sequence_column>, <sequence_expression>, <compare_column>, <expression>)
```

Return value

Any type

The value in the `<lookup_table>` that meets the lookup_seq requirements. The return type is the same as `<result_column>`.

Where

`<lookup_table>`	The table or file that contains the results or value you are looking up. Use a fully qualified table name that includes the datastore, owner, and table name. **Example** ERP_ds.OWNER.EMPLOYEES For Netezza 7.x multischema, include the datastore, owner, schema, and table name. **Example** NZ7_ds.DSDEV.SCHEMA1.TABLE1. If you do not specify the schema name, the function uses the default schema. **Example** NZ7_ds.DSDEV.TABLE1. The software caches the `<lookup_table>` automatically for the operation of the function. **Note** The `<sequence_column>` and `<compare_column>` are also located in the table or file that you specify for `<lookup_table>`.
`<result_column>`	The name of the column containing the values you want to retrieve. This column is in the `<lookup_table>`. When the column contains varchar values, the function does not trim trailing blanks.
`<default_value>`	The value returned when there is no matching row in the `<lookup_table>`.
`<sequence_column>`	The column in `<lookup_table>` that indicates the sequence of the row. This column may contain a date that indicates when new values were added to the row. For example, in some source tables, `<sequence_column>` is the `EFFDT` column, which indicates when the data in the row became effective.
`<sequence_expression>`	The value the function searches for in the `<sequence_column>` to find a matching row. **Example** To find changed rows in a dimension table as of the current date, use the return value from the sysdate function for the value in `<sequence_expression>`. If `<sequence_expression>` is an empty string, the function looks up a zero-length varchar value in the lookup table.

<compare_column>	The column in the <lookup_table> that the function uses to find a matching row.
<expression>	The value that the function searches for in the <compare_column>. This can be a simple column reference, such as a column found in both a source and the <lookup_table>. This can also be a complex expression given in terms of constants and input column references. When <expression> refers to a unique source column, you do not need to include a table name qualifier. If <expression> is from another table or is not unique among the source columns, you need a table name qualifier. If <expression> is an empty string, the function looks up a zero-length varchar value in the lookup table. The function ignores trailing blanks in comparisons of <expression > and values in <compare_column>.

> **i Note**
>
> You can specify more than one <compare_column> and <expression> pair. To specify more <compare_column> and <expression> values, add additional pairs at the end of the function statement. Ensure that the values match for all specified pairs so the lookup function can find a matching row.

Details

The lookup_seq function uses a value you provide in <expression> to find a corresponding value in a different file or table specified in <lookup_table>. When multiple rows match, the function uses the row sequence to determine the matching row.

More specifically, the function searches for the rows in the <lookup_table> where the value in the <compare_column> matches the value in <expression>. When the function finds multiple matching rows, it searches in the <sequence_column> for the row with the closest value less than or equal to the <sequence_expression>. If no row has a value less than or equal to the <sequence_expression>, the function finds the row with the closest value to the <sequence_expression>. For the matching row, the function returns the value in the <result_column>.

For example, if your source schema uses an employee ID to identify each row, and you want the employee's salary at the end of the previous year in your target schema, you can use the lookup_seq function to return the employee salary given the employee ID and the effective date of the salary. The salary returned will be the value corresponding to the latest effective date less than or equal to the value of <sequence_expression>.

In SQL terms, the lookup_seq function evaluates <expression> for each row, then determines which sequence column value meets the requirements:

```
SELECT MAX(<sequence_column>)
    FROM <lookup_table>, <source_table>
    WHERE <sequence_column> <= <sequence_expression>
        AND <compare_column> =
            <source_table.expression>)
```

Suppose this query stores the <sequence_column> value returned as <sequence_result>. Next, the function uses the <sequence_result> to find the proper <result_column>:

```
SELECT <result_column>
    FROM <lookup_table, source_table>
    WHERE <sequence_column> = <sequence_result>
        AND <compare_column> =
            <source_table.expression>
```

The value returned by these queries is the result of the lookup_seq function for the row.

You can specify multiple <compare_column> and <expression> pairs to uniquely identify the <result_column> value. However, the function wizard only provides fields for one pair; add extra <compare_column> and <expression> pairs to the output that the wizard generates.

Data Services always caches the comparison table when performing a lookup_seq function.

If the lookup_seq function does not find the value of <expression> in <compare_column>, then the function evaluates and returns the <default_value>.

Example

You can use the lookup_seq function to return a value from a slowly changing dimension table given an identifier. For example, suppose you have a source table that contains a numerical identifier, such as an employee number, and you want to retrieve the employee's salary at a specific time in the past. You can use the lookup_seq function to return the employee's salary on a particular date based on the employee number.

The source table contains the employee number and employee name.

You want the target table to contain the employee name and salary.

Use the lookup_seq function to translate the values from the source table to the desired values in the target table. The lookup_seq function uses a third, "translation," table.

To produce the desired target column, select the column in the target schema. Next, click the *Functions* button, located over the *Mapping* text box. In the function wizard, select *Miscellaneous_functions* under Function categories, then select *lookup_seq* under Function name. Enter the function parameters as follows:

Option	Value
Translate table	ODS_DS.SSB.FINANCE
Result column	Salary
Default value	'0'
Sequence column	EffectiveDate
Sequence Expression	'12.31.1999'
Compare column	EmployeeID
Expression	EmployID

The function wizard automatically produces the mapping text

```
lookup_seq(Ora_DS.RBH.FINANCE, SALARY, '0',
```

```
    EFFECTIVEDATE, '12.3.1.1999', Employee_ID, EmployID)
```

For each employee, this lookup_seq function returns the value from the salary column for that employee that is the most recent before December 31, 1999.

7.4.98 lower

Use the lower function to change the characters in a string to lower case.

Syntax

```
lower(<value>,'<locale>')
```

Return value

varchar

The lowercase string. The return type is the same as `<value>`. The function leaves any characters that are not letters unchanged.

Where

`<value>`	The string to be modified.
`<locale>`	Optional. Specify a locale to convert the string to the locale. **Note** The function supports the ISO 639 language code and the ISO 3166 country code formats.

Details

Example

Function	Results
`lower('Accounting101')`	'accounting101'

Function	Results
`upper((LastName,1,1))|lower(substr(LastName,2,LENGTH(LastName)))`	The value in column `LastName` with the first letter uppercase and the rest of the value lowercase. Note that this example does not account for two-word last names.
`lower(LastName,'tr')`	The value in column `LastName` is converted to all lowercase. It is also converted to the Turkish locale, using the ISO 639 language code.

7.4.99 lpad

Use the lpad function to pad the left side of a string with specific characters.

Syntax

```
lpad(<input_string>,<size>,'<pad_string>')
```

Return value

varchar

The modified string. The return type is the same as `<value>`. The function leaves any characters that are not letters unchanged.

Where

`<input_string>`	The string source, such as a column name.
`<size>`	An integer value indicating the number of characters in the return string.
`<pad_string>`	A character or set of characters that this function concatenates to `<input_string>`.

Details

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 returns a truncated string without adding special characters.

Example

Function	Results
`lpad('Tanaka', 15, ' ')`	`'  Tanaka'` Note The character in `<pad_string>` is a space.
`lpad(last_name, 25, ' ')`	The value in the column last_name, padded with spaces from the left to 25 characters. If the value in last_name exceeds 25 characters, truncates from the right.

7.4.100 lpad_ext

Use the lpad_ext function to pad the left side of a string with logical characters from a given pattern.

Syntax

```
lpad_ext(<input_string>,<size>,'<pad_string>')
```

Return value

varchar

The modified string. The return type is the same as `<value>`. The function leaves any characters that are not letters unchanged.

Where

`<input_string>`	The string source.
`<size>`	An integer value indicating the number of characters in the return string.
`<pad_string>`	A logical character or set of logical characters that this function concatenates to the `<input_string>`.

Details

The logical characters prohibit this function from getting pushed down to the database.

The function repeats the value in `<pad_string>` from the beginning of the input string until the final string is the length set in `<size>`. If the value in `<input_string>` is already longer than the expected length, then this function truncates the string from the right.

Example

Function	Results
`lpad_ext('Tanaka', 15, ' ')`	`'        Tanaka'`
`lpad_ext(last_name, 25, ' ')`	The value in the column last_name, padded with spaces to 25 characters on the left. If the string alone exceeds 25 characters, truncates the string to 25 characters from the right.

Example

The lpad_ext and lpad functions exhibit the same behavior when the software evaluates the functions. However, the database behavior is different when the software pushes the function down to the database and the value in `<input_string>` and—or `<pad_string>` contain multibyte characters.

Function	Input	Output
lpad	`lpad("abc¶¶", 10,' ')`	`'   abc¶¶'`
lpad_ext	`lpad_ext("abc¶¶", 10,' ')`	`'     abc¶¶'`
lpad	`lpad("abc¶¶", 10,'')`	`'abc¶¶'`
lpad_ext	`lpad_ext("abc¶¶", 10,'')`	`'abc¶¶'`
lpad	`lpad("abcd", 10, '¶')`	`'¶¶¶abcd'`
lpad_ext	`lpad_ext("abcd", 10, '¶')`	`'¶¶¶¶¶¶abcd'`

7.4.101 ltrim

Use the ltrim function to remove specified characters from the start of the string.

Syntax

```
ltrim(<input_string>, <trim_string>)
```

Return value

varchar

The modified string. The return type is the same as `<input_string>`.

Where

`<input_string>`	The string to be modified.
`<trim_string>`	The characters to remove from `<input_string>`.

Details

The ltrim function is case-sensitive.

The function scans `<input_string>` left-to-right removing all characters that appear in `<trim_string>` until it reaches a character not in `<trim_string>`.

Example

Function	Results
`ltrim('Marilyn', ' ')`	'Marilyn'
`ltrim('ABCABCD', 'ABC')`	'D'
`ltrim('ABCABCD', 'EFG')`	'ABCABCD'
`ltrim('ABCDEABCDE', 'ABC'`	'DEABCDE'

Example

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. You may also use the ltrim_blanks or ltrim_blanks_ext functions for this.

7.4.102 ltrim_blanks

Use the ltrim_blanks function to remove blank characters from the start of a string.

Syntax

```
ltrim_blanks(<input_string>)
```

Return value

varchar

The modified string. The return type is the same as `<input_string>`.

Where

`<input_string>`	The string to be modified.

Details

Example

Function	Results
`ltrim_blanks(' Marilyn')`	'Marilyn'
`ltrim_blanks(last_name)`	The value contained in the column last_name, with all leading blanks removed.

7.4.103 ltrim_blanks_ext

Use the ltrim_blanks_ext function to remove blank and control characters from the start of a string.

Syntax

```
ltrim_blanks_ext(<input_string>)
```

Return value

varchar

The modified string. The return type is the same as `<input_string>`.

Where

`<input_string>`	The string to be modified.

Details

Example

Function	Results
`ltrim_blanks_ext(' Marilyn')`	'Marilyn'
`ltrim_blanks_ext(last_name)`	The value contained in the column last_name, with all leading blanks and control characters removed.

7.4.104 mail_to

Use the mail_to function to e-mail a specified number of lines in a trace and error log file to specified recipients.

Syntax

```
mail_to(<recipients_list>, <subject>, <message>, <number_of_trace_lines>,
<number_of_error_lines>)
```

Return value

int

Returns:

- 0: Function suceeded.

- 1: Function failed.

0 if function succeeds. Returns an integer that is not 0 if function fails.

Where

`<recipients_list>`	A string containing one or more recipient e-mail addresses separated by commas (,). This string cannot be empty. Ensure that it contains valid qualified e-mail address information.
`<subject>`	Optional. A string containing the subject of the e-mail. This string can be empty.
`<message>`	Optional. A string containing the e-mail message. This string can be empty.
`<number_of_trace><_lines>`	The number of lines from the end of the trace log file to append to the end of the e-mail. This input cannot be empty.
`<number_of_error><_lines>`	The number of lines from the end of the error log file to append at the end of the e-mail. This input cannot be empty.

Details

The function performs the following tasks:

- Captures the specified number of lines in the trace log and error log.
- Packages the information as e-mail.
- Uses your Job Server computer mail client to send e-mail messages to your local mail server for standard e-mail processing.

Use this function only within a script.

Before you use this function, ensure that the computer on which your Job Server is installed has a mail client. Also ensure that the logon account for the mail client has the same user name and password as the SAP Data Services service. The type of client varies by the operating system. The following lists the client type for each operating system:

- MAPI (message application programming interface): Applicable for Job Servers running on a computer with the Windows operating system. Ensure that the mail client is configured as the default mail client. For example, on Windows, make Microsoft Outlook your default mail client. Microsoft Outlook is a MAPI-based mail client.
- Mailx: Applicable for Job Servers running on a computer with the UNIX operating system.

> **i Note**
>
> If you do not have the required mail client, contact SAP Business User Support for assistance.

Example

Function	Results
`$myvar = mail_to('admin@company.com', 'Out of memory error in the SalesFact job. Please fix the error before running recovery job.',' ', 10, 10)`	The message is sent to one recipient. It has a subject line but no message. The e-mail contains 10 lines from the trace log and 10 lines from the error log.
`$myvar = mail_to('admin@company.com, manager@company.com', 'Out of memory error:' || systime(), 'Out of memory error while running the data flow:' || $dataflow_name || ' in the job:' || $job_name || '.', 10, 20)`	The message is sent to two recipients. The job name and data flow names are included in the text of the message as variables. Note that the software trims blank spaces from the end of strings; this example includes a blank on the beginning of the next string. You can also concatenate a string with a single blank. In the script, type "$a = ;" where $a is the local integer variable defined in the work flow. Put the cursor just ahead of the semicolon before clicking the *Functions* button to construct a mail_to statement.

Note

If your mail system is compatible with the software mail_to function, you can use e-mail nicknames (comma separated) as values in the `<recipients_list>`. When you use nicknames, 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 for Windows 2000

The mail_to function might not work properly on Windows 2000. The malfunction may occur after downloading a security patch for Microsoft Outlook that includes e-mail prompts. If you experience this problem, you have two options:

- Change the Outlook security settings to suppress prompts.
- Use the smtp_to function instead of the mail_to function.

7.4.105 match_pattern

Use the match_pattern function to match a whole input string to simple patterns supported by the software.

Syntax

```
match_pattern(<input_string>,<pattern_string>)
```

Return Value

int

Returns:

- 1: Pattern matched
- 0: Pattern did not match

Where

`input_string`	String to be matched. Supports UNICODE characters.
`pattern_string`	Pattern to find in the whole input string. Create `<pattern_string>` using characters listed in the following table. The function does not support substring matches.

Details

Characters for pattern_string

X	Represents uppercase characters. Unicode 4.2 General Category Values specification. Key = Lu, uppercase letter (For example, Latin, Greek, Cyrillic, Armenian, Deseret, and archaic Georgian.)
x	Represents non uppercase characters. Unicode 4.2 General Category Values specifications keys: • Ll = Lowercase letter (For example, Latin, Greek, Cyrillic, Armenian, Deseret, and archaic Georgian.) • Lt = Titlecase letters (For example, Latin capital letter D with small letter Z.) • Lm = Modifier letter (For example acute accent, grave accent.) • Lo = Other letter (Includes Chinese, Japanese, and so on.)
9	Represents numbers.
\	Escape character.
*	Any characters occurring zero or more times.
?	Any single character occurring once and only once.
[ ]	Any one character inside the braces occurring once.
[!]	Any character except the characters after the exclamation point. For example, [!12] can allow any number that does not start with a 1 or 2.

All other characters represent themselves. To specify a special character as itself, use an escape character. For example, [!9] means any character except a digit. To specify any digit except 9, use [!\9].

The following table displays pattern strings that represent example values:

Example pattern strings

Example value	Pattern string
Henrick	Xxxxxxx
DAVID	XXXXX
Tom Le	Xxx Xx
Real-time	Xxxx-xxxx
JJD)$@&*hhN8922hJ7#	XXX)$@&*xxX9999xX9#
1,553	9,999
0.32	9.99
-43.88	-99.99
Returns names with last name Jones	*Jones
Returns Henrick1 or HenrickZ	Henrick?
Returns David1 or David2 or David3	David[123]

Example

Use the `match_pattern` function in the Validation transform or in a WHERE clause of a Query transform. The input string can be from sources such as columns, variables, or constant strings.

Use case	Pattern	Function call in a script	Results
To match a zip code except one that begins with 1 or 2.	'[!12]9999'	`if (match_pattern('15014', '[!12]9999') <> 0) print('matched'); else print('not matched');`	Function prints “not matched”.
To match a zip code except one that begins with 1 or 2.	'[!12]9999'	`if (match_pattern('55014', '[!12]9999') <> 0) print('matched'); else print('not matched');`	Function prints “matched”.

Use case	Pattern	Function call in a script	Results
To process only customer phone numbers that fit the same pattern.	'999-999-9999'	`WHERE MATCH_PATTERN(CUSTOMER.PHONE_NUM,'999-999-9999') <> 0`	Phone numbers that do not match the pattern throw error 0.
To check a string against a complex pattern and print result to trace log.	'XXX)$@&*xxX9999xX9#'	`if(match_pattern('JJD) $@&*hhN8922hJ7#', 'XXX) $@&*xxX9999xX9#') <> 0) print ('matched'); else print('not matched');`	The result for this call is "matched".

7.4.106 match_regex

Use the match_regex function to match whole input strings to the pattern that you specify with regular expressions and flags.

Syntax

```
match_regex (<input_string>, <regular_expression_pattern>, <flags>)
```

Return Value

int

Returns:

- 1 = Pattern matched
- 0 = Pattern does not match

Where

`<input_string>`	String to be matched. The function supports UNICODE characters.

`<regular_expression_pattern>`	Pattern you want to find in a whole input string. The function does not match substrings. Provide the pattern in regular expression format with a varchar data type.
`<flags>`	Allows you to specify additional behavior that you want to occur while Data Services searches the input_string for pattern matches. Specify a flag, or specify more than one flag separated by commas. Ensure that you enter flags using the correct case because flag options are case sensitive. If you do not want a flag, enter NULL.

Details

Use POSIX standards when you enter regular expressions. 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 that we list in the following tables adhere to the current standard. For more information and updates, see "Regular Expressions" in the International Components for Unicode (ICU) User Guide at https://unicode-org.github.io/icu/userguide/ .

Use the regular expression patterns in the following table for the `<regular_expression_pattern>` argument.

Character	Description
\a	Match a BELL, \u0007.
\A	Match at the beginning of the input. Differs from ^ in that \A does not match after a new line within the input.
\b, outside of a [Set]	Match if the current position is a word boundary. Boundaries occur at the transitions between word (\w) and non word (\W) characters, with combining marks ignored. For better word boundaries, see ICU Boundary Analysis.
\b, within a [Set]	Match a BACKSPACE, \u0008.
\B	Match if the current position is not a word boundary.
\cX	Match a control-X character.
\d	Match any character with the Unicode General Category of Nd (Number, Decimal Digit).
\D	Match any character that is not a decimal digit.
\e	Match an ESCAPE, \u001B.
\E	Terminates a \Q ... \E quoted sequence.

Character	Description
\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\f\r\p{Z}].
\S	Match a non white space character.
\t	Match a HORIZONTAL TABULATION, \u0009.
\uhhhh	Match the character with the hex value hhhh.
\Uhhhhhhhh	Match the character with the hex value hhhhhhhh. Provide exactly eight hex digits, even though the largest Unicode code point is \U0010ffff.
\w	Match a word character. Word characters are [\p{Ll}\p{Lu}\p{Lt}\p{Lo}\p{Nd}].
\W	Match a non word character.
\x{hhhh}	Match the character with hex value hhhh. From one to six hex digits may be supplied.
\xhh	Match the character with two digit hex value hh.
\X	Match a Grapheme Cluster.
\Z	Match if the current position is at the end of input, but before the final line terminator, if one exists.
\z	Match if the current position is at the end of input.
\n	Back Reference. Match whatever the nth capturing group matched. n must be a number > 1 and < total number of capture groups in the pattern. Note: Octal escapes, such as \012, are not supported in ICU regular expressions.

Character	Description
[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 require quotes to be treated as literals are * ? + [( ) { } ^ $ | \ . /

Use the regular expression operators in the following table for the `<regular_expression_pattern>` argument.

Operator	Description
|	Alternation. A|B matches either A or B.
*	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 time. 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 time. 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 time. Possessive match.

Operator	Description
{n}+	Match exactly n times.
{n,}+	Match at least n times. Possessive match.
{n,m}+	Match between n and m times. Possessive match.
(...)	Capturing parentheses. Range of input that matched the parenthesized subexpression is available after the match.
(?: ...)	Non-capturing parentheses. Groups the included pattern, but does not provide capturing of matching text. Somewhat more efficient than capturing parentheses.
(?> ...)	Atomic-match parentheses. First match of the parenthesized subexpression is the only one tried; if it does not lead to an overall pattern match, back up the search for a match to a position before the "(?>"
(?# ...)	Free-format comment (?# comment).
(?= ...)	Look-ahead assertion. True if the parenthesized pattern matches at the current input position, but does not advance the input position.
(?! ...)	Negative look-ahead assertion. True if the parenthesized pattern does not match at the current input position. Does not advance the input position.
(?<= ...)	Look-behind assertion. True if the parenthesized pattern matches text preceding the current input position, with the last character of the match being the input character just before the current position. Does not alter the input position. The length of possible strings matched by the look-behind pattern must not be unbounded (no * or + operators).
(?<! ...)	Negative look-behind assertion. True if the parenthesized pattern does not match text preceding the current input position, with the last character of the match being the input character just before the current position. Does not alter the input position. The length of possible strings matched by the look-behind pattern must not be unbounded (no * or + operators).
(?ismx-ismx: ...)	Flag settings. Evaluate the parenthesized expression with the specified flags enabled or disabled.
(?ismx-ismx)	Flag settings. Change the flag settings. Changes apply to the portion of the pattern following the setting. For example, (?i) changes to a case insensitive match.

Use the flags in the following table for the `<flag>` argument.

Flag Options	Description
'CASE_INSENSITIVE'	If set, matching takes place in a case-insensitive manner.
'COMMENTS'	If set, allows use of white space and #comments within patterns.

Flag Options	Description
'DOTALL'	If set, a "." in a pattern matches a line terminator in the input text. By default, it will not. Note that a carriage return or line feed pair in text behave as a single line terminator and match a single "." in a regular expression pattern.
'MULTILINE'	If set, the function treats the input string as multiple lines instead of a single line. The '^' and '$' characters apply to each line in the input string instead of the entire input string.
'NO_PUSHDOWN'	If set, the match_regex function is processed but is never pushed down to the database.

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 transform. The input string can be from sources such as columns, variables, or constant strings.

Use Case	Pattern	Function Call
To match phone numbers in (408)-933-6000 format.	'([0-9]{3}-[0-9]{3}-[0-9]{4}'	`match_regex (pho_number,'([0-9]{3}-[0-9]{3}-[0-9]{4}',NULL)`
To match a string that starts with "topicA" regardless of case.	'topicA.*'	`match_regex (subject, 'topicA.*','CASE_INSENSITIVE')`

7.4.107 match_simple

Use the match_simple function to match a whole input string to simple patterns supported by the software for this function.

Syntax

```
match_simple(<input_string>,<pattern_string>)
```

Return Value

int

Returns:

- 1 = Pattern matches
- 0 = Pattern does not match

Where

`<input_string>`	String to be matched. Supports UNICODE characters.
`<pattern_string>`	Pattern you want to find in a whole input string.

Details

The match_simple function does not match substrings.

Use the characters in the following table for `<pattern_string>`.

.	Represents any single character.
*	Represents any character zero or more times.
#	Represents any single alphabetic character including non-English letters.
$	Represents any alphabetic character, including non-English letters, zero or more times.
+	Matches the previous character one or more times.
(string)+	Matches the string one or more times.
[number1..number2]	Numeric range (integers only). Matches any number between number1 and number2.
\	Escape character
;	OR operator. If the data matches any of the identified patterns, the result is TRUE. Enclose the list with curly brackets {}. Example: `{ABC+;XYZ*}` If the data matches either `ABC+` or `XYZ*`, the result is TRUE.
<>	NOT operator. Specify the pattern after the <>. Example: `<><pattern>`
{EMPTY} and {empty}	Special predefined patterns that match empty data.
{NULL} and {null}	Special predefined patterns that match NULL data.

If the pattern is empty, then the function matches all data.

If the value of a pattern column is NULL, then the function does not match with any value.

All other characters represent themselves. If you want to specify a special character as itself, then use an escape character.

Example

Example patterns

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 transform. The input string can be from sources such as columns, variables, or constant strings. The following example illustrates sample code used in a script.

Use Case	Pattern	Function Call	Results
To match account numbers from ACCT1 to ACCT5000	ACCT[1..5000]	`if (match_simple('ACCT 14', 'ACCT[1..5000]')) <> 0 print ('matched'); else print('not matched');`	Function prints "matched".

7.4.108 max

Use the max function to return the maximum value from a list.

Syntax

```
max(<value_list>)
```

Return value

Any type

The maximum value of the column values. The return type is the same as the values in `<value_list>`.

Where

`<value_list>`	The source values for which to identify a maximum.

Details

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.

7.4.109 min

Use the min function to return the minimum value from a list.

Syntax

```
min(<value_list>)
```

Return value

Any type

The minimum value of the column values. The return type is the same as the values in `<value_list>`.

Where

`<value_list>`	The source values for which to identify a minimum.

Details

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.

7.4.110 mod

Use the mod function to return the remainder when one number is divided by another.

Syntax

```
mod(<numerator>, <denominator>)
```

Return Value

integer

Where

`<numerator>`	Integer to be divided.
`<denominator>`	Divisor of first integer.

Details

Note

The % operator from SAP Information Steward syntax produces the same result.

Example

Function	Result
`mod(10,3)`	1
`mod(17,5)`	2
`mod(10,5)`	0

7.4.111 month

Use the month function to determine the month in which the given date falls.

Syntax

```
month(<date1>)
```

Return value

int

The number from 1 to 12 that represents the month component of `<date1>`.

Where

`<date1>`	The source date.

Details

Example

Function	Results
`month(to_date('Jan 22, 1997', 'mon dd, yyyy'))`	1
`month(to_date('3/97', 'mm/yy'))`	3

7.4.112 num_to_interval

Use the num_to_interval function to convert a numeric value to an interval.

Syntax

```
num_to_interval(<number1, format>)
```

Return value

interval

The converted interval.

Where

`<number1>`	The value of type `int`, `real`, `decimal`, or `numeric` to convert.
`<format>`	A string describing the format of the interval. Choose from the following values: D: Days H: Hours M: Minutes S: Seconds

Details

Example

Function	Results
`num_to_interval(elapsed_days, 'D')`	The value from the column `elapsed_days` converted to an interval of days.
`start_time + num_to_interval(elapsed_seconds, 'S')`	This example assumes that the function is acting on an input schema that 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. The example indicates a time that is the number of elapsed seconds after the start time ('2005-12-01 00:03:20').

7.4.113 nvl

Use the nvl function to replace NULL values with a given value.

Syntax

```
nvl(<expression1>, <replacement_value>)
```

Return value

Any type

The value of `<expression1>` if not NULL, otherwise, the value of `<replacement_value>`.

Where

`<expression1>`	The value to be tested for NULL.
`<replacement_value>`	The value to replace `<expression1>` if `<expression1>` is NULL. Ensure that `<replacement_value>` is the same data type as `<expression1>`.

Details

Example

Function	Results
`nvl(modification_date, sysdate())`	If the column `modification_date` for a row hasn't been set, this function inserts the current date.
`nvl(lookup(r3..vbpa, kunnr, 'NULL', vbeln, vbak.vbeln, posnr, vbap.posnr, parvw, 'RE'), lookup(r3..vbpa, kunnr, 'NULL', vbeln, vbak.vbeln, posnr, vbap.posnr, parvw, 'RG'))`	Both expressions are determined by the result of lookup functions.

7.4.114 power

Use the power function to return the value of the given expression to the specified power.

Syntax

```
power(<num>, <num>)
```

Return Value

Float data type

Where

`<num>`	Numeric expression representing a base number.
`<num>`	Numeric expression representing the power.

Details

Example

Function	Results
`power(2.2,3));`	10.648000

7.4.115 previous_row_value

Use the previous_row_value function to return the column value of the previous row.

Syntax

```
previous_row_value(<expression>)
```

Return Value

Data type of the input parameter. First row always returns `NULL`.

Where

`<expression>`	Valid Input expression.

Details

Each call to the previous_row_value() function returns the value stored during the previous call of this function. If the function is not called for each row, the results of this function might not be what you expect because it may not be the previous row value.

Example

This scenario can happen for example, if you use the previous_row_value() inside an ifthenelse() function: `If_then_else (table1.status = 'new', 0 , previous_row_value(table1.value))`

A better solution to the scenario is to use the following expression: `If_then_else (table1.status = 'new', 0 , 1) * previous_row_value(table1.value)`

Alternately, use two queries: One for the previous_row_value() and one for the final result including the if_then_else().

Example

The previous_row_value function is useful in Query transform. 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

The following is a list of records of sales figures for a series of days. Each record lists the record number, date, and revenue.

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 "order by Date" and subtracts the previous row revenue from the current row revenue.

Results:

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

7.4.116 print

Use the print function to print a given string to the trace log.

Syntax

```
print('<input_string>')
```

Return value

int

Value is `<input_string>` when the string contains valid data. Value is NULL and no string prints when the string contains NULL data.

Where

`<input_string>`	The message to be written to the trace log.

Details

Example

Function	Results
`print('Reached decision point for running full or incremental data flows')`	Writes "Reached decision point for running full or incremental flows" to trace log and returns `<input_string>`.
`print('The date is: [$start_date]')`	Writes "The date is 2000.06.03" to trace log and returns `<input_string>`.
`print('[$month_sal*12]')`	Writes "48000" to trace log and returns `<input_string>`.
`print('Total Sal is: [$month_sal*12]');`	Writes "Total Sal is: 48000" to trace log and returns `<input_string>`.
`print('The return value from the SQL() function is > [$y]');`	Writes "The return value from the SQL() function is > 23456" to trace log and returns `<input_string>`.

7.4.117 pushdown_sql

Use the pushdown_sql function to create dynamic WHERE clauses.

Syntax

```
pushdown_sql (<datastore>, <input_string>)
```

Return Value

None

Where

`<datastore>`	The name of the datastore containing the data you want to retrieve. Data Services creates a WHERE clause and pushes it down to this database. Surround the datastore name by single quotes.
`<input_string>`	A character string that forms the WHERE clause. Surround the character string by single quotes. Typically, the `<input_string>` 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, you do not prespecify the WHERE clause. The pushdown_sql function is particularly useful in real-time jobs to select data based on input from an XML message.

Unlike other functions, the pushdown_sql function can be used only in the WHERE clause of a Query transform. You cannot use the pushdown_sql function in other places, such as in a query mapping, in a conditional, or in a script. Using the pushdown_sql function in the Query transform ensures that Data Services pushes the WHERE clause that it creates from the function to the database.

Because Data Services does not parse the SQL contained in the input string, ensure that the input is well formed with the correct syntax.

i Note

Data Services does not allow use of the backslash as an escape character to delimit curly braces within the pushdown_sql function. If your input string contains a curly brace, ensure that the string is a variable. Therefore, instead of entering 'a\{b\}c', pass your data through as 'a{$x}c' where $x = '\{b\}'.

Example

You have a datastore named EC_DS that contains the table BIKES. BIKES stores information about different bike models. 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
```

When you set up your real-time data flow, use the pushdown_sql function in a query to select data from the BIKES table based on the data in the XML_IN message. Return the data in another XML message. The following table shows the function and the results.

Function	Results
`pushdown_sql ('EC_DS', '{XML_IN.QUERY_REQUEST}')`	Data Services includes the following WHERE clause in the SQL SELECT statement: `WHERE TYPE = 'MOUNTAIN' and PRICE < 1500`

7.4.118 quarter

Use the quarter function to determine the quarter in which the given date falls.

Syntax

```
quarter(<date1>)
```

Return value

int

The number from 1 to 4 that represents the quarter component of `<date1>`.

Where

`<date1>`	The source date.

Details

Example

Function	Results
`quarter(to_date('Jan 22, 1997', 'mon dd, yyyy'))`	1
`quarter(to_date('5/97', 'mm/yy'))`	2

7.4.119 raise_exception

Use the raise_exception function to generate an exception message for the Job Server error log..

Syntax

```
raise_exception(<error_msg>)
```

Return Value

int

Always returns 1.

Where

<error_msg>	The string that the software writes to the Job Server error log.

Details

If you surround the function with a try—catch block, the work flow or job may or may not terminate based on how you set the block.

Example

```
ifthenelse(sal < 1000000, 0, raise_exception('Salary exceeds 1 million
dollars.'))
```

7.4.120 raise_exception_ext

Use the raise_exception_ext function generates an exception with an exit code.

Syntax

```
raise_exception_ext(<error_msg>, <exit_code>)
```

Return Value

int

Always returns 1.

Where

`<error_msg>`	The string that the software writes to the Job Server error log.
`<exit_code>`	If the exception is not caught in a try—catch block, the code with which the job exits. Use a number in range 1 to 255 (zero means "success" to all operating systems).

Details

The software may or may not terminate the work flow or job may 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))
```

7.4.121 rand

Use the rand function to return a random number between 0 and 1.

Syntax

```
rand()
```

Return value

real

The random number between 0 and 1.

Example

Function	Results
`100 * rand()`	The function multiplies the random number by 100. The result is a random number between 0 and 100.

7.4.122 rand_ext

Use the rand_ext to return a random number between 0 inclusive and 1 exclusive.

Syntax

```
real rand_ext(<seed>)
```

Return value

real

The random number. The return value is between 0 and 1.

Where

`<seed>`	(Optional) Can be any positive integer greater than or equal to 0. If unspecified, the software uses the current time to create a seed.

Details

Similar to, and more powerful than the rand function. 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, the software 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. Specifying the same seed number is useful in repeat experiments.

Example

Function	Results
`100 * rand_ext()`	A random number between 0 and 100.

7.4.123 regex_replace

Use the regex_replace function to match a whole input string to a pattern that you specify with regular expressions. The function flags and replaces the matching portion of the input string with a replacement string.

Syntax

```
regex_replace(<input_string>, <regular expression pattern string>,
<replacement string>, <regular expression processing flags>)
```

Return Value

String

Where

`<input_string>`	String to be replaced. Supports UNICODE characters.
`<regular_expression_pattern_string>`	Pattern you want to find in a whole input string. The function does not support substring matches.
`<replacement_string>`	String to replace the `<input_string>`.
`<regular_expression_processing_flags>`	Allows you to specify additional behavior that you want to occur while Data Services searches the input_string for pattern matches. Enter NULL if you do not want to specify a flag. To specify multiple flags, delimit each flag using commas. Specify flag options in upper case.

Base the regular expression on the POSIX standard. For a list of valid regular expression pattern parameters, operators, and flags, see match_regex [page 1219].

Details

Example

Function	Results
`regex_replace( 'Accounting', '\\[a-h\\]','X', 'CASE_INSENSITIVE')`	'XXXountXnX'
`regex_replace('Accounting', '\\[a-h\\]', 'X')`	'AXXoutXnX'

7.4.124 replace_substr

Use the replace_substr function to replace each occurrence of a specified substring with a replacement string.

Syntax

```
replace_substr(<in_str>, <search_str>, <replace_str>)
```

Return Value

varchar

Returns the input string with the specified substring replaced with the replacement string.

Where

`<in_str>`	The input string that contains the substring to be changed. If `<in_str>` is NULL, the software returns NULL.
`<search_str>`	Substring to be replaced. If `<search_str>` is NULL, the software returns the string in `<in_str>`.

`<replace_str>`	Substring to use in place of `<search_str>`. If `<replace_str>` is blank or NULL, the software removes all occurrences of `<search_str>` from the `<in_str>`.

Details

Example

Function	Result
`replace_substr('a penny saved is a penny earned', 'penny', 'million')'`	Replaces the word "penny" with "million." `'a million saved is a million earned'`

7.4.125 replace_substr_ext

Use the replace_substr_ext function to replace each occurrence of a specified substring with a replacement string. The specified substring can contain hexadecimals that refer to a UNICODE character, or non printable character references such as form feed or new line.

Syntax

```
replace_substr_ext(<in_str>, <search_str>, <replace_str>,
<start_at_occurrence>, <number_of_occurrences>)
```

Return Value

varchar

Where

`in_str`	The input string that contains the substring to be changed. If `<in_str>` is NULL, the software returns NULL.

search_str	Substring to be replaced. If <search_str> is NULL, the software returns the string in <in_str>. 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 converts the next 4 characters to a hexadecimal value. This function converts the hexadecimal value to a UNICODE character. This option provides more flexibility when you use a search string. You can also represent special characters using the escape character '/'. The software supports the following characters: /a Bell (alert) /b Backspace /f Formfeed /n New line /r Carriage return /t Horizontal tab /v Vertical tab To include the escape character '/' in the search string, escape it using '//'. For example, if the input is 'abc/de', Data Services converts search_str to 'abcde'. If the input is 'abc//de', Data Services converts search_str to 'abc/de'. If search_str is NULL, Data Services returns a varchar with the data in in_str.
replace_str	Substring to use in place of <search_str>. If <replace_str> is blank or NULL, the software removes all occurrences of <search_str> from the <in_str>.
start_at_occurrence	Occurrence of the <search_str> with which to start replacing. If NULL, start at the 1st occurrence. For example, enter 2 to replace or remove the second occurrence of a search_str.
number_of_occurrences	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.

Details

Example

Function	Result
`replace_substr_ext('ayyyayyyayyyayyy', 'a', 'B', 2, 2)'`	Replaces 'a' with 'B' starting from the second occurrence and replaces two occurrences. 'ayyyByyyByyyayyy'
`replace_substr_ext('ayyya</n>yyya</n>yyyayyy', 'a/n', 'B', 2, 2)`	Searches a string containing 'a' followed by a new line; replaces it with 'B' starting from the second occurrence; replaces two occurrences. 'ayyyByyyByyyayyy'
`replace_substr_ext('ayyya</n>yyya</n>yyyayyy', 'a/x000a', 'B', 2, 2)`	Searches a string containing 'a' followed by a new line; replaces it with 'B' starting from the second occurrence; replaces two occurrences. 'ayyyByyyByyyayyy'

7.4.126 repository_name

Use the repository_name function to return a database connection string and owner name.

Syntax

```
repository_name()
```

Return Value

varchar

Details

For example, a return value of `beq-local.DBUser` is the ID for the repository from which the job is run.

Example

```
print('Repository Name: [repository_name()]')
```

7.4.127 restore_repserver_cdb_backlogged_transactions

Use the restore_repserver_cdb_backlogged_transactions function to restore backlogged transactions saved in the Replication Server real-time CDC (changed data capture) datastore.

Syntax

```
restore_repserver_cdb_backlogged_transactions (tablespec, dataflowname, jobname)
```

Return values

int

0 = Recovery request is pending.

1 = Recovery request has been recorded and is processed by the CDC job when it finishes processing the current changed dataset transactions.

Where

> **i Note**
>
> All parameters are mandatory and support global variables and substitution parameters.

<tablespec>	Consists of <datastore>.<owner>.<table>. Ensure that the datastore name is a valid Replication Server real-time CDC datastore. Values for <owner> and <table> can have a wild card character *. Otherwise, ensure that the owner and table that you specify in the parameter are imported to CDC datastore and executed to retrieve data from the queue. **Example** The following are example values for <tablespec>: ORACLE_CDC_DATASTORE.PUBLISHER.BOOKS ORACLE_CDC_DATASTORE.*.BOOKS ORACLE_CDC_DATASTORE.PUBLISHER.* ORACLE_CDC_DATASTORE.*.*
<dataflowname>	Valid data flow name or wild card character *.
<jobname>	Valid job name.

7.4.128 round

Use the round function to round a given number to a specified precision.

Syntax

```
round(<num1>, <precision>)
```

Return value

decimal, double, int, or real

The rounded number using the same data type as the original number, <num1>.

Where

<num1>	The source number.
<precision>	An integer indicating the number of decimals in the result. If <precision> is negative, the software rounds the digits left of the decimal point.

Details

Example

Function	Results
`round(120.12345, 2)`	120.12
`round(120.12999, 2)`	120.13
`round(120, -2)`	100
`round(120.123, 5)`	120.12300

7.4.129 rpad

Use the rpad function to pad a string of characters from a given pattern.

Syntax

```
rpad(<input_string>, <size>, '<pad_string>')
```

Return value

varchar

The new string.

Where

`<input_string>`	The source string.
`<size>`	An integer value indicating the number of characters in the resulting string.
`<pad_string>`	A character or set of characters that this function concatenates to `<input_string>`.

Details

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, the function truncates the string.

Example

Function	Results
`rpad('Tanaka',15,' ')`	`'Tanaka  '`
`rpad(last_name,25,' ')`	The value in the column `last_name`, padded with spaces to 25 characters, or truncated to 25 characters.

7.4.130 rpad_ext

Use the rpad_ext function to pad a string with logical characters from a given pattern.

Syntax

```
rpad_ext(<input_string>, <size>, '<pad_string>')
```

Return value

varchar

The new string.

Where

`<input_string>`	The source string.
`<size>`	An integer value indicating the number of characters in the resulting string.
`<pad_string>`	A logical character or set of logical characters that this function concatenates to `<input_string>`.

Details

> **Note**
>
> The logical characters prohibit this function from getting pushed down to an Oracle database.

The function repeats the pattern at the end of the input string until the final string is the appropriate length. If the input string is already longer than the expected length, this function truncates the string.

> **Example**

Function	Results
`rpad_ext('Tanaka',15,' ')`	`'Tanaka  '`
`rpad_ext(last_name,25,' ')`	The value in the column `last_name`, padded with spaces to 25 characters, or truncated to 25 characters.

The rpad_ext and rpad functions exhibit the same behavior when the software evaluates the functions. In situations where the function is pushed down to the database, the database behavior may differ when `<input_string>` and—or `<pad_string>` parameters contain multibyte characters.

Function	Input	Output
rpad	("abc¶¶", 10,' ')	'abc¶¶ '
rpad_ext	("abc¶¶", 10,' ')	'abc¶¶ '
rpad	("abc¶¶", 10,'')	'abc¶¶'
rpad_ext	("abc¶¶", 10,'')	'abc¶¶'
rpad	("abcd", 10, '¶')	'abcd¶¶¶'
rpad_ext	("abcd", 10, '¶')	'abcd¶¶¶¶¶¶'

7.4.131 rtrim

Use the rtrim function to remove specified characters from the end of a string.

> **Syntax**

```
rtrim('<input_string>', '<trim_string>')
```

Return value

varchar

The modified string. The return type is the same as `<input_string>`.

Where

`<input_string>`	The string to be modified.
`<trim_string>`	The characters to remove from `<input_string>`.

Details

The function scans `<input_string>` from right to left removing all characters that appear in `<trim_string>` until it reaches a character not in `<trim_string>`.

Removes trailing blanks only if `<trim_string>` contains trailing blanks. If the length of the modified string becomes zero after trimming, the function returns '' (empty string).

To remove all trailing blanks in a string, use the rtrim_blanks function.

Example

Function	Results
`rtrim('Marilyn ', ' ')`	'Marilyn'
`rtrim('ZABCABC', 'ABC')`	'Z'
`rtrim('ZABCABC', 'EFG')`	'ZABCABC'

You may also use the rtrim_blanks or rtrim_blanks_ext functions for this.

7.4.132 rtrim_blanks

Use the rtrim_blanks function to remove blank characters from the end of a string.

Syntax

```
rtrim_blanks(<input_string>)
```

Return value

varchar

The modified string. The return type is the same as `<input_string>`.

Where

`<input_string>`	The string to be modified.

Details

If the length of the modified string becomes zero after trimming, the function returns '' (empty string).

Example

Function	Results
`rtrim_blanks('Marilyn ')`	'Marilyn'
`rtrim_blanks(last_name)`	The value contained in the column `last_name` with trailing blanks removed.

7.4.133 rtrim_blanks_ext

Use the rtrim_blanks_ext function to remove blank and control characters from the end of a string.

Syntax

```
rtrim_blanks_ext(<input_string>)
```

Return value

varchar

The modified string. The return type is the same as `<input_string>`.

Where

`<input_string>`	The string to be modified.

Details

If the length of the modified string becomes zero after trimming, the function returns '' (empty string).

Example

Function	Results
`rtrim_blanks('Marilyn ')`	'Marilyn'
`rtrim_blanks(last_name)`	The value contained in the column `last_name` with trailing blanks and control characters removed.

7.4.134 sap_openhub_processchain_execute

Use the `sap_openhub_processchain_execute` function to start a process chain that extracts data from an InfoProvider on SAP Business Warehouse (BW) and loads the extracted data into an Open Hub Destination table. Monitors the process chain status and Open Hub Destination request notification.

Syntax

```
sap_openhub_processchain_execute('<datastore>','<open_hub_table>',
'<process_chain>',<$logid>,<$ReturnTxt>)
```

Return value

varchar(1)

Returns one of the values described in the following table.

Return values

Return value	Description
B	Open Hub Destination is being read by another user.
E	Data Services error while executing the function.

Return value	Description
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.

Where

`<datastore>`	Specifies the datastore name. Specify either a constant string or a substitution parameter. The data type is varchar(256). **i Note** The maximum length depends on the Data Services repository type. For most repository types the maximum length is 256, for MySQL the length is 64, and for MS SQL server the length is 128.
`<open_hub_table>`	Specifies the Open Hub Destination table. Specify either a constant string or a substitution parameter. The data type is varchar(30).
`<process_chain>`	Specifies the name of the process chain that extracts data from the InfoProvider in SAP BW and loads the data to the Open Hub Destination table. Specify either a constant string or a substitution parameter. The data type is varchar(30).
`<$logid >`	Optional. Specifies a variable to obtain a value that depends on the function return value. The required variable data type is varchar(25).
`<$ReturnTxt>`	Optional. Specifies a variable to retrieve the description of the return status of the process chain. The required variable data type is varchar. You can define the length for this variable.

Details

This function performs the following tasks:

- Starts the process chain that extracts data from an InfoProvider such as InfoArea, InfoCube, or DataStore object on SAP BW.
- Loads the extracted data into an Open Hub Destination table.
- Monitors the process chain status.
- Monitors the Open Hub Destination request notification.

When the function returns successfully, an Open Hub table source in SAP Data Services reads the data from the Open Hub Destination table.

> **Note**
>
> Use this function only in a script. It is not valid in a query or audit object.

> **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);
> ```

Limitations

The following are limitations 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.

Related Information

Defining an sap_openhub_processchain_execute function

7.4.134.1 Relationship between return value and logid variable value

The logid output variable depends on the sap_openhub_processchain_execute function return value.

The following table shows how the function return value relates to the other values in the function, and action to take.

Return value	$logid variable value	$ReturnText variable value	Action
B	Process chain log ID of another user who is currently reading the Open Hub Destination.	Status of current Open Hub extraction.	Either wait and try executing the data flow 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).

7.4.135 sap_openhub_set_read_status

Use the `sap_openhub_set_read_status` function to send the read status for the Open Hub table to SAP Business Warehouse (BW).

Syntax

```
sap_openhub_set_read_status('<datastore>','<destination>',
<status>,<$returntxt>)
```

Return value

varchar(1)

A successful read status causes SAP BW to delete the data from the Open Hub Destination table. Returns one of the following values.

Return value	Description
S	Success
E	Error

Where

<datastore>	Specifies the datastore name. Specify either a constant string or a substitution parameter. The data type is varchar(256). **i Note** The maximum length depends on the Data Services repository type. For most repository types the maximum length is 256, for MySQL the length is 64, and for MS SQL server the length is 128.
<destination>	Specifies the Open Hub Destination in the BW system. Specify either a constant string or a substitution parameter. The data type is varchar(30).
<status>	Specifies the read status. Possible values are either a variable or one of the following string constants: • X for Read Successful • Any other value indicates that the Read failed. The data type is varchar(1).
<$returntxt>	Optional. Specifies a variable to return the status log of the function. The required variable data type is varchar. Define the length you want for this variable.

Details

Example

The following script 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);
```

7.4.136 search_replace

Use the search_replace function to perform a simple search and replace based on a string or ford value, or an entire field.

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 (...)
```

Return value

varchar

Where

`<sr_table_spec>`	A constant string that specifies the search and replace table or file. Use one of the valid forms: • `<Datastore.owner.table>`: Specifies a database table containing the search and replace values. • `<Datastore.owner.schema.table>`: Specifies a database table containing the search and replace values for Netezza 7.x multischema only. • `<Fileformat.filename>`: Specifies a fixed or delimited file containing the search and replace values. • **NULL**: Use when the software performs the search and replace using custom SQL or an internal table that you define in the SET options.
`<search_column>`	Specifies the column name in the table or file containing the search values. If `<sr_table_spec>` is an internal table, set `<search_column>` to **NULL**. Varchar.
`<replace_column>`	Specifies the column name in the `<sr_table_spec>` table or file containing the replacement values. If `<sr_table_spec>` is an internal table, set `<replace_column>` to **NULL**. Varchar.

<sr_type>	Specifies the type of search and replace operation to perform. Use one of the following values: • **'SR_FIELD'**: Matches the entire contents of the search field and replaces the entire contents of the search field. • **'SR_WORD'**: Replaces only the word that matches the search value. Any unmatched data remains in the search field. A word is a set of characters set apart by whitespace. • **'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.
<case_sensitivity>	A value that indicates whether or not the search and replace operation is case sensitive. **yes**: Case sensitive **no**: Not case sensitive
<default_replace_value>	A varchar that specifies the default replacement value if the software does not fine the search value. The <default_replace_value> setting applies only when you set <sr_type> to **'SR_FIELD'**.
<input_column_list>	A varchar that specifies a comma-separated list of input expressions on which the search and replace operation is performed.
<output_column_list>	A varchar that specifies a comma-separated list of output columns.
<output_variable_list>	A varchar that specifies a comma-separated list of output variables. When you specify more than one output column in <output_column_list>, the software uses the output variables to receive output returns. The software matches variables and output columns by position. This parameter is optional except when you specify more than one output column in <output_column_list>.

SET options

Specifies custom SQL or search and replace values in XML format.

Example

For an internal search and replace table:

```
SET (
"internal_table"=
'<?xml version="1.0" encoding="UTF-8"?>
   <searchTable>
      <item>
         <Search><value></Search>
         <Replace><value></Replace>
      </item>
      ...
      ...
   </searchTable>
')
```

Example

For custom SQL:

```
SET (
"external_custom_sql"=
'<?xml version="1.0" encoding="UTF-8"?>
   <database_datastore>
      <datastore_name>
   </database_datastore>
   <search_column>
      <search_column_name>
   </search_column>
   <replace_column>
      <replace_column_name>
   </replace_column>
   <SQLText>
      <custom_sql_string>
   </SQLText>
')
```

Details

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 software loads the search and replace values into memory to optimize performance while performing the operation.

Note

We recommend that you use the search_replace function 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 parameters. You can also return to the wizard at any time to change the parameters. using search_replace as a function call also allows you to create multiple output columns when you use multiple input expressions. Using the function in a script or regular mapping is possible, but the syntax can be hard to read and difficult to maintain.

> **Note**
>
> The search_replace function wizard makes it easy to select search and replace columns, and if needed, define search terms and replacement values.

> **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>
)
```

Related Information

Defining a search_replace function

7.4.137 set_cdc_checkpoint

Use the set_cdc_checkpoint function to set a check point in a data flow for a Microsoft SQL Server changed data capture (CDC) job.

> **Syntax**
>
> ```
> set_cdc_checkpoint(<datastore>)
> ```

Return value

int

Returns 1 if successful, otherwise 0.

Where

`<datastore>`	The name of the CDC-enabled datastore containing the tables used to obtain the changes.

Details

Use the set_cdc_checkpoint function for data flows that run in a WHILE loop. The function retrieves changed data for each iteration of the loop. Call this function for all the datastores used in all the data flows of the job.

Example

```
set_cdc_checkpoint('MyCdcSource');
```

7.4.138 set_env

Use the set_env function to set a system environment variable to a specific value for the duration of a job.

Syntax

```
set_env('<variable_name>', <variable_value>)
```

Return value

int

Returns 1 if successful, otherwise, 0.

Where

`<variable_name>`	The name of an environment variable. Surround the name with single quotes.
`<variable_value>`	The value to assign to the environment variable. If the value is text, surround the text by single quotes. Data Services only sets the variable to this value for the duration of the current job.

Details

Use the get_env and set_env functions to set and retrieve variables across operations in a job.

Example

Function	Results
`set_env('TMP','C:\Data Services\Temp')`	Sets the value of the TMP environment variable to "C:\Data Services\Temp" and returns a 1.

7.4.139 sleep

Use the sleep function to suspend the execution of the calling data flow or work flow.

Syntax

```
sleep(<num_millisecs>)
```

Return Value

int

Always returns 1.

Where

`<num_millisecs>`	The number of milliseconds to "sleep".

Details

Calling this function causes the thread that executes this function to halt operations for the given number of milliseconds. To force a job to halt operations until a condition becomes true, call this function in a work flow, not in a data flow.

Example

The following example invokes sleep for one second when a file exists in a directory called 'c'.

```
while (file_exists('c:/temp.msg') == 0)
begin
sleep(1000);
end
```

7.4.140 soundex

Use the soundex function to encode the input string using the soundex algorithm and return a string.

Syntax

```
soundex(<input_str>)
```

Return Value

varchar(4)

Returns a string containing the soundex encoding of the input string. The return string length is always four characters.

Where

`<input_str>`	The source string to be encoded.

Details

Function results may vary when you push down to different database types.

Only use this function for input strings in English. Non-English characters are ignored.

The software ignores any invalid leading characters in the input string.

If an input string cannot be encoded, then the software returns '0000'.

Example

Function	Result
`Print(soundex('Hello');`	Prints the soundex of the word “Hello.”
`$VAR=soundex(emp_name);`	Returns the soundex encoding for the string stored in the variable emp_name and then assigns it to $VAR
`$VAR=soundex('1234567');`	Returns '0000' because the input data is numeric.

7.4.141 sql

Use the sql function to run an SQL operation against tables in the specified database.

Syntax

```
sql(<datastore, sql_command>)
```

Return value

varchar(1020)

Keep in mind the following results if you assign the value returned to a variable:

- 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 NULL.

Where

`<datastore>`	A string containing the name of the datastore where the tables involved in the SQL operation reside. This name is the name you specified when you created the datastore in Data Services. Include this string in single quotation marks.

<sql_command>	A string containing the text of the SQL command to execute. Enclose the string in single quotation marks. If the string contains quoted values, make the internal quotation marks single quotation marks preceded by the backslash (\) escape character. Data Services makes column and table names uppercase when sending the <sql_command> to Oracle to resolve. To specify a lowercase column or table name from an Oracle database, enclose the name with double quotation marks (").

Details

Example

Function	Results
`sql('source_ds', 'SELECT EmpID FROM Emp WHERE sal > 100000')`	Runs the SQL command against the database connected through the datastore source_ds.
`sql('source_ds','SELECT customer.LastName FROM customer WHERE customer.State = \'CA\'')`	Returns the last names of the customers in California from the customer table. Note that the quotation marks around the state value require a backslash to indicate that the single quotation marks are considered part of the <sql_command> string.
`sql('oracle_ds', ('SELECT "start_timestamp" FROM "status_table" WHERE "extract_name" = \'DF_RecoverDim\' AND "stop_timestamp" = NULL')`	Returns the timestamp from a status table for the completed data flow. An Oracle datastore requires double quotation marks around the lowercase column and table names.
`$start_date=sql('warehouse_ds', 'SELECT finish_timestamp FROM time_table WHERE table_name= "Component_Orders" ');`	In this script example, because the function returns a varchar value, it is not possible to assign the return value to a date variable directly. Modify your statement to the next example.
`$temp_char=sql('warehouse_ds', 'SELECT finish_timestamp FROM time_table WHERE table_name= "Component_Orders" ');` `$start_date=to_date($temp_char,'dd-mon-yyyy');`	This script example assumes the database returns the date in dd-mon-yyyy format. If you are unsure of the format the database returns, then force it to return the date in a specific format by doing a conversion. To accomplish this, use the to_char function in Oracle or the convert function in MS SQL.

7.4.142 sqrt

Use the sqrt function to return the square root of the given expression.

Syntax

```
sqrt(<num>)
```

Return Value

Float

Where

<num>	The number for which you want the square root.

Details

Return value is NULL if the input is negative.

Example

Function	Results
sqrt(625.25);	25.005000

7.4.143 smtp_to

Use the smtp_to function to capture the specified number of lines in the trace and error logs, package the information into an e-mail, and send it to the recipients using an SMTP server.

Syntax

```
smtp_to(<recipients_list>, <subject>, <message>, <number_of_trace_lines>,
<number_of_error_lines>)
```

Return value

int

Returns 0 if function succeeds. Returns a non-zero integer if function fails.

Where

<recipients_list>	Required. A string containing one or more recipient e-mail addresses separated by commas (,). Ensure that you enter valid, qualified e-mail address information.
<subject>	Optional. A string containing the subject of the e-mail.
<message>	Optional. A string containing the e-mail message.
<number_of_trace> <_lines>	Required. The number of lines from the end of the trace log file to append to the end of the e-mail.
<number_of_error> <_lines>	Required. The number of lines from the end of the error log file to append at the end of the e-mail.

Details

Use the smtp_to function in a script. For example, use the function in a conditional clause, while loop, or try-catch block.

> **i Note**
>
> The smtp_to function does not support nicknames.

> **Example**
>
> The following example goes to one recipient, admin@company.com.
>
> ```
> $myvar = smtp_to('admin@company.com', 'Out of memory error in the SalesFact job. Please fix the error before running recovery job.',' ',10, 10)
> ```

> **Example**
>
> The following message goes to two recipients, admin@company.com and manager@company.com. The job name and data flow names are included in the text of the message as variables. Note that Data Services trims blank spaces from the end of strings; this example includes a blank on the beginning of the next string. You can also concatenate a string with a single blank.

In the script, type `$a = ;` where `$a` is the local integer variable defined in the work flow. Put the cursor just ahead of the semicolon before clicking the *Functions* button to construct a smtp_to statement.

```
$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)
```

Example

The smtp_to function also supports global variables. This example from a script shows the smtp_to function substituting the values from global variables.

```
$address_list = 'nkumar@businessobjects.com,pkulkarn@businessobjects.com';
$subject_text = 'Testing SMTP job smtp011';
$message_text = 'The job smtp011 has the following trace lines & errors';
$trace_log = 8;
$error_log = 19;
$smtp_output = 9;
print ('before execution :- ' || $smtp_output);
$smtp_output = smtp_to($address_list,$subject_text,$message_text,
$trace_log,$error_log);
print ('after execution :- ' || $smtp_output);
```

Example

Use smtp_to in any user-defined custom function and invoke it in a script.

Create a custom function my_smtp that contains the following:

```
begin return(smtp_to('nkumar@ness-gsg.com','Testing job smtp012','Test
message of smtp012 job',4,5)); End
```

Use the custom function my_smtp in a script as follows:

```
print('before smtp :- '); my_smtp( ); print('after smtp :- ');
```

Related Information

Defining and enabling the smtp_to function

7.4.144 string_to_number

Use the string_to_number function to return the integer sum of all characters from the input string.

Syntax

```
string_to_number(<input string>)
```

Return Value

Int

Details

Returns the integer sum of all characters from the input string. Input string size is limited to 4000 characters.

Example

Function	Results
`string_to_number('SAP')`	228

7.4.145 substr

Use the substr function to return a specific portion of a string starting at a given point in the string.

Syntax

```
substr(<input_string>, <start>, <length>)
```

Return value

varchar

The modified string. The return data type is the `<input_string>`. If the length is a constant, then it is a varchar of the given length.

Where

`<input_string>`	The string to be modified.

<start>	The position in the <input_string> where the function obtains the first character of the new string. The function counts characters from the beginning of <input_string>. • In normal data flows, the first character is position number 1. • If <start> is 0, the new string begins with the first character (position 1). • If <start> is negative, the function counts characters from the end of <input_string>. The new string begins with the character in that position from the end of the string. The function returns NULL or an empty string under the following circumstances: • If <start> is greater than the number of characters in <input_string>, the function returns NULL. • If <length> is less than 1, the function returns an empty string.
<length>	The number of characters in the resulting string. • If <length> is 0 or negative, the function returns an empty string. • If <length> is greater than the number of characters remaining in <input_string> after <start> , the function returns only the remaining characters. The function keeps the trailing blanks in the remaining <input_string> after <start>.

For information about how Data Services uses the substr function with HANA, see SAP Note 2808903.

Details

Example

Function	Results
substr('94025-3373', 1, 5)	'94025'
substr('94025-3373', 7, 4)	'3373'
substr('94025', 7, 4)	NULL
substr('Dr. Schultz', 4, 18)	'Schultz'
substr('San Francisco, CA',-4, 18)	', CA'

7.4.146 sum

Use the sum function to calculate the sum of a given set of values.

Syntax

```
sum(<value_list>)
```

Return value

decimal, double, int, or real

The total of the values. The return type is the same as the values in `<value_list>`.

Where

`<value_list>`	The source values to sum.

Details

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.

7.4.147 sysdate

Use the sysdate function to return the current date as listed by the system.

Syntax

```
sysdate()
```

Return value

date

Today's date.

Details

Returns the current date as listed by the Job Server 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. It ignores the time data. If you change the data type to `datetime`, Data Services uses both a date and a time.

> **Example**

Function	Results
`isweekend(sysdate())`	Tests whether today is a Saturday or Sunday.
`to_char(sysdate(), 'yyyy.mm.dd')`	Converts the sysdate function `datetime` value to a string that displays only the date. Use this, for example, to exclude part of the `datetime` data by providing only a format for the data to display in a report. To convert a `datetime` value to a string containing only the date, use this expression and change the column data type to `varchar`.

7.4.148 system_user_name

Use the system_user_name function to return the name of the user who logged onto the Job Server operating system.

> **Syntax**

```
system_user_name()
```

Return Value

varchar

Details

Example

```
print('Starting execution of Job: [job_name()] as user:
[system_user_name()]');
```

7.4.149 systime

Use the systime function to return the current time as listed by the system.

Syntax

```
systime()
```

Return value

time

The current time.

Details

Returns the current time as listed by the Job Server's operating system.

Example

Function	Results
`$timestamp = sql('my_datastore', ('UPDATE status_table SET job_start_time = \' '|| to_char(systime(),'hh24:mi:ss.ff'))|| '\'');`	Updates the `job_start_time` column of the `status_table` with the current time. It also formats the time data.
`to_char(systime(),'hh24:mi:ss.ff')`	Trims date data from the systime() function in cases where it is added by default. Set the column that contains this expression to the data type `varchar`. The data type for a column that calls the systime() function should be `time`. If the data type is set to `datetime`, Data Services will add the default date for the `datetime` data type (1900:01:01) because systime() does not read dates.

7.4.150 sysutcdate

Returns the current UTC date as listed by the operating system of the server where the Agent is installed.

Note

The value that the sysutcdate function returns is a UTC datetime value. Internally Data Services reads both the date and the time when it runs a sysutcdate function. The data that is used by the task 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

```
sysutcdate()
```

Return value

date

Today's date.

Example

Function	Results
`isweekend(sysutcdate())`	Tests whether today (UTC) is a Saturday or Sunday.
`to_char(sysutcdate(), 'yyyy.mm.dd')`	Converts the sysutcdate function's `datetime` value to a string that displays only the date (UTC). For example, you can use this to exclude part of the `datetime` data by only providing a format for the data you want to display in a report. To convert a `datetime` value to a string containing only the date, use this expression and change the column's data type to `varchar`.

7.4.151 table_attribute

Use the table_attribute function to retrieve 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, the software returns NULL.

where

`<table_name>`	Enter the table name using the following format: `datastore.owner.table`. For Netezza 7.x multi-schema, use the following format: `datastore.owner.schema.table`. If a valid table by this name does not exist, the software returns NULL. This parameter is case sensitive.
`<attribute_name>`	The name of a table attribute. Valid attributes for a table are listed in the applicable table Properties. This parameter is case sensitive.

Details

Example

Function	Result
`table_attribute('mssql.awez.CUSTOMER', 'Number_Of_Inserts')`	'1788'

7.4.152 to_char

Use the to_char function to convert a date or numeric data type to a string.

Syntax

```
to_char(<date or numeric_expression>,'<format>')
```

Return

varchar

A formatted string that describes the specified `<numeric_expression>`.

Where

`<numeric expression>`	The source int, real, double or decimal data value.
`<format>`	A string indicating the format of the generated string. **Note** Provide format to ensure correct results. Choose from the codes listed in the following Format codes table.

Format codes

Format	Description	Example
9	Number Suppresses leading and trailing zeros from the results Includes a leading minus sign (-) for negative numbers or one leading space for pos numbers.	to_char(123,'9999') = ' 123'
0	Number Includes leading and trailing zeros.	to_char(123,'09999') = ' 0123' to_char(123,'9999D.00') = ' 123.00'
D<.|,>	Sets position of decimal point followed by character to use as decimal separator. Currently the software supports only dot (.) and comma (,) as decimal separators.	to_char(12.34,'99D.99') = ' 12.34'
G<.|,|space >	Position of group separator followed by character to be used as group separator. Currently the software supports only dot (.), comma (,) and space (' ') as group separators.	to_char(1234,' 9G,999') = ' 1,234'
x	Lower case "x." String containing unsigned hexadecimal integer, using "abcdef". If the number is not 2 bytes long, the software does not pad the output.	to_char(123,'xx') = ' 7b' to_char(12,'x') = ' c'
X	Upper case "X." String containing unsigned hexadecimal integer, using "ABCDEF". If the number is not 2 bytes long, the software does not pad the output.	to_char(123,'XX') = ' 7B' to_char(12,'X') = ' C'
0	String containing unsigned octal integer. This option is not case sensitive. If the number is not 2 bytes long, the software does not pad the output.	to_char(12,'oo') = ' 14' to_char(1,'o') = ' 1'

Where

> **Note**
>
> The to_char function supports the Oracle 9i `timestamp` data type up to 9 digits precision for sub-seconds.

`<date>`	The source date, time, or datetime value.
`<format>`	A string indicating the format of the generated string. Choose from the following codes: DD: 2-digit day of the month MM: 2-digit month MONTH: Full name of month MON: 3-character name of month YY: 2-digit year YYYY: 4-digit year HH24: 2-digit hour of the day (00-23) MI: 2-digit minute (00-59) SS: 2-digit second (00-59) FF: Up to 9-digit sub-seconds

Other values included in `<format>` appear unchanged in the result.

Details

> **Example**
>
> | Function | Results |
> |---|---|
> | `to_char(call_date, 'dd-mon-yy hh24:mi:ss.ff')` | The date value from the `call_date` column formatted as a string. Result:
`28-FEB-97 13:45:23.32` |
>
> The software reproduces the hyphens and spaces in the `<format>` parameter. The software recognizes all the other characters as part of a parameter string from the Date string table and substitutes with appropriate current values.

7.4.153 to_date

Use the to_date function to convert an input string to a date type based on the input format.

Syntax

```
to_date(<input_string>,'<format>')
```

Return value

date, time, or datetime

A date, time, or both representing the original string.

Where

<input_string>	The source string.
<format>	A string indicating the format of the source string. Choose from the following codes: DD: 2-digit day of the month MM: 2-digit month MONTH: Full name of month MON: 3-character name of month YY: 2-digit year YYYY: 4-digit year HH24: 2-digit hour of the day (00-23) MI: 2-digit minute (00-59) SS: 2-digit second (00-59) FF: Up to 9-digit sub-seconds **Note** Ensure that you set a format. If you do not set a format, the results may be incorrect.

Details

If the input string has more characters than the format string, the software ignores the extra characters in the input string and initializes to the default value.

Example

The software converts the following expression but ignores and initializes the extra characters to zero in the time part of the input string:

`to_date('10.02.2007 13:25:45', 'DD.MM.YYYY')` converts to `10.02.2007 00.00.00`

This function also supports the Oracle 9i `timestamp` data type. Its precision allows up to 9 digits for sub-seconds.

Example

Function	Results
`to_date('Jan 8, 1968', 'mon dd, yyyy')`	`1968.01.08` stored as a date.

7.4.154 to_decimal

Use the to_decimal function to convert a varchar to a decimal.

Syntax

```
to_decimal('<in_str>','<decimal_sep>','<thousand_sep>',<scale>)
```

Return Value

decimal

Uses a precision of 28 and the given scale.

Where

`<in_str>`	The number string. Null implies a NULL return. If `<in_str>` is invalid, the software returns a 0.
`<decimal_sep>`	The character that separates the decimal component from the whole number component.
`<thousand_sep>`	The character that separates thousands from hundreds in the whole number component.

`<scale>`	The number of digits to the right of the decimal point in the returned value.

Details

Example

Function	Result
`to_decimal('99,567.99', '.', ',', 3)`	`99567.990`

7.4.155 to_decimal_ext

Use the to_decimal_ext function to convert a varchar to a decimal and includes precision as a parameter.

Syntax

```
to_decimal_ext('<in_str>','<decimal_sep>','<thousand_sep>',<precision>,<scale>
)
```

Return Value

decimal

Uses the given precision and scale.

Where

`<in_str>`	The number string. Null implies a NULL return.
`<decimal_sep>`	The character that separates the decimal component from the whole number component.
`<thousand_sep>`	The character that separates thousands from hundreds in the whole number component.

`<precision>`	The total number of digits in the returned value.
`<scale>`	The number of digits to the right of the decimal point in the returned value.

Details

The to_decimal_ext function supports the use of DECIMAL data types with up to 96 precision.

Example

Function	Result
`to_decimal_ext('99,567.99', '.', ',', 38,3)`	`99567.990`

7.4.156 to_varchar

Use the `to_varchar` function to convert a date or numeric data type expression to a string.

Syntax

```
to_varchar(<date_expression or numeric_expression>,'<format>')
```

Return value

varchar

Where

`<date_expression>`	The source date, time, or date/time value. For example, 2010-01-11 00:00:00.
`<numeric_expression>`	The source int, real, double, or decimal value. For example, 10.30 or -0.001.

`<format>`	Indicates the format of the generated string. Include a format to ensure correct results. For applicable formats, see the **Date formats** table and the **Numeric formats** table.

Details

The `to_varchar` function generates a string that describes either the specified `<date_expression>` or the specified `<numeric_expression>` in the given numeric or data format specified in `<format>`.

For examples of the `to_varchar` function, see the **Date formats** table and the **Numeric formats** table.

Date formats for to_varchar function [page 1284]
Set an output format for the date expression in the `to_varchar` function.

Numeric formats for to_varchar function [page 1287]
Set an output format for the numeric expression given in the `to_varchar` function.

7.4.156.1 Date formats for to_varchar function

Set an output format for the date expression in the `to_varchar` function.

Use an applicable date format in the '`<format>`' portion of the `to_varchar` function to set the output format.

For simplicity, instead of including the entire date expression in the examples, we include the following representations of the source date expression:

- **D** = 2010-01-11 00:00:00
- **D1** = 2010-01-11 11:59:00
- **D2** = 2010-01-11 23:59:59

The following table contains applicable date format values with descriptions and examples.

Date formats

Format	Description	Example
YYYY	Outputs a four-digit year. Format is case insensitive.	`to_varchar(D, 'YYYY') = '2010'`
YY	Outputs a two-digit year. Format is case insensitive.	`to_varchar(D, 'YY') = '10'`
MM	Outputs a numeric month. Format is case insensitive.	`to_varchar(D, 'MM') = '01'`

Format	Description	Example

MONTH	Outputs the month name. Format is case-sensitive only for the first two characters as follows:
	• If the format starts with **MO**, the function outputs a month name in upper case.
	• If the format starts with **Mo**, the function outputs the month name with the first letter capitalized.
	• If the format starts with **mo** or **mO**, the function outputs the month name in lower case.
		`to_varchar(D, 'MONTH') = 'JANUARY'`
		`to_varchar(D, 'Month') = 'January'`
		`to_varchar(D, 'month') = 'january'`

MON	Outputs the three-character abbreviation for the month. Format is case-sensitive only for the first two characters as follows:
	• If the format starts with **MO**, the function outputs a month abbreviation in upper case.
	• If the format starts with **Mo**, the function outputs a month abbreviation with the first letter capitalized.
	• If the format starts with **mo** or **mO**, the function outputs a month abbreviation in lower case.
		`to_varchar(D, 'MON') = 'JAN'`
		`to_varchar(D, 'Mon') = 'Jan'`
		`to_varchar(D, 'mon') = 'jan'`

DDD	Outputs a three-digit value representing the day of the year. Format is case insensitive.	`to_varchar(D, 'DDD') = '011'`
DD	Outputs a two-digit value representing the day of the month. Format is case insensitive.	`to_varchar(D, 'DD') = '11'`
D	Outputs the one-digit day value representing the day of the week, ranging from 1 through 7. Format is case insensitive.	`to_varchar(D, 'D') = '1'`

Format	Description	Example

DAY	Outputs the name of the day. Format is case sensitive for only the first two characters as follows:
	• If the format starts with **DA**, the function outputs the day in upper case.
	• If the format starts with **Da**, the function outputs the day with the first letter capitalized.
	• If the format starts with **da** or **dA**, the function outputs the day in lower case.
		`to_varchar(D, 'DAY') = 'MONDAY'`
		`to_varchar(D, 'Day') = 'Monday'`
		`to_varchar(D, 'day') = 'monday'`

AM, PM	Outputs the given hour with **AM** or **PM**.
	• If the given hour is before noon, outputs AM.
	• If the given hour is after noon, outputs PM.
	The function is case-sensitive, and outputs value using the input case.
		`to_varchar(D, 'PM') = 'AM'`
		`to_varchar(D1, 'AM') = 'AM'`
		`to_varchar(D2, 'am') = 'am'`
		`to_varchar(D, 'aM') = 'aM'`

HH12, HH	Outputs an hour between 1 and 12.	`to_varchar(D2, 'HH12') = '11'` `to_varchar(D2, 'HH') = '11'`
HH	Outputs an hour between 0 and 23.	`to_varchar(D2, 'HH24') = '23'`
MI	Outputs the minutes in the given hour from 0 through 59.	`to_varchar(D2, 'MI') = '59'`
SS	Outputs the seconds in the given minute from 0 through 59.	`to_varchar(D2, 'SS') = '59'`

Format	Description	Example
FF	Outputs the milliseconds in the given second, excluding the number of sub-seconds.	`to_varchar(D, 'FF') = '000000000'` **i Note** In the example we didn't set milli-seconds, therefore, the value is ze-ros.
Other characters	Outputs the replicated character in the specified position.	`to_varchar(D, 'dd/mm/yyyy') = '11/01/2010'`

Parent topic: to_varchar [page 1283]

Related Information

Numeric formats for to_varchar function [page 1287]

7.4.156.2 Numeric formats for to_varchar function

Set an output format for the numeric expression given in the `to_varchar` function.

Use an applicable numeric format code in the '`<format>`' portion of the `to_varchar` function as described in the following table.

Numeric formats

Format	Description	Example
9	Outputs a number in the specified posi-tion. If there's no number in the source for this position, the function outputs a blank. This format is repeatable.	`to_varchar(100, '9999.00') = '100.00'`
0	Outputs a number in the specified posi-tion. If there's no number in the source for this position, the function outputs a zero. This format is repeatable.	`to_varchar(100, '0000.00') = '0100.00'`

Format	Description	Example
S	Outputs a positive or negative symbol in the specified position. • Outputs a "+" for a positive number • Outputs a "-" for a negative number	`to_varchar(100, 'S0000.00') = '+0100.00'`
E	Outputs the source numeric expression divided into the significant part and the exponent part.	`to_varchar(-0.001, 'S0.0E00') = '-1.0E-03'`
%	Outputs the result of multiplying the source number by 100 and adds a percent symbol at the end of the output.	`to_varchar(10.30, 'S9.9%') = '+1030.%'`
.	Outputs the source numeric expression with a period inserted in the specified position.	`to_varchar(1, '00.00') = '10.00'`
Other characters	Outputs the source numeric expression with the character in the specified position.	`to_varchar(1000, '$9,999.99') = '$1,000.'`

Parent topic: to_varchar [page 1283]

Related Information

Date formats for to_varchar function [page 1284]

7.4.157 to_WKT_point

Use the to_WKT_point function to convert latitude and longitude to a geometry point in Well Known Text (WKT) format.

Syntax

```
to_WKT_point (x, y)
```

Return value

Long

Point data that represents the location on a map for the given latitude and longitude.

Where

`<x>`	Latitude
`<y>`	Longitude

The software issues an error if the input latitude and longitude fields are not in one of the following formats:

- Decimal
- Integer
- Float
- Double

Details

The software outputs point data using geometry data type. You can upload the data objects to your external spatial data, such as a shapefile. This function is limited to loading data to SAP HANA.

The return value represents a point on the map based on the decimal latitude and longitude from your input data. The software outputs the WKT point information as long data type. Long data type provides enough space so that the software does not truncate the WKT point information.

Example

Input:

x = 3

y = 6

Output:

POINT (3 6)

7.4.158 total_rows

Use the total_rows function to return the number of rows in a spcified table in a datastore.

Syntax

```
total_rows(<datastore.owner.table_name>)
```

For a Netezza 7.x multischema datastore:

```
total_rows(<datastore.owner.schema.table_name>)
```

For a memory datastore:

```
total_rows(<datastore..table_name>)
```

Return value

int

The number of rows in the table.

Where

`<datastore>`	The name of the datastore containing the table.
`<owner>`	The name of the datastore owner. Not applicable for memory tables.
`<schema>`	The name of the datastore schema. Applicable only for Netezza 7.x datastore.
`<table_name>`	The name of the database table or memory table containing the rows you want to count.

Details

Example

Function	Results
`total_rows (ora_ds.scott.emp_table)`	Retrieves the total number of rows from an Oracle table.
`total_rows (mem_ds..bigtable)`	Retrieves the total number of rows from a memory table.

7.4.159 translate

Use the translate function to translate selected characters of an input string into other specified characters.

Syntax

```
translate(<input string>, <from string>, <to string>)
```

Return Value

String

Returns the input string translated in the following way: The software replaces all occurrences of each character in the `<from string>` with the corresponding character in the `<to string>`.

Where

`<input string>`	The string to be translated.
`<from string>`	The characters to be replaced from the `<input string>`. • The software does not replace the characters in the `<input string>` that are not in the `<from string>`. • The software removes the characters in the `<from string>` that do not have a corresponding character in the `<to string>` from the `<input string>`.
`<to string>`	The corresponding characters to replace the characters in `<from string>`.

Details

If the `<from string>` or `<to string>` is null, then the software returns null. This function is case sensitive with parameter values.

Example

Function	Results
`translate('Business Objects','sne','xyz')`	'Buxiyzxx Objzctx'
`translate('Business Objects','st','x')`	'Buxinexx Objecx'
`translate('Business Objects','s','')`	'Buine Object'
`translate('Business Objects','abcd',NULL)`	NULL

7.4.160 trunc

Use the trunc function to truncate a given number to the specified precision without rounding the value.

Syntax

```
trunc(<num1>, <precision>)
```

Return value

decimal, double, int, or real

The truncated number. The return type is the same as the original number, `<num1>`.

Where

`<num1>`	The source number.
`<precision>`	An integer indicating the number of decimals in the result. If `<precision>` is negative, digits to the left of the decimal point are truncated and the value is padded with zeros.

Details

Example

Function	Results
`trunc(120.12345, 2)`	`120.12`
`trunc(120.12999, 2)`	`120.12`
`trunc(180, -2)`	`100`
`trunc(120.123, 5)`	`120.12300`

7.4.161 truncate_table

Use the truncate_table function to explicitly expunge data from a memory table or truncate physical files used for a persistent cache table.

Syntax

```
trunc(<ds..tab_name>)
```

Return value

int

The return value is always 1.

Where

`<ds>`	The datastore containing the memory table or persistent cache table.
`<tab_name>`	The name of the memory table from which you want to expunge data or the name of the persistent cache table from which you want to truncate physical files.

Details

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.

Example

Function	Results
`truncate_table (ds..bigtable)`	Truncates rows from the memory table or persistent_cache table.

Usage scenarios

The following scenarios show some situations in which you use the truncate_table function.

1. A data flow in your job creates a persistent cache table. You can use the persistent cache table in subsequent data flows within the job by using the lookup_ext function, for example.
   Add a custom function and include the truncate_table function. Set 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 truncates your physical files, which frees disk space.

7.4.162 upper

Use the upper function to change 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>`. The software does not change the characters that are not letters.

Where

`<value>`	The string to be modified.
`<locale>`	Optional. A locale to which the function converts the string. **i Note** The software supports ISO 639 language code and ISO 3166 country code formats.

Details

Example

Function	Results
`upper('Accounting101')`	`'ACCOUNTING101'`
`upper(substr(LastName,1,1))|lower(substr(LastName, 2,LENGTH(LastName)))`	The value in column `LastName` with the first letter uppercase and the rest of the value lowercase. Note that this example does not account for last names with two words.
`upper(LastName, 'tr')`	The value in column `LastName` is converted to all uppercase. It is also converted to the Turkish locale, using the ISO 639 language code.

7.4.163 utc_to_local

Use the utc_to_local function to convert an input that is in Coordinated Universal Time (UTC) to the set time zone value.

Syntax

```
utc_to_local(<input datetime>, <timezone to convert with UTC offset>)
```

Return Value

datetime

Details

Converts the input in UTC to the desired time zone value. The second parameter UTC offset is a constant value. If the UTC offset is not provided, then the software uses the time zone of the job server host to calculate the UTC offset.

Example

Function	Results
`utc_to_local('2014.01.31 15:30:00', 'UTC+08:30')`	'2014.02.01 00:00:00'

7.4.164 varchar_to_long

Use the varchar_to_long function to convert a data type value of a given column from varchar to long.

Syntax

```
varchar_to_long(<column_name>)
```

Return value

long

Where

`<column_name>`	The name of the table column for which you want to convert a data type from varchar to long.

Related Information

XML extraction and parsing for columns

7.4.165 wait_for_file

Use the wait_for_file function to look for a specified file pattern in a file system, polling for the file at intervals, until the job timeout is reached.

Syntax

wait_for_file (file_name_pattern, timeout, poll_interval,max_match, file_name_list, list_size, list_separator)

Return Values

int

Values are:

- 0 - No file matched.
- 1 - At least one file is matched.
- -1 - Timed out.
- -2 - At least one input value is illegal.

Where

`<file_name_pattern>`	The file name and path, relative to where the Job Server is running. It can be an absolute or relative path. File name can contain wildcard characters.
`<timeout>`	Wait timeout in milliseconds. • If timeout is 0, then the function doesn't block. • If timeout is -1, then the function waits indefinitely for at least one file to exist that matches the file pattern. If you enter any other negative value, the software considers it illegal. On a computer where millisecond timing accuracy isn't available, timeout is rounded up to the nearest legal value available on that system.
`<poll_interval>`	Polling interval in milliseconds to look for the existence of the file. On a computer where millisecond timing accuracy isn't 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 is rounded up to time out value.
`<max_match >`	Optional. Specifies the maximum number of matched file names that the function returns. The default value is 0. -1 specifies that the function return all the matched file names.

`< file_name_list >`	Optional. Output varchar variable that returns the list of matched file names. Order of the file names in the list is determined by the way the operating system returns the file names.
`< list_size>`	Optional. Output integer variable that returns the list size.
`<list_separator>`	Optional. File name list separator character(s). Default value is comma (,).

Details

This function looks for the specified file pattern in the file system. If it doesn't 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.

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

Use this function in a script at the beginning of a job. When you execute the job, the job suspends until it finds the specified file. A job stays suspended until the file is present, as shown in the following business use case.

During the night, an external process puts source files in a file system that the system can access. The process is usually complete at 1:00 AM or later. Tonight, however, you schedule the job to start at 1:00 AM. You include a script in the first step of the job that checks for the existence of the last file. If the last file doesn't exist, the job waits for an interval of time and tries again. Once the file is present, the job finds the file and continues with the rest of the process. You set a timeout so that the job stops if the file is still not found at 9:00 tomorrow morning.

7.4.166 week_in_month

Use the week_in_month function to determine the week number of the month in which the given date falls.

Syntax

```
week_in_month(<date1>)
```

Return value

int

The number from 1 to 5 that represents which week in the month that `<date1>` occurs.

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.

Where

`<date1>`	The source date.

Details

Example

The following examples use the to_date function to convert the input date to a date type.

Function	Results
`week_in_month(to_date('Jan 22, 1997', 'mon dd, yyyy'))`	4
`week_in_month(to_date('Jan 21, 1997', 'mon dd, yyyy'))`	3

7.4.167 week_in_year

Use the week_in_year function to return the week in the year in which the given date falls.

Syntax

```
week_in_year(<inputdate>,'<weektype>')
```

Return value

int

Returns from 1 to 53.

Where

`<inputdate>`	The source date
`<weektype>`	Optional. This function returns the week in the year in two ways based on your setting: • 'WW' - Absolute week number of the given date. • 'IW' - ISO week number of the given date.

Details

Consider the following information when you use this function:

- This function considers the first week of the year to be the first seven days when it determines 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 always has at least 4 days.
- If 1-Jan falls on a Friday, Saturday, or Sunday, the first few days of the year are defined as being in the last (52nd or 53rd) week of the previous year.

Example

Some business applications use week numbers to categorize dates. For example, a business may report sales amounts by week, and identify each period as "9912", representing the 12th week of 1999. An ISO week is more meaningful than an absolute week for such a purpose.)

Following are more example results for week_in_year applied to three different input dates:

Function	Results
`week_in_year(to_date('Jan 01, 2001','mon dd, yyyy'))`	1
`week_in_year(to_date('2005.01.01', 'yyyy.mm.dd'),'WW')`	1
`week_in_year(to_date('2005.01.01', 'yyyy.mm.dd'),'IW')`	53

7.4.168 WL_GetKeyValue

Use the WL_GetKeyValue function to return the value of a given keyword in Web log search strings.

Syntax

```
WL_GetKeyValue(<string>, <keyword>)
```

Details

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'.
```

7.4.169 word

Use the word function to return one word out of a given string.

Syntax

```
word(<input_string>, <word_num>)
```

Return value

varchar

A string containing the indicated word. The return type is the same as `<input_string>`.

Where

`<input_string>`	The source string.

<word_num>	A nonnegative integer specifying the index of the target word in the string. The first word in a string is word number 1. If <word_num> is 0 or greater than the number of words in <input_string>, then the word function returns a NULL string.

Details

A word is defined to be any string of consecutive non-white space characters terminated by white space, or the beginning and end of <input_string>. White space characters are the following:

- Space
- Horizontal or vertical tab
- Newline
- Linefeed

Example

Function	Results
word('Accounting Department', 1)	'Accounting'
word('Accounting', 1)	'Accounting'
word('Accounting', 2)	NULL

7.4.170 word_ext

Use the word_ext function to return a word that you identify by a position in a delimited string.

Syntax

```
word_ext('<string>', <word_num>, '<separator(s)>')
```

Return value

varchar

A string containing the indicated word. Return type is the same as <string>.

Where

`<string>`	The source string.
`<word_num>`	A nonnegative integer specifying the index of the target word in the string. The first word in a string is word number 1. If `<word_num>` is 0 or greater than the number of words in `<string>`, then the word function returns a NULL string.
`<separator(s)>`	Any character specified.

The function considers a word to be:

- Any string of consecutive characters, not including white spaces
- String of characters terminated by white space or the beginnning and end of `<string>`.

The function considers white space characters to be:

- Space
- Horizontal or vertical tab
- Newline
- Linefeed

Details

This function is useful for parsing Web log URLs or file names.

Example

Function	Results
`word_ext('www.sap.com',2,'.')`	`'sap'`
`word_ext('www.cs.wisc.edu', -2, '.')`	`'wisc'` A negative word number, -2 in the example, means the function counts from right to left.
`word_ext('www.cs.wisc.edu', 5, '.')`	`NULL`
`word_ext('aaa+=bbb+=ccc+zz=dd', 4, '+=')`	`'zz'` If 2 separators are specified (+=), the function looks for either one.
`word_ext(',,,,,aaa,,,,bb,,,c ', 2, ',')`	`'bb'` This function skips consecutive delimiters.

7.4.171 workflow_name

Use the workflow_name function to reurn 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 the function does not find a work flow, it returns the job name.

Example

```
print('Work Flow Name: [workflow_name()]')
```

7.4.172 year

Use the year function to determine the year in which the given date falls.

Syntax

```
year(<date1>)
```

Return value

int

The number that represents the year component of `<date1>`.

Where

`<date1>`	The source date.

Details

Example

Function	Results
`year(to_date('Jan 22, 1997','mon dd, yyyy'))`	1997
`year(to_date('03/97', 'mm/yy'))`	1997
`year(to_date('03/19', 'mm/yy'))`	2019

7.5 Data Services procedures

SAP Data Services supports stored procedures, which are executable objects or named entities that you store in a database.

A stored procedure is one or more precompiled SQL statements. By calling a stored procedure from within Data Services, you can call business logic that you have already coded. The ability to use your existing business logic enables you to quickly and conveniently develop data extraction and data management tasks. You can also use stored procedures for the following purposes:

- Maintain business logic rules and provide a single point of control to ensure rules are accurate and enforced
- Reduce network overhead with client/server applications. The software stores procedures on the database server and retains compiled execution plans for procedures in the data dictionary.

Data Services supports using stored procedures for the following databases:

- DB2
- ODBC
- Oracle
- Microsoft SQL Server
- SAP HANA
- SAP SQL Anywhere
- SAP ASE
- SAP Sybase IQ

- Teradata

Call stored procedures from the jobs you create and run in Data Services.

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, users who have execution permissions for procedures can call the stored procedure. After you import a stored procedure into Data Services, use stored procedures like functions in Data Services jobs.

Stored procedures include parameters. Each parameter has the following elements:

- Name
- Data type
- Mode such as `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.

For complete information about working with procedures, see the *Designer Guide*.

8 Data Services scripting language

Use the SAP Data Services scripting language to write scripts, custom functions, and expressions. also known as user-script functions. Further, you can use the scripting language to write expressions such as complex column mapping expressions and WHERE clause conditions.

Custom functions are also known as user-script functions. Further, use the scripting language to write expressions such as complex column mapping expressions and WHERE clause conditions.

The software uses the scripting language internally to save objects that you create into repository tables. Export saved objects to a file with the `.atl` extension. Later import the file to another repository. You export and import objects for such tasks as migrating to a new environment or backing up a repository before an upgrade.

8.1 SAP scripting language operators

Operators act like functions but the are symbols that specify the action the function takes.

The following table contains descriptions of the operators that you use in scripts and expressions. The table lists the operators in order of precedence.

> **i Note**
>
> When the software pushes operations to a DBMS, the DBMS determines the precedence based on DBMS rules.

Operator	Description
+	Addition
–	Subtraction
*	Multiplication
/	Division
=	Assignment, comparison
<	Comparison, less than
<=	Comparison, less than or equal to

Operator	Description
>	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
LIKE	Comparison, matches a specific character string with a specified pattern. Use the following wildcards with LIKE: • %: A string of zero or more characters. • _ (underscore): A single character. • []: A single character with a specific range or set. • [^]: A single character not within the specified range or set.
NOT LIKE	Comparison, excludes rows that match the LIKE criterion.

When you use the smart editor to add operators, type operators or use the built-in key pad. Access the key pad from the smart editor tool bar.

Use a comparison operator 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 valid comparisons); for example `NOT ($x IN (1,2,3))`

```
comparison OR comparison
comparison AND comparison
```

The following syntax is not valid:

```
$x NOT IN (1,2,3)
EXIST or NOT EXIST
```

Parent topic: Data Services scripting language [page 1307]

Related Information

SAP scripting language keywords [page 1309]

8.2 SAP scripting language keywords

Keywords are select words in the scripting language that you use in expressions based on syntax rules and desired behavior.

Keywords are listed in the selection list of the smart editor filtered by the context from which the smart editor is opened.

Parent topic: Data Services scripting language [page 1307]

Related Information

SAP scripting language operators [page 1307]

8.2.1 BEGIN

The keyword BEGIN indicates the beginning of the code that becomes the function, script, or other construct.

The software automatically adds BEGIN and END statements to function, transform, and script definitions.

Parent topic: SAP scripting language keywords [page 1309]

Related Information

CATCH [page 1310]
ELSE [page 1311]
END [page 1312]
IF [page 1312]
RETURN [page 1313]
TRY [page 1313]
WHILE [page 1314]

8.2.2 CATCH

The keyword 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, the software 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
```

Parent topic: SAP scripting language keywords [page 1309]

Related Information

BEGIN [page 1310]
ELSE [page 1311]
END [page 1312]
IF [page 1312]
RETURN [page 1313]
TRY [page 1313]
WHILE [page 1314]

8.2.3 ELSE

The keyword ELSE defines the second branch for an IF statement.

If there is no ELSE following an IF statement, the software takes no action if the condition is not met.

Parent topic: SAP scripting language keywords [page 1309]

Related Information

BEGIN [page 1310]
CATCH [page 1310]
END [page 1312]
IF [page 1312]
RETURN [page 1313]
TRY [page 1313]
WHILE [page 1314]

8.2.4 END

The keyword END indicates the end of the code that becomes the function, script, or other construct.

The software automatically adds BEGIN and END statements to function, transform, and script definitions.

Parent topic: SAP scripting language keywords [page 1309]

Related Information

BEGIN [page 1310]
CATCH [page 1310]
ELSE [page 1311]
IF [page 1312]
RETURN [page 1313]
TRY [page 1313]
WHILE [page 1314]

8.2.5 IF

The keyword IF indicates a conditional step in the code.

Construct an IF statement 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.

Parent topic: SAP scripting language keywords [page 1309]

Related Information

BEGIN [page 1310]
CATCH [page 1310]
ELSE [page 1311]

END [page 1312]
RETURN [page 1313]
TRY [page 1313]
WHILE [page 1314]

8.2.6 RETURN

The keyword 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.

Parent topic: SAP scripting language keywords [page 1309]

Related Information

BEGIN [page 1310]
CATCH [page 1310]
ELSE [page 1311]
END [page 1312]
IF [page 1312]
TRY [page 1313]
WHILE [page 1314]

8.2.7 TRY

The keyword TRY indicates the start of a try—catch block.

Parent topic: SAP scripting language keywords [page 1309]

Related Information

BEGIN [page 1310]
CATCH [page 1310]
ELSE [page 1311]
END [page 1312]

IF [page 1312]
RETURN [page 1313]
WHILE [page 1314]
CATCH [page 1310]

8.2.8 WHILE

The keyword 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.

Parent topic: SAP scripting language keywords [page 1309]

Related Information

BEGIN [page 1310]
CATCH [page 1310]
ELSE [page 1311]
END [page 1312]
IF [page 1312]
RETURN [page 1313]
TRY [page 1313]

9 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*.

9.1 Audit metadata tables

The audit tables in the Data Services repository store statistics about executed data flows.

The auditing metadata tables store statistics collected by the SAP Data Services audit feature.

9.1.1 AL_AUDIT

The AL_AUDIT metadata table contains information about each data flow execution.

The column OBJECT_KEY uniquely identifies a data flow execution. The following table describes each column in the AL_AUDIT metadata table.

AL_AUDIT contents

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: • 0 — Not audited • 1 — Audit rule succeeded • 2 — Audit information collected. This status occurs when you define audit labels to collect statistics but don't define audit rules. • 3 — Audit rule failed
RULEINFO	VARCHAR(255)	The audit rule that failed and the values of the left-hand-side (LHS) and right-hand-side (RHS) of the Boolean expression.

Example

The following query accesses the data in the AL_AUDIT table to create a list of data flows that failed an audit, including the failed rules.

```
SELECT L.NAME, RULEINFO
    FROM AL_AUDIT A, AL_LANG L
    WHERE A.STATUS = 3
        AND A.DF_LANG_KEY = L.OBJECT_KEY
```

Parent topic: Audit metadata tables [page 1315]

Related Information

AL_AUDIT_INFO [page 1316]

9.1.2 AL_AUDIT_INFO

The AL_AUDIT_INFO table contains information about audit statistics.

The following table describes the columns in the AL_AUDIT_INFO table.

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.

Column Name	Data Type	Description
VALUE	VARCHAR(255)	Value of the label. This value can be one of the following: • Number of rows in a table or whole row set • Sum of the values in a column • Average of the values in a column • Checksum of the values in a column

Example

The following query returns the values of the labels of audit points that failed.

```
SELECT L.NAME DF_NAME, B.LABEL, B.VALUE AUDIT_VALUE
    FROM AL_AUDIT A, AL_AUDIT_INFO B, AL_LANG L
    WHERE A.OBJECT_KEY = B.AUDIT_KEY
        AND A.STATUS = 3
        AND A.DF_LANG_KEY = L.OBJECT_KEY;
```

The following query returns values of audit labels for a specific data flow.

```
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';
```

Parent topic: Audit metadata tables [page 1315]

Related Information

AL_AUDIT [page 1315]

9.2 Imported object metadata tables and views

SAP Data Services collects information about imported metadata from external databases and applications, such as Oracle, PeopleSoft, and SAP Applications.

Data Services stores imported metadata in tables, prefixed with AL, and in views, prefixed with ALVW.

The AL_PCOLUMN table contains partition information for tables imported into the *SAP Data Services* repository.

AL_PKEY [page 1320]
The AL_PKEY table contains primary key information for tables that you imported into SAP Data Services.

ALVW_COLUMNATTR [page 1321]
The ALVW_COLUMNATTR view contains column attribute information

ALVW_COLUMNINFO [page 1322]
The ALVW_COLUMNINFO view contains information about the columns of tables that you imported to the SAP Data Services repository.

ALVW_FKREL [page 1323]
The ALVW_FKREL view contains information about the primary and foreign key relationships in tables.

ALVW_MAPPING [page 1324]
The ALVW_MAPPING view joins the AL_COLMAP and the AL_COLMAP_TEXT tables, which SAP Data Services uses for impact analysis in Metadata Reports.

ALVW_TABLEATTR [page 1333]
The ALVW_TABLEATTR view contains information about the attributes of imported tables in the SAP Data Services repository.

ALVW_TABLEINFO [page 1335]
The ALVW_TABLEINFO view contains a list of tables that you've imported into the SAP Data Services repository.

9.2.1 AL_INDEX

The AL_INDEX table contains table index information from table metadata that you imported into the SAP Data Services repository.

The following table describes the columns in the AL_INDEX table.

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'
```

Parent topic: Imported object metadata tables and views [page 1317]

Related Information

AL_PCOLUMN [page 1319]
AL_PKEY [page 1320]
ALVW_COLUMNATTR [page 1321]
ALVW_COLUMNINFO [page 1322]
ALVW_FKREL [page 1323]
ALVW_MAPPING [page 1324]
ALVW_TABLEATTR [page 1333]
ALVW_TABLEINFO [page 1335]

9.2.2 AL_PCOLUMN

The AL_PCOLUMN table contains partition information for tables imported into the *SAP Data Services* repository.

The following table describes the columns in the AL_PCOLUMN table.

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.

Parent topic: Imported object metadata tables and views [page 1317]

Related Information

AL_INDEX [page 1318]
AL_PKEY [page 1320]
ALVW_COLUMNATTR [page 1321]
ALVW_COLUMNINFO [page 1322]
ALVW_FKREL [page 1323]
ALVW_MAPPING [page 1324]
ALVW_TABLEATTR [page 1333]
ALVW_TABLEINFO [page 1335]

9.2.3 AL_PKEY

The AL_PKEY table contains primary key information for tables that you imported into SAP Data Services.

The following table contains descriptions for the columns in the AL_PKEY table.

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'
```

Parent topic: Imported object metadata tables and views [page 1317]

Related Information

AL_INDEX [page 1318]

9.2.4 ALVW_COLUMNATTR

The ALVW_COLUMNATTR view contains column attribute information

The following table describes the columns in the ALVW_COLUMNATTR view.

Column Name	Data type	Description
TABLE_NAME	Varchar(256)	Name of the table.
TABLE_OWNER	Varchar(256)	Owner of the table.
DATASTORE	Varchar(256)	The datastore to which this table belongs.
COLUMN_NAME	Varchar(256)	Name of the column.
COLUMN_ATTR	Varchar(64)	Name of the attribute (property of this table). The name of the attribute will be the same as seen when viewing the attributes of a table in the Designer's interface. Examples: Description, Business_Name, Date_last_loaded, Date_created, Total_Number_Of_Rows_Processed.
COLUMN_ATTR_VALUE	Varchar(255)	Value of this attribute.

Example

The following query returns all columns and their descriptions for table `EMP` in datastore `HR`:

```
SELECT COLUMN_NAME, COLUMN_ATTR_VALUE
    FROM ALVW_COLUMNATTR
    WHERE TABLE_NAME = 'EMP'
        AND COLUMN_ATTR = 'Description'
        AND DATASTORE = 'HR'
```

Parent topic: Imported object metadata tables and views [page 1317]

Related Information

AL_INDEX [page 1318]
AL_PCOLUMN [page 1319]
AL_PKEY [page 1320]
ALVW_COLUMNINFO [page 1322]
ALVW_FKREL [page 1323]
ALVW_MAPPING [page 1324]
ALVW_TABLEATTR [page 1333]
ALVW_TABLEINFO [page 1335]

9.2.5 ALVW_COLUMNINFO

The ALVW_COLUMNINFO view contains information about the columns of tables that you imported to the SAP Data Services repository.

The following table contains descriptions of the columns in the ALVW_COLUMNINFO view.

Column Name	Data type	Description
TABLE_NAME	Varchar(256)	Name of the table.
TABLE_OWNER	Varchar(256)	Owner of the table.
DATASTORE	Varchar(256)	The datastore to which this table belongs.
COLUMN_NAME	Varchar(256)	Name of the column.
COLUMN_DATATYPE	Varchar(20)	Data type of this column. Examples: `integer`, `datetime`, `decimal`, `real`.
COLUMN_LENGTH	Int	Length of this column in bytes.
COLUMN_PRECISION	Int	Precision of this column (applies only to columns with the data type `decimal`).
COLUMN_SCALE	Int	Scale for this column (applies only to columns with the data type `decimal`).

Column Name	Data type	Description
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'
```

Parent topic: Imported object metadata tables and views [page 1317]

Related Information

AL_INDEX [page 1318]
AL_PCOLUMN [page 1319]
AL_PKEY [page 1320]
ALVW_COLUMNATTR [page 1321]
ALVW_FKREL [page 1323]
ALVW_MAPPING [page 1324]
ALVW_TABLEATTR [page 1333]
ALVW_TABLEINFO [page 1335]

9.2.6 ALVW_FKREL

The ALVW_FKREL view contains information about the primary and foreign key relationships in tables.

The following table contains column descriptions for the ALVW_FKREL view.

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'
```

Parent topic: Imported object metadata tables and views [page 1317]

Related Information

AL_INDEX [page 1318]
AL_PCOLUMN [page 1319]
AL_PKEY [page 1320]
ALVW_COLUMNATTR [page 1321]
ALVW_COLUMNINFO [page 1322]
ALVW_MAPPING [page 1324]
ALVW_TABLEATTR [page 1333]
ALVW_TABLEINFO [page 1335]

9.2.7 ALVW_MAPPING

The ALVW_MAPPING view joins the AL_COLMAP and the AL_COLMAP_TEXT tables, which SAP Data Services uses for impact analysis in Metadata Reports.

The joined 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.

metadata

The column mapping calculation generates the following information for target columns:

- The source columns from which the target column is mapped.
- metadata 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.

> **i Note**
>
> For those column names with varchar(256), the maximum length depends on the Data Services repository type. For most repository types the maximum length is 256, for MySQL the length is 64, and for MS SQL server the length is 128.

ALVW_MAPPING view

Column Name	Data type	Description
DF_NAME	varchar(256)	Data flow that populates the target table.
TRG_TAB_NAME	varchar(256)	Name of the target table.
TRG_TAB_ID	int	ID for this table within the repository.
TRG_TAB_DESC	varchar(100)	Description of the target table.
TRG_OWNER	varchar(256)	Owner of the target table.
TRG_DS	varchar(256)	Datastore of the target table.
TRG_TYPE	varchar(64)	Type of target. Examples: table, BW transfer structure.
TRG_USAGE	varchar(65)	Usage of the target table. Examples: fact, dimension, lookup. Currently set to NULL.
TRG_COL_NAME	varchar(256)	Column name in the target.
TRG_COL_ID	int	ID for this column in the repository.
TRG_COL_DESC	varchar(100)	Description of this column.
SRC_TAB_NAME	varchar(256)	Name of the source table used to populate the target.
SRC_TAB_ID	int	ID for this table within the repository.
SRC_TAB_DESC	varchar(100)	Description of the source table.
SRC_OWNER	varchar(256)	Owner of the source table.
SRC_DS	varchar(256)	Datastore of the source table.
SRC_TYPE	varchar(64)	Type of source. Examples: table, file.
SRC_COL_NAME	varchar(256)	Name of the source column.
SRC_COL_ID	int	ID for this column in the repository.
SRC_COL_DESC	varchar(100)	Description of this column.
MAPPING_TYPE	varchar(65)	Types of source to target mapping. Examples: direct, computed, lookup.
MAPPING_TEXT	varchar(255)	The expression used to map the source to the target column.

Example

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'
```

AL_COLMAP_TEXT Mapping types [page 1327]
The AL_COLMAP_TEXT table contains information in the MAPPING_TYPE column that qualifies the mapping relationships in the ALVW_MAPPING view.

How SAP Data Services computes mapping [page 1328]
SAP Data Services performs complex transformations to prepare data for loading.

Mapping complexities [page 1329]
Transforms, such as Merge, Hierarchy_Flatteninng, and Pivot introduce mapping situations that increase the complexities for mapping.

Column mapping for nested data flows [page 1330]
SAP Data Services calculates column mappings for all data flows including data flows that use nested data.

AL_COLMAP_NAMES [page 1331]
SAP Data Services uses the AL_COLMAP_NAMES table to support nested data that is too large to store in the ALVW_MAPPING view.

Parent topic: Imported object metadata tables and views [page 1317]

Related Information

AL_INDEX [page 1318]
AL_PCOLUMN [page 1319]
AL_PKEY [page 1320]
ALVW_COLUMNATTR [page 1321]
ALVW_COLUMNINFO [page 1322]
ALVW_FKREL [page 1323]
ALVW_TABLEATTR [page 1333]
ALVW_TABLEINFO [page 1335]

9.2.7.1 AL_COLMAP_TEXT Mapping types

The AL_COLMAP_TEXT table contains information in the MAPPING_TYPE column that qualifies the mapping relationships in the ALVW_MAPPING view.

The information that qualifies the mapping relationships is stored in the MAPPING_TYPE column. The following table describes the values in the MAPPING_TYPE column.

MAPPING_TYPE	Description
Direct	The target column is mapped directly from a source column with no expression to transform it. For example, `EMPID` (employee ID) mapped directly from source to target.
Computed	There's an expression associated with the target column. For example, `NAME` is `LAST_NAME||','||FIRST_NAME`.
Generated	There's no source column associated with the target column. For example, the target table is mapped to a constant or a function, such as sysdate, or is obtained from a transform, such as Date_Generation.
LookedUp	A lookup function is used in the expression.
Merged	Two data streams are merged to populate the target table. The two expressions mapped to the target table are separated by AND.
Not mapped	The column in the target table isn't being populated by the data flow.
Unknown	Data Services is unable to identify the expression used to map the target column. The Unknown map type occurs only under unusual error conditions.

Parent topic: ALVW_MAPPING [page 1324]

Related Information

How SAP Data Services computes mapping [page 1328]
Mapping complexities [page 1329]
Column mapping for nested data flows [page 1330]

AL_COLMAP_NAMES [page 1331]

9.2.7.2 How SAP Data Services computes mapping

SAP Data Services performs complex transformations to prepare data for loading.

When you execute a data flow, Data Services prepares the data for loading into one or more target tables. During this preparation, Data Services performs complex transformations on data, which includes the following operations:

- Reads data from the appropriate sources.
- Processes data using Query transforms or other transforms.
- Splits the data stream and then merges it again.

Example

Two transformations operate against a value from one column of a source table. Data Services captures the information in the AL_COLMAP_TEXT table as follows:

Target column	Source column	Mapping expression
Total_value	Price	`((Price x 1.17) x 112)`

The kind of information that Data Services captures becomes more valuable as transformation complexity increases. Consider the following example:

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 output from **Q1** has four columns:
  - **Qa**
  - **Qb**
  - **Qc**
  - **Qd**
- The mapping expressions are:
  - **S.a**
  - **S.b**
  - **S.c**
  - **S.a – S.b**.
- The output of **Q1** is connected to Query transform **Q2**.
- **Q2** has two columns **Q2y** and **Q2z** with the exppressions:
  - **Qa – Qb**
  - **Qc – Qd**
- The output of **Q2** is loaded into target table **T**, which has two columns:

T1
T2

Data Services computes the mapping expressions for target columns **T1** and **T2** as follows:

- Starts from the end point (target), “walks” back through the list of transforms.
- Writes each column of each transform as expressions from the previous transform.

When Data Services starts processing **DF_1**, it starts with column **T1** of target table **T**. The expression for column **T1** is:

- **Q2y** , which in turn is **A – dB**
- **A – dB**, which is written as **S.a – S.b**
- Mapping expression is **S.a – S.b**

Column **T1** has two source columns, mapped from:

- **S.a**
- **S.b**

The AL_COLMAP_TEXT table contains two rows for the target column to describe the two source columns:

- **T2**
  - Mapped from **DC – JD**
  - Written as **S.c – (S.a – S.b)**

There are three rows for the target column in the AL_COLMAP_TEXT table, one for each source column.

Parent topic: ALVW_MAPPING [page 1324]

Related Information

AL_COLMAP_TEXT Mapping types [page 1327]
Mapping complexities [page 1329]
Column mapping for nested data flows [page 1330]
AL_COLMAP_NAMES [page 1331]

9.2.7.3 Mapping complexities

Transforms, such as Merge, Hierarchy_Flatteninng, and Pivot introduce mapping situations that increase the complexities for mapping.

If one data flow calls a second data flow, and loads the results to a target table, SAP Data Services expresses the mappings in terms of the tables and columns used in the second data flow. Data Services generates mapping information by "drilling down" into the second data flow to continue the mapping process.

If you introduce the Merge transform in a data flow, mapping is more complex. When two data streams are merged, Data Services has two ways to populate a target table. Separate the mapping expressions with the keyword `AND`, and a target column is populated from `S.a AND R.a`, for example.

Transforms like Hierarchy_Flattening and Pivot also introduce column-mapping complexities.

Further, some target columns are mapped by constants or expressions that don't use source columns. When mapping doesn't use source columns, there are no rows in the AL_COLMAP_TEXT table for the target column. consequently, the mapping expression in the AL_COLMAP_TEXT table doesn't include rows.

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.

Parent topic: ALVW_MAPPING [page 1324]

Related Information

AL_COLMAP_TEXT Mapping types [page 1327]
How SAP Data Services computes mapping [page 1328]
Column mapping for nested data flows [page 1330]
AL_COLMAP_NAMES [page 1331]

9.2.7.4 Column mapping for nested data flows

SAP Data Services calculates column mappings for all data flows including data flows that use nested data.

When Data Services calculates column mapping, it identifies the source columns and expressions in use and maps them to the target columns.

Data Services supports the following objects and conditions when it maps columns in nested data flows:

- XML files or messages
- IDOC files or messages
- Custom and adapter functions
- SAP Applications and PeopleSoft hierarchies
- Column mappings that perform nesting or unnesting (target columns mapped from a nested or unnested 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: If you map a column from an ancestor, the column is correlated:
  - Map a column in a nested schema from a column in the input schema of the nested schema
  - Map from a column in the input schema of the parent (or any ancestor) of the nested schema

Data Services supports nested column mapping for transforms 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 don't process nested data.

Nested (NRDM) notations that represent column names are longer than columns names used for a flat schema column:

- A column in a flat schema is represented by `<Table>. <Column>`, for example, `mats_emp.empno`.

  > **i Note**
  >
  > Table represents such objects as a database, file, XML message, XML file, IDOC message, or IDOC file.

- A column in a nested schema is represented by `<Table>.<Subschema1…SubschemaN>.<Column>`. For example `personnel.name.give` 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.

Parent topic: ALVW_MAPPING [page 1324]

Related Information

AL_COLMAP_TEXT Mapping types [page 1327]
How SAP Data Services computes mapping [page 1328]
Mapping complexities [page 1329]
AL_COLMAP_NAMES [page 1331]

9.2.7.5 AL_COLMAP_NAMES

SAP Data Services uses the AL_COLMAP_NAMES table to support nested data that is too large to store in the ALVW_MAPPING view.

The TRG_COL_NAME and SRC_COL_NAME columns in the ALVW_MAPPING view hold Varchar data up to 64 characters from MySQL. Consequently, they aren't large enough to store long NRDM column names. Therefore, Data Services uses the AL_COLMAP_NAMES table to support nested data.

The following table contains descriptions for the columns in the AL_COLMAP_NAMES table.

Column Name	Data type	Description
DF_NAME	varchar(256)	Data flow with that populates a target table.

Column Name	Data type	Description
COL_ID	varchar(65)	The software generates this value using the following format when a nested column is encountered. `"__DI_NESTED_COLNAME_n"` where n is an integer that starts from 1
COL_NAME	varchar(256)	If the software generates a COL_ID value, this column stores the original nested column name.
SEQNUM	int	The software generates this value if more than one set of 256 characters is required to store data in the COL_NAME column. For each set of 256 characters, it generates a new row and a sequence number.

The AL_COLMAP_NAMES table has the following primary key columns:

- DF_NAME
- COL_ID
- SEQNUM

The DF_NAME and COL_ID columns are keyed to the columns in the ALVW_MAPPINGS view follows:

AL_COLMAP_NAMES column	ALVW_MAPPINGS column
DF_Name	DF_Name
COL_ID	SRC_COL_NAME and TRG_COL_NAM

The AL_COLMAP_NAMES table also provides an internal mapping mechanism from COL_ID column to COL_NAME.

Example

If a source column `BOOKS.AUTHOR.FIRST_NAME` is mapped into a target column `BOOK.AUTHOR_NAME` (an unnesting is in place), you create a report to query the following column values in the repository:

ALVW_MAPPING view	Column value
TRG_TAB_NAME	BOOK
TRG_COL_NAME	AUTHOR_NAME
SRC_TAB_NAME	BOOKS
SRC_COL_NAME	__DI_NESTED_COLNAME_1
MAPPING_TEXT	BOOKS.AUTHOR.FIRST_NAME

AL_COLMAP_NAMES table	Column value
COL_ID	__DI_NESTED_COLNAME_1
COL_NAME	AUTHOR.FIRST_NAME

If the target or source column is nested, the TRG_COL_NAME or SRC_COL_NAME columns in the ALVW_MAPPING view store the COL_ID. To get the actual column name, lookup the AL_COLMAP_NAMES table using the DF_Name, COL_ID, and COL_NAME.

Flat or unnested target or source column names are stored using the format Column in TRG_COL_NAME and SRC_COL_NAME.

Example

Of the three source columns shown in the following table, only the second one is nested. The second value is the only one for which Data Services generates a column ID. To find the real name for this source column, create a report that looks up its COL_NAME from the AL_COLMAP_NAMES table.

SRC_COL_NAME
EMPNO
_DI_Nested_Names_1
ENAME

Parent topic: ALVW_MAPPING [page 1324]

Related Information

AL_COLMAP_TEXT Mapping types [page 1327]
How SAP Data Services computes mapping [page 1328]
Mapping complexities [page 1329]
Column mapping for nested data flows [page 1330]

9.2.8 ALVW_TABLEATTR

The ALVW_TABLEATTR view contains information about the attributes of imported tables in the SAP Data Services repository.

The following table contains descriptions of the columns in the ALVW_TABLEATTR view.

Column Name	Data type	Description
TABLE_NAME	Varchar(256)	Name of the table.
TABLE_OWNER	Varchar(256)	Owner of the table. For SAP applications, the value is `NULL`.

Column Name	Data type	Description
DATASTORE	Varchar(256)	The datastore to which this table belongs.
TABLE_ATTR	Varchar(64)	Name of the attribute (property of this table). The name of the attribute is the same the attribute of the table in the SAP Data Services Designer. 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'
```

Parent topic: Imported object metadata tables and views [page 1317]

Related Information

AL_INDEX [page 1318]
AL_PCOLUMN [page 1319]
AL_PKEY [page 1320]
ALVW_COLUMNATTR [page 1321]
ALVW_COLUMNINFO [page 1322]
ALVW_FKREL [page 1323]
ALVW_MAPPING [page 1324]
ALVW_TABLEINFO [page 1335]

9.2.9 ALVW_TABLEINFO

The ALVW_TABLEINFO view contains a list of tables that you've imported into the SAP Data Services repository.

The following table contains descriptions of the columns in the ALVW_TABLEINFO view.

Column Name	Data type	Description
TABLE_NAME	Varchar(256)	Name of the table.
TABLE_OWNER	Varchar(256)	Owner of the table. For SAP applications, this table is NULL.
DATASTORE	Varchar(256)	The datastore to which this table belongs.
TABLE_ID	int	Internal ID of the table within the repository.

Parent topic: Imported object metadata tables and views [page 1317]

Related Information

AL_INDEX [page 1318]
AL_PCOLUMN [page 1319]
AL_PKEY [page 1320]
ALVW_COLUMNATTR [page 1321]
ALVW_COLUMNINFO [page 1322]
ALVW_FKREL [page 1323]
ALVW_MAPPING [page 1324]
ALVW_TABLEATTR [page 1333]

9.3 Internal metadata tables and views

The internal metadata tables and views contain information about built-in metadata objects used by SAP Data Services.

Use the data in internal metadata tables and views to capture information about built-in metadata objects and their relationships.

AL_LANG [page 1336]
: The AL_LANG table contains SAP Data Services objects that are also listed in the object library.

AL_LANGXMLTEXT [page 1337]
: The AL_LANGXMLTEXT table contains SAP Data Services XML-related objects that are also displayed in the object library.

9.3.1 AL_LANG

The AL_LANG table contains SAP Data Services objects that are also listed in the object library.

The following table describes the columns in the AL_LANG table.

Column Name	Description
OBJECT_KEY	Internal ID of the object.
OBJECT_TYPE	Type of object (an integer). Query the AL_REPO-TYPE_NAMES table to find the corresponding name for this type. Examples: data flow, table, datastore.
NAME	Name of the object.
VERSION	Indicates the number of times this object has been updated.
TYPE	Subtype of the object.
OWNER	For table objects, the owner.
DATASTORE	For table objects, the datastore to which they belong.
NORMNAME	Unique name for this object in this table.

Example

The following query returns all the data flows in the repository:

```
SELECT OBJECT_KEY, NAME
```

```
    FROM AL_LANG A
    WHERE OBJECT_TYPE = 1
        AND VERSION =
            (SELECT MAX(VERSION) FROM AL_LANG B
            WHERE A.NORMNAME = B.NORMNAME)
```

Parent topic: Internal metadata tables and views [page 1335]

Related Information

AL_LANGXMLTEXT [page 1337]
AL_ATTR [page 1338]
AL_SETOPTIONS [page 1339]
AL_USAGE [page 1340]
ALVW_FUNCINFO [page 1344]
ALVW_PARENT_CHILD [page 1345]

9.3.2 AL_LANGXMLTEXT

The AL_LANGXMLTEXT table contains SAP Data Services XML-related objects that are also displayed in the object library.

The following table contains descriptions of the columns in the AL_LANGXMLTEXTA table.

Column Name	Description
OBJECT_KEY	Internal ID of the object.
DATE_MODIFIED	Date the object was last modified.
OBJECT_TYPE	Type of object (an integer). Query the AL_REPO-TYPE_NAMES table to find the corresponding name for this type. Examples: data flow, table, datastore.
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.

Parent topic: Internal metadata tables and views [page 1335]

Related Information

AL_LANG [page 1336]
AL_ATTR [page 1338]
AL_SETOPTIONS [page 1339]
AL_USAGE [page 1340]
ALVW_FUNCINFO [page 1344]
ALVW_PARENT_CHILD [page 1345]

9.3.3 AL_ATTR

The AL_ATTR table contains object attributes stored in the SAP Data Services repository.

The following table contains descriptions of the columns in the AL_ATTR table.

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)
```

Parent topic: Internal metadata tables and views [page 1335]

Related Information

AL_LANG [page 1336]

AL_LANGXMLTEXT [page 1337]
AL_SETOPTIONS [page 1339]
AL_USAGE [page 1340]
ALVW_FUNCINFO [page 1344]
ALVW_PARENT_CHILD [page 1345]

9.3.4 AL_SETOPTIONS

The AL_SETOPTIONS table contains option settings for all objects in the SAP Data Services repository.

The following table contains descriptions of the columns in the AL_SETOPTIONS table.

Column Name	Data type	Description
PARENT_OBJ_ID	numeric(38)	ID of the parent object to which this option belongs.
PARENT_OBJ_TYPE	numeric(38)	Type of the parent object. Examples: batch job or data flow.
CALL_OBJID	varchar(100)	ID of the calling object. (Future use)
CALL_OBJTYPE	varchar(255)	Type of the calling object. (Future use)
OPTION_NAME	varchar(100)	Name of the option.
OPTION_VALUE	varchar(255)	Value of the option.
OVERFLOW_KEY	numeric(38)	KEY value pointing to a row in the AL_OVERFLOW_ATTR table. When an OPTION_VALUE exceeds 255 characters, Data Services adds the remaining characters to AL_OVERFLOW_ATTR and stores the row ID as OVERFLOW_KEY in the AL_SETOPTIONS table.

Parent topic: Internal metadata tables and views [page 1335]

Related Information

AL_LANG [page 1336]
AL_LANGXMLTEXT [page 1337]
AL_ATTR [page 1338]

AL_USAGE [page 1340]
ALVW_FUNCINFO [page 1344]
ALVW_PARENT_CHILD [page 1345]

9.3.5 AL_USAGE

The AL_USAGE table contains the call hierarchy for objects in the SAP Data Services repository.

> **Example**
>
> If a table is used in a data flow, which is called from a work flow, then rows appear in the AL_USAGE table that associate the work flow (parent) to the table (descendent).

The AL_USAGE table is identical to the ALVW_PARENT_CHILD view with two exceptions:

- The AL_USAGE table captures the entire call hierarchy
- The AL_USAGE table has a column named DEPTH

The DEPTH column is unique to the AL_USAGE table.

> **Note**
>
> To populate the AL_USAGE table, use the *Calculate Usage Dependencies* command.

The following table contains the descriptions for the columns in the AL_USAGE table.

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 isn't unique, Data Services generates a unique numeric suffix and appends that to the given name.
DESCEN_OBJ_TYPE	varchar(64)	Type of the called object. Examples: data flow, table, function, file.
DESCEN_OBJ_DESC	varchar(255)	Description associated with the called object.
DESCEN_OBJ_USAGE	varchar(20)	Applies only to tables and files. How the child is used. Examples: source, target, lookup table.

Column Name	Data type	Description
DESCEN_OBJ_KEY	int(4)	Key in AL_LANG of the descendant object.
DESCEN_OBJ_DS	varchar(64)	Applies only to tables and files. The datastore of the child object.
DESCEN_OBJ_OWNER	varchar(256)	Owner of the child table.
DEPTH	int	Indicates the number of levels between objects in a parent/descendent relationship.

Parent topic: Internal metadata tables and views [page 1335]

Related Information

AL_LANG [page 1336]
AL_LANGXMLTEXT [page 1337]
AL_ATTR [page 1338]
AL_SETOPTIONS [page 1339]
ALVW_FUNCINFO [page 1344]
ALVW_PARENT_CHILD [page 1345]

9.3.5.1 Querying the AL_USAGE table: Example use cases

Create queries using SQL statements to obtain information from AL_USAGE, such as a list of objects related to a specific table, or a list of objects that call a specified object.

Query to list objects related to a specific target table

Example

The following table is a report generated by a query of the AL_USAGE table. The query generated a list of objects related to the target table SALES_ORDER.

Parent_Obj	Parent_Obj_Type	Descen_Obj	Descen_Obj_Type	Descen_Obj_Desc	Descen_Obj_Usage
Build_Fact	data flow	SALES_ORDER	Table	Sales order target fact table	Target

Parent_Obj	Parent_Obj_Type	Descen_Obj	Descen_Obj_Type	Descen_Obj_Desc	Descen_Obj_Usage
Daily_Job	work flow	SALES_ORDER	Table	Sales order target fact table	Target
DF_NewOrders	Job	SALES_ORDER	Table	Sales order target fact table	Target
Get_IDoc	data flow	SALES_ORDER	Table	Sales order target fact table	Target
IDoc_job	data flow	SALES_ORDER	Table	Sales order target fact table	Target
initial_Job	Job	SALES_ORDER	Table	Sales order target fact table	Target

Query to list which jobs load to a specific table

Example

The following query returns the jobs that 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>'
```

Query to list objects dependent on a specific source

Example

The following query returns a list of objects that are affected when you drop a source table:

```
SELECT PARENT_OBJ, PARENT_OBJ_TYPE
    FROM AL_USAGE
    WHERE DESCEN_OBJ_TYPE = 'Table'
        AND DESCEN_OBJ = '<targetTable>'
```

Produce a “where used” report for an object

Example

The following query produces a list of objects that call the target table SALES_ORDER.

```
SELECT
            AL_USAGE.PARENT_OBJ,
            AL_USAGE.PARENT_OBJ_TYPE,
            AL_USAGE.DESCEN_OBJ,
            AL_USAGE.DESCEN_OBJ_DESC,
            AL_USAGE.DESCEN_OBJ_TYPE,
            AL_USAGE.DESCEN_OBJ_USAGE
    FROM AL_USAGE
    WHERE (AL_USAGE.DESCEN_OBJ_TYPE = 'Table'
            AND AL_USAGE.DESCEN_OBJ='SALES_ORDER')
```

The following table shows the results:

Parent_Obj	Parent_Obj_Type	Descen_Obj	Descen_Obj_Type	Descen_Obj_Desc	Descen_Obj_Usage
Build_Fact	data flow	SALES_ORDER	Table	Sales order target fact table	Target
Daily_Job	work flow	SALES_ORDER	Table	Sales order target fact table	Target
DF_NewOrders	Job	SALES_ORDER	Table	Sales order target fact table	Target
Get_IDoc	data flow	SALES_ORDER	Table	Sales order target fact table	Target
IDoc_job	data flow	SALES_ORDER	Table	Sales order target fact table	Target
initial_Job	Job	SALES_ORDER	Table	Sales order target fact table	Target

Query to show the jobs and related targets

Example

The following query returns all jobs in a repository and their targets:

```
SELECT
    AL_USAGE.PARENT_OBJ,
    AL_USAGE.PARENT_OBJ_TYPE,
    AL_USAGE.DESCEN_OBJ,
    AL_USAGE.DESCEN_OBJ_DESC,
    AL_USAGE.DESCEN_OBJ_TYPE,
    AL_USAGE.DESCEN_OBJ_USAGE
    FROM AL_USAGE
    WHERE (AL_USAGE.PARENT_OBJ_TYPE = 'Job'
```

```
        AND AL_USAGE.DESCEN_OBJ_TYPE = 'Table'
        AND AL_USAGE.DESCEN_OBJ_USAGE = 'Target')
```

The following table shows the results:

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

9.3.6 ALVW_FUNCINFO

The ALVW_FUNCINFO view contains a list of functions that you define in SAP Data Services and functions that you import into the Data Services repository.

The following table descriptions for the columns in the ALVW_FUNCINFO view.

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.

Parent topic: Internal metadata tables and views [page 1335]

Related Information

AL_LANG [page 1336]
AL_LANGXMLTEXT [page 1337]
AL_ATTR [page 1338]
AL_SETOPTIONS [page 1339]
AL_USAGE [page 1340]

ALVW_PARENT_CHILD [page 1345]

9.3.7 ALVW_PARENT_CHILD

The ALVW_PARENT_CHILD view contains information about parent objects that contain or call other children objects.

The following table contains descriptions for the columns in the ALVW_PARENT_CHILD view.

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

Query the ALVW_PARENT_CHILD view for the following information:

- To view the data flows that load table `EMP`:

```
SELECT PARENT_OBJ
    FROM ALVW_PARENT_CHILD
    WHERE DESCEN_OBJ_TYPE = 'Table'
        AND DESCEN_OBJ = 'EMP'
        AND DESCEN_OBJ_USAGE = 'Target'
```

- To view the data flows that are 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'
```

Parent topic: Internal metadata tables and views [page 1335]

Related Information

AL_LANG [page 1336]
AL_LANGXMLTEXT [page 1337]
AL_ATTR [page 1338]
AL_SETOPTIONS [page 1339]
AL_USAGE [page 1340]
ALVW_FUNCINFO [page 1344]

9.4 Operational metadata tables and views

Operational metadata includes statistics, for example, statistics about job executions, and data quality, from the SAP Data Services repository.

AL_HISTORY [page 1346]
: The AL_HISTORY table contains statistics from the job executions for jobs in the SAP Data Services repository.

DQVW_AGR_RPTSTATS [page 1348]
: The DQVW_AGR_RPTSTATS view contains aggregated data quality statistics data.

ALVW_FLOW_STAT [page 1349]
: The ALVW_FLOW_STAT view contains information about the execution statistics of transforms within data flows.

DQVW_REPORTS_STAT [page 1349]
: The DQVW_REPORTS_STAT view contains information about report statistics for jobs.

9.4.1 AL_HISTORY

The AL_HISTORY table contains statistics from the job executions for jobs in the SAP Data Services repository.

The following table contains descriptions for the columns in the AL_HISTORY table.

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 (D).
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)
```

Parent topic: Operational metadata tables and views [page 1346]

Related Information

DQVW_AGR_RPTSTATS [page 1348]
ALVW_FLOW_STAT [page 1349]
DQVW_REPORTS_STAT [page 1349]

9.4.2 DQVW_AGR_RPTSTATS

The DQVW_AGR_RPTSTATS view contains aggregated data quality statistics data.

The following table describes the columns in the DQVW_AGR_RPTSTATS view.

Column	Description
SERVICE_ID	
JOB_KEY	Identifier that represents a job run. Whenever a job is executed, a new ID is given.
OBJECT_ID	GUID (globally unique identifier) assigned to a transform.
OBJECT_NAME	Name of the object in the data flow.
JOB_NAME	Name of the executed job.
OBJECT_TYPE	Type of the object in the data flow.
DATAFLOW_NAME	Name of the data flow.
PATHS	Position of the object in the data flow.
ROW_COUNT	Number of rows in object.
EXECUTION_TIME	Difference between start time and end time in seconds.
START_TIME	Time when the job was launched.
END_TIME	Time when the job completed.

Parent topic: Operational metadata tables and views [page 1346]

Related Information

AL_HISTORY [page 1346]
ALVW_FLOW_STAT [page 1349]
DQVW_REPORTS_STAT [page 1349]

9.4.3 ALVW_FLOW_STAT

The ALVW_FLOW_STAT view contains information about the execution statistics of transforms within data flows.

The following table contains descriptions for the columns in the ALVW_FLOW_STAT view.

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.

Parent topic: Operational metadata tables and views [page 1346]

Related Information

AL_HISTORY [page 1346]
DQVW_AGR_RPTSTATS [page 1348]
DQVW_REPORTS_STAT [page 1349]

9.4.4 DQVW_REPORTS_STAT

The DQVW_REPORTS_STAT view contains information about report statistics for jobs.

The following table contains descriptions of the columns in the DQVW_REPORTS_STAT view.

Column	Description
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.
OBJECT_ID	GUID (globally unique identifier) assigned to a transform.
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.
PATH	Position of the object in the data flow.
OBJECT_NAME	Name of the object in the data flow.
OBJECT_TYPE	Type of the object in the data flow.
ROW_COUNT	Number of rows in object.
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.
DATAFLOW_NAME	Name of the dataflow.

Parent topic: Operational metadata tables and views [page 1346]

Related Information

AL_HISTORY [page 1346]
DQVW_AGR_RPTSTATS [page 1348]
ALVW_FLOW_STAT [page 1349]

10 Data quality reports and statistics tables

There are two types of data quality repository tables: regulatory and informative tables and reports, and data quality statistics tables.

Data quality tables in the repository contain job processing statistics.

Some transforms create automatic reports when you enable reports in the transform editor. The software creates the reports based on the availability of specific statistics in repository tables. These reports may be regulatory, like the AMAS form for Australia address matching processing, or informative, like the U.S. Addressing Report. The following list contains the regulatory and informative tables:

- Data Cleanse
- DSF2 Walk Sequencer
- Geocoder
- Global Address Cleanse
- Match
- USA Regulatory Address Cleanse

The data quality statistic tables populated by the Data Cleanse, Global Address Cleanse, and Geocoder transforms identify significant cleanse process changes based on the entities. The data quality statistics tables contain summary and record-level statistics.

The following list contains the data quality statistics tables that include information about the assignment process:

- CLEANSE_ADDRESS_RECORD_INFO_ (available for Global Address Cleanse)
- CLEANSE_CHANGE_INFO_ (available for Data Cleanse and Global Address Cleanse)
- CLEANSE_COMPONENT_INFO_ and the TASK_COLUMN_DEFINITIONS_ (available after 4.2.6.0 release for Data Cleanse and Global Address Cleanse)
- CLEANSE_INFO_CODES_ (available for Data Cleanse and Global Address Cleanse)
- CLEANSE_STATISTICS_ (available for Data Cleanse and Global Address Cleanse)
- GEOCODE_INFO_CODES_ (available for Geocoder)
- GEOCODE_STATISTICS_ (available for Geocoder)

The TASK_LOCALIZATION files available for download provide the descriptions for the data included in some of the columns.

Enable statistics in the applicable transform option editor. The software populates the corresponding repository tables with statistics when you run your job.

For information about controlling the history and log file retention periods, see the topic "History file and log file retention" in the *Administrator Guide*.

Related Information

Data quality statistics tables and supplemental content information [page 1402]
Repository tables and related reports [page 1353]
Match repository statistics tables [page 1383]

10.1 Repository tables common columns

The following table contains descriptions for columns that are common to most of the repository tables and the data quality statistics tables.

These columns, combined with a unique column from the table, make up the primary key for each table. These columns contain platform-generated data. Each repository 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.
ROW_ID	INT	An internal identification generated by the transform that uniquely identifies a row processed by that transform. Appears in each non-summary data quality statistics table.
TABLE_ID	INT	An internal identification generated by the transform that uniquely identifies the source table.

Parent topic: Data quality reports and statistics tables [page 1351]

Related Information

Repository tables and related reports [page 1353]
Data quality statistics tables and supplemental content information [page 1402]

10.2 Repository tables and related reports

The software generates many reports automatically when you enable them in the transforms.

Enable various reports in the USA Regulatory Address Cleanse, DSF2 Walk Sequencer, Global Address Cleanse, and Match transforms. The following table lists the transform-generated report names, the corresponding repository table name, and the applicable transforms.

Report	Repository table	Transform(s)
US CASS report: USPS Form 3553	PSFORM3553DATA	USA Regulatory Address Cleanse
NCOALink Processing Summary report	NCOALINKSUMMARY MEDSTATS NCOACERTIFICATIONDATA	USA Regulatory Address Cleanse
Customer Service Log	CSLSTATS	USA Regulatory Address Cleanse
Broker and List Administrator file	PAFBALASTATS	USA Regulatory Address Cleanse
Delivery Sequence Invoice report	DSFSEQUENCESTATS	DSF2 Walk Sequencer
US Addressing Report	DPVLACSLINKSUMMARY	USA Regulatory Address Cleanse
US Regulatory Locking Report	USREGULATORYLOCKING	USA Regulatory Address Cleanse
Canada SERP report: Statement of Address Accuracy	SERPADDRACCURACY	Global Address Cleanse
Australia AMAS report: Address Matching Processing Summary	AMASADRPROCSUMMARY	Global Address Cleanse
Address Information Code Sample report	ADDRINFOCODEDATA	Global Address Cleanse USA Regulatory Address Cleanse
Address Information Code Summary report	ADDRINFOCODESUMMARY ADDRSTATUSCODEDATA	Global Address Cleanse USA Regulatory Address Cleanse
Address Validation Summary report	ADDRVALIDATESUMMARY	Global Address Cleanse USA Regulatory Address Cleanse
Address Type Summary report	ADDRTYPESUMMARY	Global Address Cleanse USA Regulatory Address Cleanse
Address Standardization Sample	ADDRINFOCODEDATA	Global Address Cleanse USA Regulatory Address Cleanse

Report	Repository table	Transform(s)
Geocoder Summary Report (part of the US Addressing report)	GEO_ASSIGN_LEVEL GEO_INFO_CODE	Geocoder transform USA Regulatory Address Cleanse
Address Quality Code Summary	ADDRINFOCODESUMMARY	Global Address Cleanse
Best Record Summary	MTBRINFO MTBRACTION	Match
Match Contribution Report	MTRULESRES MTBRKGRPINFO MTBRKGRP	Match
Match Criteria Summary report	MTCRITINFO MTCRITDEF MTKEYDEF MTCMPCRIT	Match
Match Source Stats Summary report	MTGSSRCSTS MTGSSRCBYSRCSTS	Match
Match Duplicate Sample report	MTDUPESDATA	Match
Match Input Source Output Select Report	MTINSRCINFO	Match
Match Multi-Source Frequency Report	Match statistics tables	Match
New Zealand SOA	SENDRIGHTADDRACCURACY	Global Address Cleanse
Data Cleanse Information Code Summary report	UDCINFOCODESUMMARY	Data Cleanse
Data Cleanse Status Code Summary report	UDCSTATUSCODESUMMARY	Data Cleanse

Repository tables for USA and Global address cleanse [page 1355]

Repository tables and reports for Data Cleanse [page 1378]

The following table contains a list of repository tables used for statistical reports related to status and information codes generated from the Data Cleanse transform.

Repository tables and reports for Geocoder [page 1381]

The Geocoder transform generates statistics that the software uses in US postal reporting.

Match repository statistics tables [page 1383]

Parent topic: Data quality reports and statistics tables [page 1351]

Related Information

Repository tables common columns [page 1352]
Data quality statistics tables and supplemental content information [page 1402]
Data Quality Reports
Data quality statistics tables and supplemental content information [page 1402]

10.2.1 Repository tables for USA and Global address cleanse

The following table contains a list of repository tables used by the software to create postal regulatory and statistical reports for the USA Regulatory Address Cleanse and Global Address Cleanse transforms. The sections following this chart contain a topic for each table with descriptions for the fields (columns) in the table.

Repository table name	Description	Transform
ADDRINFOCODEDATA	Contains statistics about each record that generated an address information code during processing. Used for the Address Information Code Sample report.	Global Address Cleanse USA Regulatory Address Cleanse
ADDRINFOCODESUMMARY	Contains statistics about each information code found during processing. Used for the Address Information Code Summary and the Address Quality Code Summary reports.	Global Address Cleanse USA Regulatory Address Cleanse
ADDRSTATUSCODEDATA	Contains status codes found by the transform and their descriptions. Used for the Address Information Code Sample Report.	Global Address Cleanse USA Regulatory Address Cleanse
ADDRTYPESUMMARY	Contains statistics about address types found during processing. Used for the Address Type Summary.	Global Address Cleanse USA Regulatory Address Cleanse
ADDRVALIDATESUMMARY	Contains record validation statistics found by the transform. Used for the Address Validation Summary.	Global Address Cleanse USA Regulatory Address Cleanse
AMASADRPROCSUMMARY	Contains summary statistics for address processing. Used for the AMAS (Australia Matching Approval System) Address Matching Processing Summary.	Global Address Cleanse
CSLSTATS	Contains statistics from NCOALink processing. Used for the NCOALink CSL (Customer Service Log).	USA Regulatory Address Cleanse

Repository table name	Description	Transform
DPVLACSLINKSUMMARY	Contains statistics about the DPV, DSF2, SuiteLink, and LACSLink processing. Used for the U.S. Addressing Report.	USA Regulatory Address Cleanse
DSFAUGMENTSTATS	Contains a detailed record plus the DSF2 licensee name and processing date. Used in the DSF2 Augment Statistics Log File.	USA Regulatory Address Cleanse
DSFSEQUENCESTATS	Contains DSF2 sequence statistics per postcode/sortcode combination. Used in the Delivery Sequence Invoice Report.	USA Regulatory Address Cleanse
MEDSTATS	Contains statistics for NCOALink move effective dates. Used for the NCOALink Report.	USA Regulatory Address Cleanse
NCOALCERTIFICATIONDATA	Contains statistics required for NCOALink processing. Used for the NCOALink Processing Summary Report.	USA Regulatory Address Cleanse
NCOALINKSUMMARY	Contains summary statistics required for NCOALink processing. Used for the NCOALink Processing Summary Report.	USA Regulatory Address Cleanse
PAFBALASTATS	Contains statistics for NCOALink processing. Used for the Processing Acknowledgement Form and the Broker and List Administrator file.	USA Regulatory Address Cleanse
PSFORM3553DATA	Contains data for the USPS Form 3553 report that is submitted with all CASS-certified mailings.	USA Regulatory Address Cleanse
SERPADDRACCURACY	Contains statistics about the accuracy of the addresses in the list. Used for the Statement of Address Accuracy report (Canada).	Global Address Cleanse
SENDRIGHTADDRACCURACY	Contains summary statistics for New Zealand address processing. 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

ADDRINFOCODEDATA [page 1357]

ADDRINFOCODESUMMARY [page 1358]

Parent topic: Repository tables and related reports [page 1353]

Related Information

Repository tables and reports for Data Cleanse [page 1378]
Repository tables and reports for Geocoder [page 1381]
Match repository statistics tables [page 1383]

10.2.1.1 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).

> **i Note**
>
> The OBJECT_KEY column is included in this table but it is not listed. it is a primary key common to most tables. The OBJECT_ID column does not apply to this table.

Related Information

Repository tables common columns [page 1352]

10.2.1.2 ADDRINFOCODESUMMARY

This table contains statistics about each information code found during processing. The information is used for the Address Information Code Summary report (USA Regulatory Address Cleanse transform) and the Address Quality Code Summary (Global Address Cleanse transform).

Column	Data type definition	Description
COUNTRYID	VARCHAR (2)	Country code applicable to the listed INFOCODE.
INFOCODE	VARCHAR (4)	Information code.
INFOCOUNT	INT	Total number of records in the list that received the listed INFOCODE.
ENGINENAME	CHAR (50)	Name of the engine applicable to the listed INFOCODE. For the Global Address Cleanse transform, it is the name of the engine that processed the data (USA, CANADA, GLOBAL_ADDRESS, or GLOBAL_ADDRESS_CJK). For the USA Regulatory Address Cleanse transform, it is always USA.
DATA_SOURCE_ID	VARCHAR (80)	Code that uniquely identifies the list that contains the listed INFOCODE.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.1.3 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).

> **i Note**
>
> The OBJECT_KEY column is included in this table but it is not listed. it is a primary key common to most tables. The OBJECT_ID column is not applicable to this table.

Related Information

Repository tables common columns [page 1352]

10.2.1.4 ADDRTYPESUMMARY

The ADDRTYPESUMMARY 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).

The following table lists the columns in alphabetical order.

Column	Data type definition	Description
BLDNAMERECS	INT	Number of records that were determined to be building name addresses (Global Address Cleanse transform).
COUNTRYID	CHAR (2)	Country code applicable to the listed address type.

Column	Data type definition	Description
DATA_SOURCE_ID	NVARCHAR (80)	Unique identification code assigned to the list.
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.
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.
HIGHRISERECS	INT	Number of records that were determined to be highrise addresses.
MILITARYRECS	INT	Number of records that were determined to be military addresses.
MOBILEROUTERECS	INT	Number of records that were determined to be mobile route addresses (Canadian addresses, Global Address Cleanse transform).
POSTALRECS	INT	Number of records that were determined to be post office box addresses.
PROCESSEDRECS	INT	Number of records processed by the transform.
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.
SUITEL_TYPE	INT	Pre-SuiteLink or post-SuiteLink processing. Values include: 1 (Post-SuiteLink processing) -1 (Pre-SuiteLink processing)
UNDERTERMINDRECS	INT, NULL	Number of records that were determined to be non US records (USA Regulatory Address Cleanse transform).
UNIQSUBURBANRECS	INT	Number of records that were determined to be unique suburban addresses.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.1.5 ADDRVALIDATESUMMARY

The ADDRVALIDATESUMMARY 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.

The following table lists the columns in alphabetical order.

Column	Data type definition	Description
ADDEDRECS	INT	Number of records processed by the transform.
COMPONENT-NAME	NVARCHAR (64)	Address component (ADDRESS1, LOCALITY1, and so on) to which the counts and percentages apply.
CORRECTEDRECS	INT	Number of records that were corrected during processing for the applicable COMPONENTNAME.
COUNTRYID	CHAR (2)	Country code applicable to the COMPONENTNAME column.
DATA_SOURCE_ID	NVARCHAR (80)	Unique identification code assigned to the list.
DELETEDRECS	INT	Number of records that were deleted during processing for the applicable COMPONENTNAME.
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.
SUITEL_TYPE	INT	Indicates whether the count information applies to pre-SuiteLink or post-SuiteLink processing, or pre-NCOALink or post-NCOALink processing. Values include: 1 (post-SuiteLink/NCOALink processing) -1 (pre-SuiteLink/NCOALink processing)
UNCHANGEDRECS	INT	Number of records that remained unchanged for the applicable COMPONENTNAME.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.1.6 CSLSTATS

The CSLSTATS table contains statistics for NCOALink processing for the Customer Service Log.

It is applicable for the USA Regulatory Address Cleanse transform. The following table lists the columns in alphabetical order.

Column	Data type definition	Description
DATA_SOURCE_ID	NVARCHAR (80)	Unique identification code assigned to the list.
DTLRECORD	NVARCHAR (1000)	String of Customer Service Log field values.
LICENSEEID	NVARCHAR (4)	NCOALink licensee's identification number assigned by the USPS.
LICENSEETYPE	NVARCHAR (1)	Type of service provider. Values are: E (End user) F (Full service provider) L (Limited service provider)
PROCESSINGDATE	DATETIME	Date and time the job was processed through the software in YYYY-MM-DD HR:MIN:SEC format.
RESERVED_COUNT1	INT	Reserved for future record counts.
RESERVED_COUNT2	INT	Reserved for future record counts.
RESERVED_COUNT3	INT	Reserved for future record counts.
RESERVED_STRING1	NVARCHAR (100)	Reserved for future strings.
RESERVED_STRING2	NVARCHAR (100)	Reserved for future strings.
RESERVED_STRING3	NVARCHAR (100)	Reserved for future strings.
SEQNUM	INT	DSF2 sequence number assigned to the address.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.1.7 DPVLACSLINKSUMMARY

The DPVLACSLINKSUMMARY table contains statistics about the DPV, DSF2, LACSLink, and SuiteLink processing.

The software uses the information in the US Addressing Report, which applies to the USA Regulatory Address Cleanse transform. The following table contains the columns in alphabetical order.

Column	Data type definition	Description
DATA_SOURCE_ID	NVARCHAR (80)	A unique identification for a list.
DPVNOTVALIDATED	INT	Number of records that were not validated for DPV.
DPVVALIDATE_NOSTATS	INT	Number of records that have a No Stat indicator.
DPVVALIDATED_CMRA	INT	Number of records that were validated as a CMRA (Commercial Mail Receiving Agency).
DPVVALIDATED_D	INT	Number of records that did not have a secondary range (DPV status of D, secondary range not present).
DPVVALIDATED_DNA	INT	Number of records that were validated as DNA (Door Not Accessible).
DPVVALIDATED_NSL	INT	Number of records that were validated as NSL (No Secure Location).
DPVVALIDATED_PBSA	INT	Reserved for future use.
DPVVALIDATED_S	INT	Number of records that either contained an invalid secondary range or contained an invalid trailing alpha in primary range. (DPV status of S, dropped secondary range or trailing alpha on primary range.)
DPVVALIDATED_VACANT	INT	Number of records that have a Vacant indicator.
DPVVALIDATED_Y	INT	Number of records that were validated for DPV (status of Y, primary and secondary range is valid).
DSF2_BUSINESS	INT	Number of records that have a business address.
DSF2_CENTRAL	INT	Number of records that have a central delivery indicator.
DSF2_CURB	INT	Number of records that have a curb-side delivery indicator.
DSF2_DOOR	INT	Number of records that have a door-slot delivery indicator.
DSF2_DROP	INT	Number of records that are dropped at a delivery point that serves businesses or families (for example, a CMRA).
DSF2_EDUCATIONAL	INT	Number of records that are educational institutions.
DSF2_NDCBU	INT	Number of records that have an NDCBU (Neighborhood Delivery Centralized Box Unit) delivery indicator.
DSF2_SEASONAL	INT	Number of records that are seasonally occupied.

Column	Data type definition	Description
DSF2_THROWBACK	INT	Number of records that are PO Box Throwbacks (customer wants delivery at PO Box instead of street address). **i Note** Populated for DSF2 processing and for DPV processing without DSF2 processing.
LACSCONVERTED_92	INT	Number of LACSLink records that matched after dropping the secondary number from the input address (return code 92).
LACSCONVERTED_A	INT	Number of LACSLink records that were converted by the transform (return code A).
LACSNOTCONVERTED_00	INT	Number of records that had no match to LACSLink and, therefore, no addresses converted (return code 00).
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_14	INT	Number of LACSLink records that matched to LACSLink, but couldn't be converted to a deliverable address (return code 14).
SUITEL_MATCH_A	INT	Number of records that matched to SuiteLink and had secondary information added.
SUITEL_NOMATCH_00	INT	Number of records that did not match to SuiteLink.
SUITEL_TYPE	INT	Indicates whether the information is from pre-NCOALink or post-NCOALink processing. • 1 (Pre-NCOALink processing) • -1 (Post-NCOALink processing) • 0 (NCOALink is not enabled so there is no pre-NCOALink or post-NCOALink processing sections)

i Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.1.8 DSFAUGMENTSTATS

This table contains information used in the DSF2 Augment Statistics Log file. It is applicable for the USA Regulatory Address Cleanse transform.

Column	Data type definition	Description
LICENSEE_NUMBER	NVARCHAR (4)	DSF2 licensee number.
SEQNUM	INT	Sequence number from 0000 to 9999. • If the Postcode2 field is blank, this column is blank. • If the Postcode2 field is not blank and the DPV_Status is not Y, then the column contains “0000”.
LICENSEE_NAME	NVARCHAR (40)	DSF2 licensee name.
PROCESSING_DATE	DATETIME	Date and time the job was processed through the software in YYYY-MM-DD HR:MIN:SEC format.
DTLRECORD	NVARCHAR (1000)	String of log file field values for DSF2 Augment Statistics.
DATA_SOURCE_ID	NVARCHAR (80)	Unique identification code assigned to the list.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.1.9 DSFSEQUENCESTATS

This table contains information for the Delivery Sequence Invoice Report. It is applicable for the USA Regulatory Address Cleanse transforms.

Column	Data type definition	Description
POSTCODE1	NVARCHAR (5)	Postcode (ZIP Code) for the records being processed.
SORTCODE_ROUTE	NVARCHAR (4)	Sortcode route (carrier route) for the records being processed.
LICENSEE_NAME	NVARCHAR (40)	DSF2 licensee name.

Column	Data type definition	Description
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.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.1.10 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

Column	Data type definition	Description
PERIOD	NVARCHAR (2)	Time period for the number of records for each return code category. Values include the following: 1 = Months 0-3 2 = Months 4-6 3 = Months 7-12 4 = Months 13-18 5 = Months 19+
MEDCOUNT	INT	Number of moves in the specified period.
DATA_SOURCE_ID	NVARCHAR (80)	Unique identification for a list.

> **Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.1.11 NCOALCERTIFICATIONDATA

The NCOALCERTIFICATIONDATA table contains statistics from NCOALink processing and is used for the NCOALink Reports.

It is applicable for the USA Regulatory Address Cleanse transform. The following table lists the columns in alphabetical order.

Column	Data type definition	Description
ANK_ENABLED	VARCHAR (1)	ANKLink is enabled in the transform (Y/N indicator in reports).
ANK_MATCHES	INT	Number of ANKLink matches found in the list.
BUSINESS_RECORDS	INT	Number of business type records found in the list.

Column	Data type definition	Description
CONCUR-RENT_PROCESSES	VARCHAR (1)	Indicates that the list was concurrently processed and the data modifications that ocurred. Values are: N (None). Y (Yes, but with no data modifications). D (Yes, data modifications from sources other than postal data). P (Yes, data modifications from postal data only. For example, ZIP+4 or DPV). B (Yes, data modifications from postal and other sources).
DATA_RETURNED	VARCHAR (1)	Purpose for NCOALink processing. C (Change of address) F (Return codes) S (Statistics)
DATA_SOURCE_ID	VARCHAR (80)	Unique identification code assigned to the list.
DPV_ENABLED	VARCHAR (1)	DPV is enabled in the transform (Y/N indicator in reports).
DPV_MATCHES	INT	Number of DPV matches found in the list.
FAMILY_RECORDS	INT	Number of family type records found in the list.
INDIVIDUAL_RE-CORDS	INT	Number of individual type records found in the list.
LACSL_ENABLED	VARCHAR (1)	LACSLink is enabled in the transform (Y/N indicator in reports).
LACSL_MATCHES	INT	Number of LACSLink matches found in the list.
LICENSEE_COMPANY	VARCHAR (50)	Broker or list administrator name.
LICENSEE_ID	VARCHAR (4)	Unique ID for the broker or list administrator.
LIST_NAME	VARCHAR (30)	Name of the list being processed.
LIST_RETURN_DATE	VARCHAR (10)	Date obtained from the *List returned date* option in the USPS License Information group.
MAIL_CLASS	VARCHAR (1)	Mail class that was processed.
MAILER_COMPANY	VARCHAR (30)	Name of the customer that requested NCOALink processing.
MAILER_PAF_ID	VARCHAR (40)	Combination of the following information: • USPS-assigned license ID (first 4 characters) • List Owner NAICS Code (next 5 characters) • Frequency of processing (next 2 characters) • Licensee-assigned List ID (last 6 characters)

Column	Data type definition	Description
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).
NCOAL_MATCHES	INT	Number of NCOALink matches found in the list.
NCOAL_NAME	VARCHAR (16)	Name of the NCOALink-certified software.
NCOAL_PROC-ESS_DATE	VARCHAR (10)	Date based on the time that the list was processed. Automatically generated by the software, and included in reports.
NCOAL_VERSION	VARCHAR (16)	Version of the NCOALink-certified software.
POST_PROCESSES	VARCHAR (1)	Indicates that the list was post processed and what data modifications ocurred. Values are: N (None). Y (Yes, but with no data modifications). D (Yes, data modifications from sources other than postal data). P (Yes, data modifications from postal data only. For example, LACSLink). B (Yes, data modifications from postal and other sources).
PRE_PROCESSES	VARCHAR (1)	Indicates that the list was pre processed and what data modifications ocurred. Values are: N (None) Y (Yes, but with no data modifications). D (Yes, data modifications from sources other than postal data). P (Yes, data modifications from postal data only. For example, ZIP+4 or DPV) B (Yes, data modifications from postal and other sources)
PROCESS_FIRST	VARCHAR (1)	First Class mail was processed by the transform. (Y/N indicator in reports.)
PROCESS_PACKAGE	VARCHAR (1)	Package Services mail was processed by the transform. (Y/N indicator in reports.)
PROCESS_PERIODI-CAL	VARCHAR (1)	Periodicals 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.)
PROCESSES_USED	VARCHAR (768)	All USPS processes used to obtain the final data results.

Column	Data type definition	Description
PROCESSING_CATEGORY	VARCHAR (20)	Reason for the NCOALink processing. Values are: **Option** / **Printed on report** Marketing / MKTG TEST Normal / NORMAL Stage I / STAGE I Stage II / STAGE II System testing / SYS TEST
RESERVED_COUNT1	INT	Extra field for additional counts.
RESERVED_COUNT2	INT	Extra field for additional counts.
RESERVED_COUNT3	INT	Extra field for additional counts.
RESERVED_STRING1	VARCHAR (100)	Extra field for additional string.
RESERVED_STRING2	VARCHAR (100)	Extra field for additional string.
RESERVED_STRING3	VARCHAR (100)	Extra field for additional string.
SERVICE_PROVIDER	VARCHAR (1)	NCOALink list processor's provider level. Values are: 0 End user 1 Limited service provider 2 Full service provider
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).
SUITEL_ENABLED	VARCHAR (1)	SuiteLink is enabled in the transform (Y/N indicator in reports).
SUITEL_MATCHES	INT	Number of SuiteLink matches found in the list.
TOTAL_RECORDS	INT	Number of records processed by the transform.
USE_BUSINESS_MOVES	VARCHAR (1)	Business 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).
USE_INDIVIDUAL_MOVES	VARCHAR (1)	Individual moves were included in processing (Y/N indicator in reports).
ZIP4_MATCHES	INT	Number of ZIP+4 matches found in the list.

> **Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.1.12 NCOALINKSUMMARY

This table contains summary statistics required for NCOALink processing. The information is used for the NCOALink Summary Report. It is applicable for the USA Regulatory Address Cleanse transform.

Column	Data type definition	Description
CODE	NVARCHAR (5)	Return code.
CODEGROUP	NVARCHAR (1)	Group in which the code applies. 1 = New address provided by NCOALink 2 = Found COA 3 = Cannot match COA 4 = From the daily delete process
CODEDESC	NVARCHAR (50)	Description for the return code listed.
RECCOUNT	INT	Number of list records found for the code listed.
ADDRESS_PROVIDED	NVARCHAR (1)	Indicates whether an address was provided (Y/N).
ADDRESS_SOURCE	NVARCHAR (1)	D = Derived by data S = Derived from software
DETAILED_DESC	VARCHAR (1)	Provides a more detailed description of the listed code.
DATA_SOURCE_ID	NVARCHAR (80)	Unique identification code assigned to a list.

> **Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.1.13 PAFBALASTATS

This table contains statistics for NCOALink processing. The information is used for the Processing Acknowledgement Form and the Broker and List Administrator file. It is applicable for the USA Regulatory Address Cleanse transform.

Column	Data type definition	Description
LICENSEEID	NVARCHAR (4)	Service provider's NCOALink license ID.
DTLRECORD	NVARCHAR (700)	All field values per list.
LOGTYPE	NVARCHAR (1)	The type of log file: P = PAF B = BALA
SEQNUM	INT	DSF2 sequence number assigned to the address.
RESERVED_COUNT1	INT	Reserved for future record counts.
RESERVED_COUNT2	INT	Reserved for future record counts.
RESERVED_COUNT3	INT	Reserved for future record counts.
RESERVED_STRING1	NVARCHAR (255)	Reserved for future text.
RESERVED_STRING2	NVARCHAR (255)	Reserved for future text.
RESERVED_STRING3	NVARCHAR (255)	Reserved for future text.
DATA_SOURCE_ID	NVARCHAR (80)	Unique identification for a list.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.1.14 PSFORM3553DATA

The PSFORM3553DATA table contains data for the USPS Form 3553 report that is submitted with all CASS-certified mailings.

The following table lists the columns in alphabetical order.

Column	Data type definition	Description
ADDRESSLISTNAME	NVARCHAR (20)	Name of the address list processed. (Entered in *CASS Report Options* > *List Name*.)
CASSCOMPANYNAME	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.
CRISDATE	VARCHAR (15)	Not populated from the software.
CRISDBDATE	VARCHAR (15)	Not populated from the software.
CRRDATEFROM	VARCHAR (15)	Not populated from the software.
CRRTRECORDSCODED	INT	Number of records that were assigned carrier route codes.
DATA_SOURCE_ID	NVARCHAR (80)	Unique identification assigned to the list.
DPBCDATEFROM	VARCHAR (15)	Not populated from the software.
DPBRECORDSCODED	INT	Number of records with delivery point bar codes.
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.
eLOTCOMPANYNAME	NVARCHAR (40)	Name of the company that is certified for eLOT processing.
eLOTCONFIGURATION	NVARCHAR (10)	Software configuration settings for eLOT.
eLOTDATE	VARCHAR (15)	Not populated from the software.
eLOTDATEFROM	VARCHAR (15)	Not populated from the software.
eLOTDBDATE	VARCHAR (15)	eLOT directory date.
eLOTRECORDSCODED	INT	Number of records that were assigned eLOT codes.
eLOTSOFTWARE	NVARCHAR (60)	Name and version of the software that is certified for eLOT processing.
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).
FIVEDIGITDATEFROM	VARCHAR (15)	Not populated from the software.
FIVEDIGITRECCODED	INT	Number of records that were assigned 5-digit ZIP Codes.
HIGHRISEDEFAULT	INT	Number of records that were assigned as highrise defaults.
HIGHRISEEXACT	INT	Number of records that were assigned as highrise exact matches.
LACS	INT	Number of addresses that were converted through the LACSLink process.
LISTPROCESSOR	NVARCHAR (35)	MASS list processor. Not populated from the software.

Column	Data type definition	Description
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.
MASSCOMPANYNAME	NVARCHAR (40)	Not populated from the software.
MASSCONFIGURATION	NVARCHAR (10)	Not populated from the software.
MASSSOFTWARE	NVARCHAR (60)	Not populated from the software.
MLOCRSERIALNUM-BER	NVARCHAR (20)	Not populated from the software.
NUMBEROFLISTS	INT	Not populated from the software.
RDI	INT	Not populated from the software (always zero).
RESERVED_COUNT1	INT	Reserved for future record counts.
RESERVED_COUNT2	INT	Reserved for future record counts.
RESERVED_COUNT3	INT	Reserved for future record counts.
RESERVED_STRING1	NVARCHAR (100)	Reserved for future strings.
RESERVED_STRING2	NVARCHAR (100)	Reserved for future strings.
RESERVED_STRING3	NVARCHAR (100)	Reserved for future strings.
RURALROUTEDEFAULT	INT	Number of records that were assigned as rural route default matches.
RURALROUTEEXACT	INT	Number of records that were assigned as rural route exact matches.
TOTALRECPROCESSED	INT	Number of records processed in the job.
Z4CHANGEDATE	VARCHAR (15)	Not populated from the software.
Z4CHANGEDBDATE	VARCHAR (15)	Z4Change directory date.
Z4CHGRECORDS-CODED	INT	Number of records that were Z4Change coded.
Z4COMPANYNAME	NVARCHAR (40)	Name of the company that is Z4Change certified.
Z4CONFIGURATION	NVARCHAR (10)	Software configuration settings for Z4Change.
Z4SOFTWARE	NVARCHAR (60)	Name and version of the software that is certified for Z4Change processing.
ZIP4DBDATE	VARCHAR (15)	ZIP4 directory date.
ZIP4RECORDSCODED	INT	Number of ZIP+4 records that were DPV confirmed.

i Note

The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.1.15 SENDRIGHTADDRACCURACY

This table contains summary statistics about the address processing for the Global Address Cleanse SendRight certification report. The information is used for the SendRight Address Accuracy report (for New Zealand). These statistics are generated by the Global Address Cleanse transform.

Column	Data type definition	Description
CERTCOMPANYNAME	VARCHAR (40)	Name of the company that is SendRight certified. This value will be SAP Business Objects.
CERTPRODNAME	VARCHAR (40)	Name of the product that is SendRight certified. This value will be Data Services.
CERTPRODVER	VARCHAR (20)	Version of the SendRight-certified software.
PAFVERSION	VARCHAR (20)	Version of the current (most recent) PAF. PAF refers to the version number supplied by New Zealand post.
SOAUNIQUEID	VARCHAR (20)	Unique ID generated by the certified engine to guarantee a unique report for each Global Address Cleanse transform within a data flow.
CUSTOMERNAME	VARCHAR (40)	Name of the list owner or list processor.
MAILERADDR1–6	VARCHAR (60)	Name and address of the person or organization for whom you are preparing the mailing. (6 columns, 60 varchars each).
FILENAME	VARCHAR (40)	Input file name.
TOTALRECPROC	INT	Number of records processed.
UNIQUEMATCHES	INT	Number of unambiguous matches between the input addresses and one address record in the PAF.
BASEMATCHES	INT	Number of unambiguous matches between the input addresses and one base address.
DATEISSUED	DATETIME	Date that this report was generated.
DATEEXPIRED	DATETIME	Date that this report will expire. (Always exactly 1 year after generation date.)
TOTALVALIDATED	INT	Number of records validated.
ADDRESSACCURACY	VARCHAR (8)	Percentage of the total records validated from the total number of records processed.
RESERVED_COUNT1–3	INT	Reserved for future use.
RESERVED_STRING1–3	VARCHAR (255)	Reserved for future use.

10.2.1.16 SERPADDRACCURACY

This table contains statistics about the accuracy of the addresses in the list. The information is used for the SERP (Statement of Address Accuracy) report (Canada). These statistics are generated by the Global Address Cleanse transform.

Column	Data type definition	Description
VENDORNAME	VARCHAR (40)	Name of the Address Accuracy vendor. This value will be SAP Business Objects.
VENDORADDR1-2	VARCHAR (40)	Address of the Address Accuracy vendor. This will be the current SAP Business Objects address. (Two columns, 40 varchars each.)
PRODNAMEVER	VARCHAR (36)	Name of the Address Accuracy software product and version. This is hard-coded to the current product and version.
SREXPIRYDATE	VARCHAR (11)	Expiration date of the software's SERP certification.
CUSTOMERNAME	VARCHAR (40)	Company name of the organization for whom the mailing is being prepared.
CUSTOMERADDR1-4	VARCHAR (40)	Name and address of the person or organization for whom the mailing is being prepared. (Four columns, 40 varchars each.)
CUSTOMERCPCNUM	VARCHAR (15)	Customer's CPC number that is located in the Canada Post Contract.
CPCMASTERFILEVER	VARCHAR (11)	Date of the postalcode file as specified by Canada Post, which expires after 31 days.
TOTALRECPROC	INT	Number of urban and rural records processed.
RECASSIGNED	INT	Number of records assigned.
QUEST_RECS_RURAL	INT	Number of questionable rural records.
QUEST_RECS_APPT	INT	Number of questionable apartment records.
RESERVED_COUNT1–3	INT	Reserved for future record counts.
RESERVED_STRING1–3	NVARCHAR (255)	Reserved for future strings.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.1.17 USREGULATORYLOCKING

This table contains information about the record that caused DPV or LACSLink locking. The information is used for the USA Regulatory Locking Report.

Column	Data type definition	Description
LOCKMODE	INT	Whether the lock was for DPV (1) or LACSLink (2).
LOCK_CODE	VARCHAR (81)	Lock code issued by the software and found in the Alert area of the report.
POSTCODE	VARCHAR (81)	Input postalcode from the address that caused the lock.
PRIMARY_NUMBER	VARCHAR (81)	Input primary range from the address that caused the lock.
PRIMARY_NAME	VARCHAR (81)	Primary name from the address that caused the lock.
UNIT_NUMBER	VARCHAR (81)	Secondary range for the address that caused the lock.
UNIT_DESCRIPTION	VARCHAR (81)	Unit designator for the address that caused the lock.
PRIMARY_PREFIX	VARCHAR (81)	Predirectional for the address that caused the lock.
ADDRESS_TYPE	VARCHAR (81)	Suffix for the address that caused the lock.
PRIMARY_POSTFIX	VARCHAR (81)	Postdirectional for the address that caused the lock.
RESERVED_COUNT1-3	INT	Reserved for future record counts.
RESERVED_STRING1-3	NVARCHAR (255)	Reserved for future strings.
DATA_SOURCE_ID	NVARCHAR (80)	Unique identifier for the list.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.2 Repository tables and reports for Data Cleanse

The following table contains a list of repository tables used for statistical reports related to status and information codes generated from the Data Cleanse transform.

Repository table name	Description	Transform
UDCSTATUSCODESUMMARY	Lists status codes that were generated during a specific data cleanse job and the total number or records assigned to that code. Used for the Data Cleanse Status Code Summary report	Data Cleanse
UDCINFOCODESUMMARY	Lists information codes that were generated during a specific data cleanse job and the total number of records assigned to that code. Used for the Data Cleanse Information Code Summary	Data Cleanse

UDCINFOCODESUMMARY [page 1379]
A Data Cleanse repository table that contains statistics about the information codes that were generated by a specific data cleanse job.

UDCSTATUSCODESUMMARY [page 1380]
A Data Cleanse repository table that contains information about the status codes that were generated from a specific data cleanse job.

Parent topic: Repository tables and related reports [page 1353]

Related Information

Repository tables for USA and Global address cleanse [page 1355]
Repository tables and reports for Geocoder [page 1381]
Match repository statistics tables [page 1383]
Data Cleanse status codes [page 827]
Data Cleanse information codes [page 833]
Data quality statistics tables and supplemental content information [page 1402]

10.2.2.1 UDCINFOCODESUMMARY

A Data Cleanse repository table that contains statistics about the information codes that were generated by a specific data cleanse job.

The software uses the information from this table for the Data Cleanse Information Code Summary report.

Column	Data type definition	Description
INFOCODE	VARCHAR (4)	The information code. Each code begins with a letter that indicates the parser followed by three numbers that represent specific classifications.
INFOCOUNT	INT	The number of records that were assigned this information code.
PARSER	NVARCHAR (100)	The parser used to determine the status. Parsers are: • Custom • Date • Firm • Person • Phone • SSN
PARSE_NUMBER	INT	A number that corresponds to the output field when there is more than one instance. For example Person 1 has a parse number 1, Person 2 has a parse number 2, and so on.
RESERVED_COUNT1-3	INT	Reserved for future use.
RESERVED_STRING1-3	NVARCHAR (255)	Reserved for future use.

> **i Note**
>
> This table also contains the common columns OBJECT_KEY and OBJECT_ID. The primary key is made up of the OBJECT_KEY, OBJECT_ID, PARSER, and the INFOCODE.

Parent topic: Repository tables and reports for Data Cleanse [page 1378]

Related Information

UDCSTATUSCODESUMMARY [page 1380]

Repository tables common columns [page 1352]
Data Cleanse information codes [page 833]
Management Console Guide: Data Quality Reports, Data Cleanse Information Code Summary report

10.2.2.2 UDCSTATUSCODESUMMARY

A Data Cleanse repository table that contains information about the status codes that were generated from a specific data cleanse job.

The software uses the information from this status table for the Data Cleanse Status Code Summary report. Data Cleanse status codes indicate how the data was standardized, and that a specific standard could be used on the data.

Column	Data type definition	Description
STATUSCODE	NVARCHAR (100)	The status code. The status code format is `<parser name>_<output field name>`_STD.
STATUSCOUNT	INT	The number of records that were assigned this status code.
PARSER	NVARCHAR (100)	The parser used to determine the status. Parsers are: • Custom • Date • Firm • Person • Phone • SSN
OUTPUT_FIELD	NVARCHAR (88), NULL	The output field name.
RESERVED_COUNT1-3	INT	Reserved for future use.
RESERVED_STRING1-3	NVARCHAR (255)	Reserved for future use.

> **i Note**
>
> This table also contains the common columns OBJECT_KEY and OBJECT_ID. The primary key is made up of the OBJECT_KEY, OBJECT_ID, and the STATUSCODE.

Parent topic: Repository tables and reports for Data Cleanse [page 1378]

Related Information

UDCINFOCODESUMMARY [page 1379]
Repository tables common columns [page 1352]
Data Cleanse status codes [page 827]
Management Console Guide: Data Quality Reports, Data Cleanse Status Code Summary report

10.2.3 Repository tables and reports for Geocoder

The Geocoder transform generates statistics that the software uses in US postal reporting.

The following table contains a list of repository tables with statistics used for the Geocoder Summary report and the US Addressing report. The sections following this chart contain a topic for each table with descriptions for the fields (columns) in the table.

Repository table name	Description	Transform
GEO_ASSIGN_LEVEL	Contains Geo statistics per assignment level. Used in the Geocoder Summary Report, a subsection of the US Addressing Report.	Geocoder
GEO_INFO_CODE	Contains Geocoder information codes and record counts per code. Used in the US Addressing Report.	Geocoder

GEO_ASSIGN_LEVEL [page 1381]
GEO_INFO_CODE [page 1382]

Parent topic: Repository tables and related reports [page 1353]

Related Information

Repository tables for USA and Global address cleanse [page 1355]
Repository tables and reports for Data Cleanse [page 1378]
Match repository statistics tables [page 1383]
Data quality statistics tables and supplemental content information [page 1402]

10.2.3.1 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.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Parent topic: Repository tables and reports for Geocoder [page 1381]

Related Information

GEO_INFO_CODE [page 1382]
Reference Guide: Geocoder fields, Output fields [page 451]
Repository tables common columns [page 1352]

10.2.3.2 GEO_INFO_CODE

The following table contains statistics about information code found during processing for the Geocoder transform. The Geocoder transform that is used in conjunction with the Global Address Cleanse transform or the USA Regulatory Address Cleanse transform.

Column	Data type definition	Description
INFOCODE	NVARCHAR (4)	A three-character code that provides information about the geocoding results. The status for address and point-of-interest geocoding assignment is indicated in the third character. The status for reverse geocoding assignment is indicated in the second and third characters. If assigned to the best level, the Info_Code field is blank. The first character is not used at this time.
INFO_COUNT	INT	Number of records for the listed INFOCODE.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Parent topic: Repository tables and reports for Geocoder [page 1381]

Related Information

GEO_ASSIGN_LEVEL [page 1381]
Geocoder output fields [page 451]
Repository tables common columns [page 1352]

10.2.4 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.

Table	Description
MTDUPESDATA	Information about match records as a sample of the match results. Related report: Duplicate Sample report.
MTGSSRCBYSRCSTS	Inter and intra-source match counts. Related report: Match Source Statistics Summary report.
MTGSSRCSTS	Information about the distribution of the matches found in various input source records including how the matches were distributed as master records and subordinate records. Related report: Match Source Statistics report.
MTINFO	Name of the match set. Related reports: All of the Match reports except the Best Record Summary report.
MTINSRCBYSRC	Information about how often each input source matched the other input sources.
MTINSRCGRPINFO	Input source group information. The table is populated once per transform if statistics are enabled and the source groups section of the input source object is defined.
MTINSRCINFO	Input source information. The table is populated once per transform if statistics are enabled and the input source object is defined.
MTINSRCMSRC	Multi-source statistics of each input source and each source group. The table is populated once per input source group statistics object if statistics are enabled.
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).

MTBRACTION [page 1385]

MTBRINFO [page 1386]

MTBRKGRP [page 1387]

MTBRKGRPINFO [page 1387]

MTCMPCRIT [page 1388]

MTCRITDEF [page 1389]

MTDUPESDATA [page 1391]

MTGSSRCBYSRCSTS [page 1391]

MTGSSRCSTS [page 1392]

MTINFO [page 1392]

MTINSRCBYSRC [page 1393]

MTINSRCGRPINFO [page 1394]

MTINSRCINFO [page 1394]

MTINSRCMSRC [page 1395]
MTINSRCSELECT [page 1396]
MTINSRCSTATS [page 1398]
MTKEYDEF [page 1399]
MTPROCESS [page 1400]
MTRULESRES [page 1401]

Parent topic: Repository tables and related reports [page 1353]

Related Information

Repository tables for USA and Global address cleanse [page 1355]
Repository tables and reports for Data Cleanse [page 1378]
Repository tables and reports for Geocoder [page 1381]

10.2.4.1 MTBRACTION

This table contains best record information and is applicable to the Match transform. The information is used for the Best Record Summary report.

> **i Note**
>
> This table is populated only if the Best Record functionality is enabled in your job.

Column	Data type definition	Description
PROCID	INT	Sequential number that identifies a match level or an association.
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. **i Note** This column is blank if a source expression is completed instead of a source field.
DSTFLD	NVCHAR (256)	Destination field used in the Best Record Action section.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.2 MTBRINFO

This table contains best record information and is applicable to the Match transform. The information is used for the Best Record Summary report.

> **i Note**
>
> This table is populated only if the Best Record functionality is enabled in your job.

Column	Data Type Definition	Description
PROCID	INT	Sequential number that identifies a match level or an association.
BRNAME	NVARCHAR (255)	Name of the best record object.
POSTDEST	CHAR (1)	Destination for the post. Values are A (all), M (master only), or S (subs only).
POSTOPD	CHAR (1)	More than one posting per destination will be attempted or not for each record (Y/N).
PROTECTDROPS	INT	Number of post operations that were canceled because the destination record was protected.
DSTFLDDROPS	INT	Number of post operations that were canceled because the destination record had already been posted to once, and POSTOPD was set to Y.
FILTERDROPS	INT	Number of post operations that were canceled because the Best Record filter returned F (false).
POSTCOMPLETS	INT	Number of post operations that were successfully completed.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.3 MTBRKGRP

This table contains information about match break groups and is applicable to the Match transform. The information is used in the Match Contribution Report. This table is generated if matching is performed (not associating).

> **i Note**
>
> If breaking is not defined in the job setup, then all records are included in one break group.

Column	Data type definition	Description
NUMRECS	INT	Number of records processed in all of the break groups.
ELAPSEDTIME	INT	Elapsed time to process the break groups (in seconds).
CMPBUFMAXRECS	INT	Maximum number of records that can fit into the compare buffer at one time.
NOBRKCOMPARES	FLOAT	Number of comparisons that would be made without using any break group strategy (or putting all records in a single break group).
BRKGRPCOUNT	INT	Number of break groups formed based on the break group strategy.
BRKGRPLARGEST	INT	Largest break group processed.
BRKGRPCOMPARES	FLOAT	Number of comparisons made in all the break groups.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.4 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.

> **i 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.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.5 MTCMPCRIT

This table contains match level option setting information for each match set and is applicable to the Match transform. This information is in the Match Level Options section of the Match Criteria Summary report.

Column	Data type definition	Description
CMPCRITID	INT	Sequential number that identifies the criteria of a match level.
NAME	NVARCHAR (256)	Match level name.
WTMTSCORE	INT	Minimum weighted match score needed to make a match decision.
NUMNAMEMUSTMT	CHAR (1)	Number of names that must match. Values are O (one name) or A (all names).
CMPFTOMNAME	CHAR (1)	Specifies the setting for the *Compare Given Name1 to Given Name2* option. (Y/N)
MTONHYPLNAME	CHAR (1)	Specifies the setting for the *Match on hyphenated family name* option. (Y/N)
TRNONMAIDENADJ	CHAR (1)	Specifies the setting for the *Ignore family name when female* option. (Y/N)
IGNFIRMIFNAME	CHAR (1)	Specifies the setting for the *Ignore Firm if Name matches* option. (Y/N)
IGNORESTIFBOX	CHAR (1)	Specifies the setting for the *Match on Street and RR, or on Box* option. (Y/N)
ADDRBLMIFFIRM	CHAR (1)	Specifies the setting for the *Address matches blank if Firms match* option. (Y/N)

Column	Data type definition	Description
UNIQRESRRNOBOX	CHAR (1)	Specifies the setting for the *Unique on resident if RR, but no Box* option. (Y/N)

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.6 MTCRITDEF

This table contains match criteria information for each match set and is applicable to the Match transform. The information is used in the Match Input Fields and Detailed Criteria Definition sections of the Match Criteria Summary.

Column	Data type definition	Description
CMPCRITID	INT	Sequential number that identifies the criteria of a match level.
CRITID	INT	Sequential number that identifies an individual criteria.
CRITNAME	NVARCHAR (256)	Name given to the match criteria when it was created.
KEYID	INT	Sequential number that identifies the type of a key field.
MTSCORE	INT	Threshold for similarity scores. Similarity scores at or above this setting are considered a match.
NOMTSCORE	INT	Threshold for similarity scores. Similarity scores at or below this setting are not considered a match.
ONEFLDBLNKOP	CHAR (1)	Setting for the One Field Blank operation. Values are E (evaluate) and I (ignore).
ONEFLDBLNKSCORE	INT	Score given to the criteria when one of the two fields compared is blank.
BTFLDBLNKOP	CHAR (1)	Setting for the Both Fields Blank operation. Values are E (evaluate) and I (ignore).
BTFLDBLNKSCORE	INT	Score given to the criteria when both fields compared are blank.
CNTRBTOWTSCORE	INT	Weight percentage given to the match criteria's contribution to the weighted score.
USEINWTSCOREIFGT	INT	Minimum similarity score needed to qualify the match criteria for use in determining the weighted match score.

Column	Data type definition	Description
ZWTSCOREIFLTOREQ	INT	Minimum similarity score needed for the match criteria to qualify for contributing a value other than zero to the weighted match score.
CMPALGO	CHAR (1)	String comparison algorithm that was used. Values are F for field or W for word.
CHKTRANSPSLET	CHAR (1)	Indicates whether to check for transposed letters (Y/N).
INITADJSCORE	INT	Adjustment score given when fields with whole words match to fields with initials.
SUBSTRADJSCORE	INT	Adjustment score given when fields with longer strings of words match to fields with shorter strings of words (the shorter string must match the first part of the longer string).
APPRSUBADJSCORE	INT	Adjustment score given when fields with longer strings of words do not match to fields with shorter strings of words (the shorter string does not match the first part of the longer string).
ABBRADJSCORE	INT	Adjustment score given to the abbreviation substring adjustment score when the first letter of the shorter word matches the first letter of the longer word, and all remaining letters of the shorter word appear in the longer word in the same order as in the shorter word.
EXTABBRADJSCORE	INT	Adjustment score given when two fields match based on a combination of the abbreviation adjustment score. Keep in mind the following requirements for the extended abbreviation adjustment: • The first letter of the short word must match the first letter of the first word in the multiple-word string. The remaining letters of the short word must be found in order in the multiple-word string. • Letters that match are given a score of 100. The remaining letters are given the score that you specify. • The two scores are proportionally combined to render the overall score.
NUMWDMTEXTOPT	CHAR (1)	Numeric word setting. Values include N (none), A (any position), S (same position), P (any position consider punctuation), or Y (any position ignore punctuation).

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.7 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 Information

Repository tables common columns [page 1352]

10.2.4.8 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 Information

Repository tables common columns [page 1352]

10.2.4.9 MTGSSRCSTS

This table contains information about the distribution of the duplicates found in various input source records and is applicable to the Match transform. It includes information about how the duplicates were distributed as master records and subordinate records. The information is used for the Match Source Statistics Report.

Column	Data type definition	Description
SRCID	INT	Sequential number that identifies a source.
PROCID	INT	Sequential number that identifies a match level or an association.
GSNAME	NVARCHAR (15)	Group statistics name.
SRCNAME	NVARCHAR (256)	Source name.
SSSUB	INT	Single source subordinate record count for this source.
MSSUB	INT	Multiple source subordinate record count for this source.
SSMASTS	INT	Single source master record count for this source.
MSMASTS	INT	Multiple source master record count for this source.
NUMRECS	INT	Record count for this source.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.10 MTINFO

This table contains the match set name and is applicable to the Match transform. The information is used in all of the Match reports except the Best Record Summary report.

Column	Data type definition	Description
MTSET	NVARCHAR (256)	Name of the match set.

> **Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.11 MTINSRCBYSRC

This table contains information about how often each input source matched the other input sources, and these statistics are generated by the Match transform. This table is populated once per input source group statistics object if statistics are enabled. This table is similar to the MTGSSRCBYSRCSTS table. To obtain the numbers for the source groups, the rows with the same INSRCGRPID and the same INSRCGRPID_OTHER must be added.

Column	Data type definition	Description
PROCID	INT	Sequential number used to identify a match level or an association.
INSRCID	INT	Sequential number used to identify an input source.
INSRCID_OTHER	INT	Sequential number used to identify an input source that had one or more records that matched one or more records in INSRCID.
INSRCGRPID	INT	Sequential number used to identify an input source group. A value of zero implies records that do not belong to an input source group.
INSRCGRPID_OTHER	INT	Sequential number used to identify the input source group of INSRCID_OTHER. A value of zero implies that INSRCID_OTHER does not belong to an input source group.
OBJNAME	NVARCHAR (15)	Name of the input source group statistics object.
MATCHES	INT	Sequential number used to identify the input source group of INSRCID. A value of zero indicates that INSRCID does not belong to an input source group.

> **Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.12 MTINSRCGRPINFO

This table contains input source group information and is applicable to the Match transform. It is populated once per transform when statistics are enabled and the source groups section of the input source object is defined.

Column	Data type definition	Description
INSRCGRPID	INT	Sequential number that identifies an input source group.
INSRCGRPNAME	NVARCHAR (255)	Name of an input source group.
MATCHED_COUNT	INT	Number of input records that belong to this input source group because they matched a specified input source.
DEFAULT_COUNT	INT	Number of input records that belong to this input source group because it is the default input source group.

If an input source exists that does not belong to a super source, that input source is assigned a super source ID of zero. A corresponding entry is made into the MTINSRCGRPINFO table in the following fields:

- INSPRSRCID
- INSPRSRCNAME
- MATCHED_COUNT
- DEFAULT_COUNT

All of these fields are set to zero. This allows the report code to successfully perform join operations.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.13 MTINSRCINFO

This table contains input source information and is applicable to the Match transform. It is populated once per transform if statistics are enabled and the input source object is defined.

Column	Data type definition	Description
INSRCID	INT	Sequential number used to identify the input source.
INSRCGRPID	INT	Sequential number used to identify the input source group. The number is zero if this input source does not belong to a source group.
INSRCNAME	NVARCHAR (255)	Name of the input source.
INSRCVALUE	NVARCHAR (255)	Value of the input source.
INSRCTYPE	NVARCHAR (8)	Input source type. Valid values are NORMAL, SUPPRESS, and SPECIAL.
MATCHED_COUNT	INT	Number of input records that belong to this input source because their input source value matched the value of this source.
DEFAULT_COUNT	INT	Number of records that belong to this input source because it is the defaut input source.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.14 MTINSRCMSRC

This table contains the multi-source statistics of each input source and each source group and is applicable to the Match transform. The table is populated once per input source group statistics object if statistics are enabled. To get the numbers for the source groups, the rows with the same INSRCGRPID must be added.

Column	Data type definition	Description
PROCID	INT	Sequential number that identifies a match level or an association.
INSRCID	INT	Sequential number that identifies an input source.
INSRCGRPID	INT	Sequential number that identifies an input source group. When INSRCGRPID is zero, the record does not belong to a source group.
OBJNAME	NVARCHAR (15)	Name of the input source group statistics object.
SRC2	INT	Number of records in INSRCID that are masters in match groups with a source count of 2.

Column	Data type definition	Description
SRC3	INT	Number of records in INSRCID that are masters in match groups with a source count of 3.
SRC4	INT	Number of records in INSRCID that are masters in match groups with a source count of 4.
SRC5	INT	Number of records in INSRCID that are masters in match groups with a source count of 5.
SRC6	INT	Number of records in INSRCID that are masters in match groups with a source count of 6.
SRC7	INT	Number of records in INSRCID that are masters in match groups with a source count of 7.
SRC8	INT	Number of records in INSRCID that are masters in match groups with a source count of 8.
SRC9	INT	Number of records in INSRCID that are masters in match groups with a source count of 9.
SRC10	INT	Number of records in INSRCID that are masters in match groups with a source count of 10.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.15 MTINSRCSELECT

This table contains input source group information and is applicable to the Match transform. This table is populated once per input source select record object if statistics are enabled. To obtain the numbers for the source groups, the rows with the same INSRCGRPID must be added. Refer to the *Reference Guide, Output flag selection options* section for more information.

Column	Data type definition	Description
PROCID	INT	Sequential number that identifies the match level or an association.
INSRCID	INT	Sequential number that identifies an input source.
INSRCGRPID	INT	Sequential number that identifies an input source group. When INSRCGRPID is zero, the record does not belong to a source group.

Column	Data type definition	Description
SECSRCID	INT	Currently not used.
OBJNAME	NVARNVARCHAR (15)	Name of the input source group statistics object.
SECSRCNAME	NVARNVARCHAR (255)	Currently not used.
SSMASTERS_SEL	NVARNVARCHAR (1)	Indicates whether the *Single source masters* option is selected (Y/N).
SSMASTERS_KEEP	INT	Number of single source master records in INSRCID that are flagged for selection. Depending on the input source type, these are either normal or special records.
SSMASTERS_DROP	INT	Number of single source master records in INSRCID that are not flagged for selection. Depending on the input source type, these are either normal or special records.
MSMASTERS_SEL	NVARCHAR (1)	Indicates whether the *Multiple source masters* option is selected (Y/N).
MSMASTERS_KEEP	INT	Number of multi-source master records in INSRCID that are flagged for selection. Depending on the input source type, these are either normal or special records.
MSMASTERS_DROP	INT	Number of multi-source master records in INSRCID that are not flagged for selection. Depending on the input source type, these are either normal or special records.
SSSUBS_SEL	NVARCHAR (1)	Indicates whether the *Single source subordinates* option is selected (Y/N).
SSSUBS_KEEP	INT	Number of single source subordinate records in INSRCID that are flagged for selection. Depending on the input source type, these are either normal or special records.
SSSUBS_DROP	INT	Number of single source subordinate records in INSRCID that are not flagged for selection. Depending on the input source type, these are either normal or special records.
MSSUBS_SEL	NVARCHAR (1)	Indicates whether the *Multiple source subordinates* option is selected (Y/N).
MSSUBS_KEEP	INT	Number of multi-source subordinate records in INSRCID that are flagged for selection. Depending on the input source type, these are either normal or special records.
MSSUBS_DROP	INT	Number of multi-source subordinate records in INSRCID that are not flagged for selection. Depending on the input source type, these are either normal or special records.
SUPSUBS_SEL	NVARCHAR (1)	Indicates whether the *Suppression subordinates* option is selected (Y/N).
SUPSUBS_KEEP	INT	Number of suppress subordinate records in INSRCID that are flagged for selection.
SUPSUBS_DROP	INT	Number of suppress subordinate records in INSRCID that are not flagged for selection.
UNIQUES_SEL	NVARCHAR (1)	Indicates whether the *Unique* option is selected (Y/N).

Column	Data type definition	Description
UNIQUES_KEEP	INT	Number of unique records in INSRCID that are flagged for selection. Depending on the input source type, these are either normal or special records.
UNIQUES_DROP	INT	Number of unique records in INSRCID that are not flagged for selection. Depending on the input source type, these are either normal or special records.
SUPMASTERS_SEL	NVARCHAR (1)	Indicates whether the *Suppression masters* option is selected (Y/N).
SUPMASTERS_KEEP	INT	Number of suppress master records in INSRCID that are flagged for selection.
SUPMASTERS_DROP	INT	Number of suppress master records in INSRCID that are not flagged for selection.
SUPMATCHES_SEL	NVARCHAR (1)	Indicates whether the *Suppression matches* (normal and special records that follow a suppress record in a match group) option is selected (Y/N).
SUPMATCHES_KEEP	INT	Number of suppress match records in INSRCID that are flagged for selection.
SUPMATCHES_DROP	INT	Number of suppress match records in INSRCID that are not flagged for selection.
SUPUNIQUES_SEL	NVARCHAR (1)	Indicates whether the *Suppression uniques* option is selected (Y/N).
SUPUNIQUES_KEEP	INT	Number of unique suppress records in INSRCID that are flagged for selection.
SUPUNIQUES_DROP	INT	Number of unique suppress records in INSRCID that are not flagged for selection.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.16 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.

> **i 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.
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.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.17 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.
DATARECFLDNAME	NVARCHAR (256)	Field's input mapped name.
RMVPUNCT	CHAR (1)	Indicates whether the field was preprocessed by removing punctuation (Y/N).
CONVTOUPPER	CHAR (1)	Indicates whether the field was preprocessed by converting text to upper case (Y/N).
STDDIACHRS	CHAR (1)	Indicates whether the field was preprocessed by converting diacritical characters (Y/N).
CONVTXTTONUM	CHAR (1)	Indicates whether the field was preprocessed by converting numbers represented by text to numerals (Y/N).

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.18 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).

Column	Data type definition	Description
PROCNAME	NVARCHAR (15)	Process name.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.2.4.19 MTRULESRES

This table contains information about the effect of the criteria on the total matching process and is applicable to the Match transform. The information is used in the Match Contribution Report (Criteria Information subreport).

Columns	Data type definition	Description
PROCID	INT	Sequential number that identifies a match level or an association.
CMPCRITID	INT	Sequential number that identifies the criteria of a match level.
CRITID	INT	Sequential number that identifies an individual criteria. A value of 999 is used for the weighted results.
MTDECS	INT	Number of match decisions made by this criteria.
NOMTDECS	INT	Number of no match decisions made by this criteria.
CTDECS	INT	Number of records that were allowed to continue with the match process after a no match decision was made.

> **i Note**
>
> The OBJECT_KEY and OBJECT_ID columns are included in this table but they are not listed. They are primary keys common to most tables.

Related Information

Repository tables common columns [page 1352]

10.3 Data quality statistics tables and supplemental content information

The data quality statistics tables and the supplemental content file enable you to analyze many aspects of the quality of your data.

The data quality statistics tables in the repository contain summary and record-level statistics about the cleansing and assignment processes performed on your data. You use the supplemental content information file (TASK_LOCALIZATION) to translate values that appear in the statistics tables into meaningful information.

You can enable data quality statistics in the Data Cleanse, Geocoder, and Global Address Cleanse transforms. The software sends the statistics to the applicable repository table.

There are two main reasons for generating and using the data quality statistics information:

- To view aggregated results and summary information about the quality of your data.
- To drill into records to gain information about how the software cleansed specific data elements.

Data quality statistics tables

Table name	Transform	Description
CLEANSE_ADDRESS_RECORD_INFO_	Global Address Cleanse	Analyze this non-summary table to gain information about the results of the address cleanse process on the data. The table contains statistics that focus on the results of address cleansing, which includes assignment information and assignment type for each address. This table is populated only when the transform is set for global address cleansing, and only for the address data.
CLEANSE_CHANGE_INFO_	Data Cleanse Global Address	Analyze this non-summary table to identify areas of concern in your source data by the number of significant changes. For example, you may decide to research a data source further if the data shows a high count of significant changes for firm or address components.
CLEANSE_COMPONENT_INFO_ (available after 4.2.6.0 release)	Data Cleanse Global Address Cleanse	This non-summary table contains position information for each specified data element when it is parsed from an input field and/or written to an output field.

Table name	Transform	Description
CLEANSE_INFO_CODES_	Data Cleanse Global Address Cleanse	Analyze this non-summary table to find potential suspect data in your data source. You can analyze the table information to view a distribution of information codes or a count of missing, suspect, or blank data.
CLEANSE_STATISTICS_	Data Cleanse Global Address Cleanse	This summary table contains data for each unique ENTITY_ID for use in providing a high-level insight into what the software changed during the cleansing process.
GEOCODE_INFO_CODES_	Geocoder	Analyze this non-summary table to track the reasons why records were not assigned the highest level of latitude and longitude by using the information code data.
GEOCODE_STATISTICS_	Geocoder	Analyze this summary table to view a summary of the geocode process on the data. The table provides the total number of rows in the data and the total number of rows that were assigned a latitude and longitude.

Content information for data quality statistics tables [page 1404]
You must use the supplemental TASK_LOCALIZATION file to translate values that appear in the statistics tables into meaningful information.

CLEANSE_ADDRESS_RECORD_INFO_ [page 1410]
Provides information about address fields in your data that are changed because of the address cleansing process.

CLEANSE_CHANGE_INFO_ [page 1414]
Provides a non-summary table that includes a row for each significant change identified by the cleanse process.

CLEANSE_COMPONENT_INFO_ [page 1418]
Provides non-summary table that includes information for each specified data element when it is parsed from an input field and/or written to an output field.

CLEANSE_INFO_CODES_ [page 1424]
Provides non-summary table that includes a row for each significant information code generated by the cleanse process.

CLEANSE_STATISTICS_ [page 1428]
Provides a summary of statistics for the cleansing process.

GEOCODE_INFO_CODE_ [page 1432]

Provides information about records in your data that were not assigned the highest point of longitude/latitude during the geocoding process.

GEOCODE_STATISTICS_ [page 1435]
Provides a statistical summary that includes information about the records in your data that were sent through the geocode process.

Parent topic: Data quality reports and statistics tables [page 1351]

Related Information

Repository tables common columns [page 1352]
Repository tables and related reports [page 1353]
CLEANSE_ADDRESS_RECORD_INFO_ [page 1410]
CLEANSE_CHANGE_INFO_ [page 1414]
CLEANSE_COMPONENT_INFO_ [page 1418]
CLEANSE_INFO_CODES_ [page 1424]
CLEANSE_STATISTICS_ [page 1428]
GEOCODE_INFO_CODE_ [page 1432]
GEOCODE_STATISTICS_ [page 1435]

10.3.1 Content information for data quality statistics tables

You must use the supplemental TASK_LOCALIZATION file to translate values that appear in the statistics tables into meaningful information.

You need the task localization file to translate the data into recognizable words and descriptions. Download the localization file named `dqs_task_localization_<language>.csv` from the SAP Community Network. For instructions see “Downloading the data quality statistics task localization file”.

You can customize the translation file to reflect words and descriptions that are more recognizable to your business. After you download the task localization file, you are responsible for storing and preserving the file. If you customize it, you need to transfer those customizations to any future updates.

The following table contains descriptions of the contents of the TASK_LOCALIZATION file. After this table, find the task localization key file for explanations of the characters and codes that may appear in the data quality statistics tables.

TASK_LOCALIZATION file contents

Column name	Data type definition	Description
LOC_TYPE_ID	INT	Type being localized. Use this column to join with the data quality statistics for the description of the LOC_ID.

Column name	Data type definition	Description
LOC_ID	NVARCHAR(255)	ID you want to localize. Use this column to join with the data quality statistics for the description of the ID.
LANGUAGE	NVARCHAR(1)	Identifies which language the information should be translated into. Each of the downloaded files include the language of the translation file. If multiple languages are supported the language key (below) can be used to return the correct translation. For example if you require English, use E, if German use D, and so on. Language key **Key** / **Language** 1 Chinese 3 Korean D German E English F French J Japanese K Danish N Dutch P Portuguese R Russian S Spanish T Turkish
DESCRIPTION	NVARCHAR(255)	Localized value

The following table provides explanations for the characters and codes that may appear in the data quality statistics tables. Use the information as an aid to understand the contents of the various data quality statistics tables.

TASK_LOCALIZATION key

LOC_TYPE_ID	Attribute	Statistics table	Transform	Notes
1	ENTITY_ID	CLEANSE_INFO_CODES_ CLEANSE_CHANGE_INFO_ CLEANSE_STATISTICS_ CLEANSE_COMPONENT_INFO_	Data Cleanse LOC_ID = 2-7 Global Address Cleanse LOC_ID = 1	Identifies which entity type the information pertains to. Not all transform input fields apply to data quality statistics. See "Data quality statistics transform set up requirements" for more information.
2	INFO_CODE	CLEANSE_INFO_CODES_	Data Cleanse Global Address Cleanse	Data Cleanse and Global address cleanse information codes. The code that identifies the rows that may require manual review because the data is suspect. **i Note** Not all existing information codes apply to data quality statistics. See Global Address Cleanse information codes [page 809] for descriptions of Global Address Cleanse information codes. See Data Cleanse information code descriptions [page 834] for descriptions of Data Cleanse information codes.

LOC_TYPE_ID	Attribute	Statistics table	Transform	Notes
3	ISO_COUN-TRY_2CHAR	CLEANSE_AD-DRESS_RE-CORD_INFO_	Global Address Cleanse	Global Address Cleanse ISO two-character country code. See Country ISO codes and assignment engines [page 791] for information about ISO country codes.
4	ASSIGNMENT_TYPE	CLEANSE_AD-DRESS_RE-CORD_INFO_	Global Address Cleanse	Global Address Cleanse output address type. A one- or two-character code that represents the type of address. See Global Address Cleanse output fields [page 554] for information about the Address_Type output field.
5	ASSIGNMENT_INFOR-MATION	CLEANSE_AD-DRESS_RE-CORD_INFO_	Global Address Cleanse	Global Address Cleanse output assignment information. Indicates whether a record is valid, invalid, or corrected based on the status and information codes. See Global Address Cleanse output fields [page 554] for information about the Assignment_Type output field.

LOC_TYPE_ID	Attribute	Statistics table	Transform	Notes
6	ASSIGNMENT_LEVEL	CLEANSE_AD-DRESS_RE-CORD_INFO_	Global Address Cleanse	Global Address Cleanse assignment level. The level to which this transform matched the address to the data in the reference files (directories). See Global Address Cleanse output fields [page 554] for information about the Assignment_Level output field.
7	COMPONENT_ID	CLEANSE_CHANGE_INFO_ CLEANSE_COMPO-NENT_INFO_	Data Cleanse Global Address Cleanse	Identification number for a specific portion of the data entity. For example, an address entity has components of street, street number, city, postcode, and so on.
8	COMPONENT_ELE-MENT_ID	CLEANSE_CHANGE_INFO_ CLEANSE_COMPO-NENT_INFO_	Data Cleanse Global Address Cleanse	Identification number for an element within a COMPONENT_ID. For example, a city component identified by a LOC_TYPE_ID = 7 and LOC_ID of 105 has the following possible elements: • City name = 1 • City description = 2 • City qualifier = 3 • City code = 4 • City addition = 5

LOC_TYPE_ID	Attribute	Statistics table	Transform	Notes
9	CHANGE_CATEGORY_ID	CLEANSE_CHANGE_INFO_	Data Cleanse Global Address Cleanse	Identification number for the type of update performed on the output field: • Change = 12 • Add = 13 • Delete = 14
10	CHANGE_SIGNIFICANCE_ID	CLEANSE_CHANGE_INFO_	Data Cleanse Global Address Cleanse	Identification number for the change significance level.
11	INFO_CODE	GEOCODE_INFO_CODES_	Geocoder	Geocoder information codes. A three-character code that provides information about the geocoding results. **i Note** Not all existing information codes apply to data quality statistics. See Geocoder information codes [page 466] for descriptions of Geocode information codes.
12	REVIEW_GROUP	MATCH_GROUP_INFO_	Match	Not currently available
13	CONFLICT_GROUP	MATCH_GROUP_INFO_	Match	Not currently available
14	DST_ROW_TYPE	BEST_RECORD_RESULTS_	Best Record	Not currently available
15	OPERATION_TYPE	BEST_RECORD_RESULTS_	Best Record	Not currently available

Related Information

Data quality statistics tables and supplemental content information [page 1402]
Downloading the data quality statistics task localization file [page 346]

10.3.2 CLEANSE_ADDRESS_RECORD_INFO_

Provides information about address fields in your data that are changed because of the address cleansing process.

- Contains up to two rows of statistics for each address entity
- Available for Global Address Cleanse transform
- Contains record-level statistics (non-summary)

Analyze this non-summary table to retrieve information about the results of the address cleanse process on the data. The table contains statistics that focus on the results of address cleansing, which includes assignment details for the address.

Download the task localization file, `dqs_task_localization_<language>.csv` from the SAP Community Network website at http://scn.sap.com/docs/DOC-68523 to use for translating values in tables.

Column name	Data type definition	Description
OBJECT_KEY	INT	This unique internal ID is assigned at the time of execution for each of the objects created in Data Services. This is a foreign key to the JOB_KEY in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.
OBJECT_ID	NVARCHAR(255)	This ID is unique for each operation within a Data Services work flow. This is a foreign key to the OBJECT_ID in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.
TABLE_ID	INT	Set to 1.

Column name	Data type definition	Description
ROW_ID	INT	An internal identification generated by the transform that uniquely identifies a row processed by that transform. Appears in each non-summary data quality statistics table and is output by the transform. Use this field to join the non-summary data quality statistics tables to the transform output. Subsequently this can be joined to the input source using the input source primary key if it is included in the transform output.
ENTITY_INSTANCE	INT	A number to describe which entity from the record the entry pertains to. For example the ENTITY_INSTANCE for address can be 1-2, and for person 1-6. The ENTITY_INSTANCE corresponds to the instance of the input mapping of an entity, and is therefore dependent on the order of input column mapping.
ENTITY_INSTANCE_OCCURRENCE	INT	A number to distinguish entities when multiple entities are encountered in the same entity instance. For example this number will distinguish between two person names when a dual name is input in a person column or between two addresses when a dual address is input in a single group of address columns.
DATA_SOURCE	NVARCHAR(255)	The name of the data source that the record comes from. The value is input from the column input mapped to DATA_SOURCE_ID and is simply passed through to the data quality statistics table.
ISO_COUNTRY_2CHAR	VARCHAR(4)	This is the Global Address Cleanse ISO two-character country code. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 3.

Column name	Data type definition	Description
ASSIGNMENT_TYPE	VARCHAR(4)	A one or two character code that represents the type of address. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 4.
ASSIGNMENT_INFORMATION	VARCHAR(4)	Indicates whether a record is valid, invalid, corrected, or blank based on the status and information codes. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 5.
ASSIGNMENT_LEVEL	VARCHAR(4)	The level to which this transform matched the address to the data in the reference files. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 6.

Example

Use the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables to identify the unique key for the results in the data quality statistics tables for the query. The unique key of OBJECT_KEY and OBJECT_ID will be used to analyze the data in the data quality statistic tables. In this example the unique key is:

C.OBJECT_KEY = 412

AND

C.OBJECT_ID = “4f2ba82f-3e7c-44ce-8543-8f5844b44f60”

Sample Code

```
SELECT
    L.DESCRIPTION AS “Country”,
    COUNT(C.ROW_ID) AS “TOTAL”
FROM
    CLEANSE_ADDRESS_RECORD_INFO_ AS C,
    TASK_LOCALIZATION   AS L
WHERE
-- Get the statistics for the desired job and transform.
    C.OBJECT_KEY = 412   AND
    C.OBJECT_ID = “4f2ba82f-3e7c-44ce-8543-8f5844b44f60”  AND
-- Get the ISO_COUNTRY_2CHAR description.
    L.LOC_TYPE_ID = 3 AND
    C.ISO_COUNTRY_2CHAR = L.LOC_ID
GROUP BY
    L.DESCRIPTION
ORDER BY
    TOTAL DESC
```

Results:

Country/Region	Total
Bahamas	642
Italy	323
United States	36
Austria	19
Canada	19
France	16
United Kingdom	15
Germany	11
China	6
Brazil	6
Poland	6
Belgium	5
Puerto Rico	4
Australia	4
Czechia	3
India	2
Netherlands	2
Portugal	2
Republic of Korea	1
Isle of Man	1
Sweden	1
Mexico	1
Japan	1
Switzerland	1
New Zealand	1
Finland	1

10.3.3 CLEANSE_CHANGE_INFO_

Provides a non-summary table that includes a row for each significant change identified by the cleanse process.

- Contains a row of statistics for each entity that was significantly changed during the cleanse process
- Applicable for the Data Cleanse and Global Address Cleanse transforms
- Contains record-level statistics (non-summary)

A significant change is a change that is made during the cleansing process that a user may want to review to ensure the data was cleansed as expected.

Currently the table identifies high and medium significance changes. Medium significance is supported in the Global Address Cleanse transform only.

An example of a high significant change in Global Address Cleanse is when a postcode1 for an address is deleted. An example of a medium significance is when a locality_addition is changed.

An example of a high significant change in Data Cleanse is a change in the last name such as “Sanches” was input and was then output as “Sanchez”.

See the data quality statistics table in the "CLEANSE_STATISTICS_" topic to analyze the total counts for all rows of the input data that were identified as significantly changed.

Analyze this non-summary table to identify concerning areas of your source data by the number of significant changes. For example, you may decide to research a data source further if the data shows a high count of significant changes for firm or address components.

Download the task localization file, `dqs_task_localization_<language>.csv` from the SAP Community Network website at https://wiki.scn.sap.com/wiki/x/ziW7Gg to use for translating values in tables.

Column name	Data type definition	Description
OBJECT_KEY	INT	This unique internal ID is assigned at the time of execution for each of the objects created in Data Services. This is a foreign key to the JOB_KEY in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.
OBJECT_ID	NVARCHAR(255)	This ID is unique for each operation within a Data Services work flow. This is a foreign key to the OBJECT_ID in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.
TABLE_ID	INT	Set to 1.

Column name	Data type definition	Description
ROW_ID	INT	An internal identification generated by the transform that uniquely identifies a row processed by that transform. Appears in each non-summary data quality statistics table and is output by the transform. Use this field to join the non-summary data quality statistics tables to the transform output. Subsequently this can be joined to the input source using the input source primary key if it is included in the transform output.
ENTITY_ID	NVARCHAR(12)	Identification number that refers to entities such as person, firm, and address. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 1.
ENTITY_INSTANCE	INT	A number to describe which entity from the record the entry pertains to. For example the ENTITY_INSTANCE for address can be 1-2, and for person 1-6. The ENTITY_INSTANCE corresponds to the instance of the input mapping of an entity, and is therefore dependent on the order of input column mapping.
ENTITY_INSTANCE_OCCURRENCE	INT	A number to distinguish entities when multiple entities are encountered in the same entity instance. For example this number will distinguish between two person names when a dual name is input in a person column or between two addresses when a dual address is input in a single group of address columns.
DATA_SOURCE	NVARCHAR(255)	The name of the data source that the record comes from. The value is input from the column input mapped to DATA_SOURCE_ID and is simply passed through to the data quality statistics table.

Column name	Data type definition	Description
COMPONENT_ID	NVARCHAR(12)	Identification number that refers to data components. For example, the entity address has components of street, street number, city, and postcode. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 7.
COMPONENT_ELEMENT_ID	NVARCHAR(12)	Identification number that refers to more granular elements within a component. For example, the street component has component elements of street name, street type, street prefix, and street postfix. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 8.
CHANGE_CATEGORY_ID	NVARCHAR(12)	Identification number that refers to the type of change that was made to the data. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 9.
CHANGE_SIGNIFICANCE_ID	NVARCHAR(12)	Identification number that refers to the significance of the change. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 10.

Example

Use the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables to identify the unique key for the results in the data quality statistics tables for the query. The unique key of OBJECT_KEY and OBJECT_ID will be used to analyze the data in the data quality statistic tables. In this example the unique key is:

C.OBJECT_KEY = 412

AND

C.OBJECT_ID = “4f2ba82f-3e7c-44ce-8543-8f5844b44f60”

Sample Code

```
SELECT
    C.COMPONENT_ID AS "Component ID",
    L.DESCRIPTION AS "Component Description",
    COUNT (C.ROW_ID)  AS "Count"
```

```
FROM
    CLEANSE_CHANGE_INFO_  AS C,
    TASK_LOCALIZATION    AS  L
WHERE
-- Get the statistics for the desired job and transform.
    C.OBJECT_KEY = 412   AND
    C.OBJECT_ID = "4f2ba82f-3e7c-44ce-8543-8f5844b44f60" AND
-- Get the COMPONENT_ID description.
    L.LOC_TYPE_ID = 7 AND
    C.COMPONENT_ID = L.LOC_ID
GROUP BY
    C.COMPONENT_ID, L.DESCRIPTION
ORDER BY
    C.COMPONENT_ID;
```

Results:

Component ID	Component Description	Count
101	Country	971
102	Postcode	27
103	Region	7
105	City	35
106	Subcity	3
107	Subcity2	1
109	Delivery Installation	8
110	Street	43
111	Street2	2
114	House Number	2
119	Unit	1
120	Building	2
201	Person	294
202	Job Title	5
301	Organization	114
501	Phone	57
601	E-Mail	291

10.3.4 CLEANSE_COMPONENT_INFO_

Provides non-summary table that includes information for each specified data element when it is parsed from an input field and/or written to an output field.

> **i Note**
>
> The CLEANSE_COMPONENT_INFO_ TABLE is not available for use until after 4.2.6.0 release.

The purpose of this table is to track the exact location of individual data elements that were parsed during the cleansing process.

- Contains a row of statistics for each data element from the data source that was parsed during the cleanse process
- Applicable for the Data Cleanse and Global Address Cleanse transforms
- Contains record-level statistics (non-summary)

Position data includes the input field column start position and the data length for each applicable input component, the output field column start position and the data length for each applicable output component. Note an output column field can be populated when there is no direct link to an input, and an input may not have any directly linked output.

Analyze this non-summary table to identify the location of particular data elements found in your data during the cleansing process, enabling a review user interface to highlight strings when displaying the data before and/or after.

Download the task localization file, `dqs_task_localization_<language>.csv` from the SAP Community Network website at http://scn.sap.com/docs/DOC-68523 to use for translating values in tables.

Column	Data type definition	Description
OBJECT_KEY	INT	This unique internal ID is assigned at the time of execution for each of the objects created in Data Services. This is a foreign key to the JOB_KEY in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.
OBJECT_ID	NVARCHAR(255)	This ID is unique for each operation within a Data Services work flow. This is a foreign key to the OBJECT_ID in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.
TABLE_ID	INT	Set to 1. Join TABLE_ID and COLUMN_ID to the TASK_COLUMN_DEFINITIONS_ table to obtain input field information.

Column	Data type definition	Description
ROW_ID	INT	An internal identification generated by the transform that uniquely identifies a row processed by that transform. Appears in each non-summary data quality statistics table and output by the transform. Use this field to join the non-summary data quality statistics tables to the transform output. Subsequently this can be joined to the input source using the input source primary key if it is included in the transform output.
ENTITY_ID	NVARCHAR(12)	Identification number that refers to entities such as person, firm, and address. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 1.
ENTITY_INSTANCE	INT	A number to describe which entity from the record the entry pertains to. For example the ENTITY_INSTANCE for address can be 1-2, and for person 1-6. The ENTITY_INSTANCE corresponds to the instance of the input mapping of an entity, and is therefore dependent on the order of input column mapping.
ENTITY_INSTANCE_OCCURRENCE	INT	A number to distinguish entities when multiple entities are encountered in the same entity instance. For example this number will distinguish between two person names when a dual name is input in a person column or between two addresses when a dual address is input in a single group of address columns.
DATA_SOURCE	NVARCHAR(255)	The name of the data source that the record comes from. The value is input from the column input mapped to DATA_SOURCE_ID and is simply passed through to the data quality statistics table.

Column	Data type definition	Description
COMPONENT_ID	NVARCHAR(12)	Identification number that refers to data components. For example, the entity address has components of street, street number, city, and postcode. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 7.
COMPONENT_ELEMENT_ID	NVARCHAR(12)	Identification number that refers to more granular elements within a component. For example, the street component has component elements of street name, street type, street prefix, and street postfix. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 8.
COLUMN_ID	INT	The ID of the input table's column that the component element was parsed from. When the component element spans multiple columns in the input table then there are multiple rows for the component element in this table. Join COLUMN_ID and TABLE_ID to the TASK_COLUMN_DEFINITIONS_ table to obtain input field information.
COLUMN_START_POSITION	INT	The character position in the input field where the parsed component element begins.
COLUMN_DATA_LENGTH	INT	The number of characters that was parsed from the input for the component element.
OUTPUT_TABLE_ID	INT	Set to 2. Join OUTPUT_TABLE_ID and OUTPUT_COLUMN_ID to the TASK_COLUMN_DEFINITIONS_ table to obtain output field information.

Column	Data type definition	Description
OUTPUT_COLUMN_ID	INT	The ID of the column that the component element is output to. When the component element exists in multiple output columns then there are multiple rows for the component element in this table. Join OUTPUT_COLUMN_ID and OUTPUT_TABLE_ID to the TASK_COLUMN_DEFINITIONS_ table to obtain output field information.
OUTPUT_COLUMN_START_POSITION	int	The character position in the output field where the output component element begins.
OUTPUT_COLUMN_DATA_LENGTH	INT	The number of characters in the output component element.

Example

Example 1

Use the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables to identify the unique key for the results in the data quality statistics tables for the query. The unique key of OBJECT_KEY and OBJECT_ID will be used to analyze the data in the data quality statistic tables. In this example the unique key is:

C.OBJECT_KEY = 412

AND

C.OBJECT_ID = “4f2ba82f-3e7c-44ce-8543-8f5844b44f60”

Sample Code

```
SELECT
    C.COLUMN_ID AS “Input Column ID”,
    D.COLUMN_NAME  AS “Input Column Name”
FROM
    CLEANSE_COMPONENT_INFO_  AS C,
    TASK_COLUMN_DEFINITIONS_ AS D
WHERE
-- Get the statistics for the desired job and transform.
    C.OBJECT_KEY = 412   AND
    C.OBJECT_ID = “4f2ba82f-3e7c-44ce-8543-8f5844b44f60” AND
-- Get only the rows  where something was parsed but not output.
    C.COLUMN_DATA_LENGTH > 0 AND
    C.OUTPUT_COLUMN_DATA_LENGTH = 0 AND
-- Get the input column name.
    D.OBJECT_KEY = 412   AND
    D.OBJECT_ID = “4f2ba82f-3e7c-44ce-8543-8f5844b44f60” AND
    C.TABLE_ID  = D.TABLE_ID  AND
    C.COLUMN_ID = D.COLUMN_ID
GROUP BY
```

```
    C.COLUMN_ID,
    D.COLUMN_NAME
ORDER BY
    C.COLUMN_ID
```

Results:

Input Column ID	Input Column Name
2	hybrid_thread_test_3000.EMAIL1
3	hybrid_thread_test_3000.EMAILX
4	hybrid_thread_test_3000.PERSON1_FAMILY_NAME1
5	hybrid_thread_test_3000.PERSON1_GIVEN_NAME1
6	hybrid_thread_test_3000.PHONE1
7	hybrid_thread_test_3000.TITLE_LINE1
8	hybrid_thread_test_3000.PHONEX
9	hybrid_thread_test_3000.FIRM_NAME1
10	hybrid_thread_test_3000.FIRM_LINE2

Example

Example 2

Use the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables to identify the unique key for the results in the data quality statistics tables for the query.

The unique key of OBJECT_KEY and OBJECT_ID will be used to analyze the data in the data quality statistic tables. In this example the unique key is:

C.OBJECT_KEY = 412

AND

C.OBJECT_ID = “4f2ba82f-3e7c-44ce-8543-8f5844b44f60”

Sample Code

```
SELECT
    C.ROW_ID,
    D.COLUMN_NAME  AS “Input Column Name”,
    L.DESCRIPTION AS “Component”,
    L2.DESCRIPTION AS “Component Element ID”,
    ABS(C.OUTPUT_COLUMN_DATA_LENGTH - C.COLUMN_DATA_LENGTH) AS “Change in
Length”
FROM
    CLEANSE_COMPONENT_INFO_ AS C,
```

```
    TASK_LOCALIZATION  AS L,
    TASK_LOCALIZATION  AS L2,
    TASK_COLUMN_DEFINITIONS_ AS D
WHERE
-- Get the statistics for the desired job and transform.
    C.OBJECT_KEY = 412 AND
    C.OBJECT_ID = "4f2ba82f-3e7c-44ce-8543-8f5844b44f60" AND
-- Look for changes to length
    C.COLUMN_DATA_LENGTH != C.OUTPUT_COLUMN_DATA_LENGTH AND

-- Get the name of the COMPONENT_ID
    L.LOC_TYPE_ID = 7 AND
    C.COMPONENT_ID = L.LOC_ID AND

-- Get the name of the COMPONENT_ELEMENT_ID
    L2.LOC_TYPE_ID = 8 AND
    C.COMPONENT_ID||'/'||C.COMPONENT_ELEMENT_ID = L2.LOC_ID AND

-- Get the input column name.
    D.OBJECT_KEY = 412   AND
    D.OBJECT_ID = "4f2ba82f-3e7c-44ce-8543-8f5844b44f60" AND
    C.TABLE_ID  = D.TABLE_ID  AND
    C.COLUMN_ID = D.COLUMN_ID
```

Results:

ROW_ID	Input Column Name	Component	Component Element	ID Change in Length
2078	DS_DQMD_AllEngines.Firm	Organization	Organization Name	14
2039	DS_DQMD_AllEngines.Firm	Organization	Organization Name	1
2026	DS_DQMD_AllEngines.Firm	Organization	Organization Name	2

TASK_COLUMN_DEFINITIONS_ [page 1423]
Provides means to retrieve column identification information for the CLEANSE_COMPONENT_INFO_ table.

10.3.4.1 TASK_COLUMN_DEFINITIONS_

Provides means to retrieve column identification information for the CLEANSE_COMPONENT_INFO_ table.

When you generate a CLEANSE_COMPONENT_INFO_ table, the software automatically creates the TASK_COLUMN_DEFINITIONS_ table as a supporting table.

The purpose of this table is to use with the CLEANSE_COMPONENT_INFO_ table to uniquely identify a table input or output, processed by a transform.

Column	Data type definition	Description
OBJECT_KEY	INT	This unique internal ID is assigned at the time of execution for each of the objects created in Data Services. This is a foreign key to the JOB_KEY in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.
OBJECT_ID	NVARCHAR(255)	This ID is unique for each operation within a Data Services work flow. This is a foreign key to the OBJECT_ID in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.
TABLE_ID	INT	Input TABLE_ID =1, output TABLE_ID = 2.
COLUMN_ID	INT	The ID of the input and output table's columns that were used.
COLUMN_NAME	NVARCHAR(128)	The input and output field names.
MAPPED_NAME	NVARCHAR(128)	The mapped name selected when the job was created for the input and output COLUMN_NAME.

See CLEANSE_COMPONENT_INFO_ data quality statistics table for examples of use (CLEANSE_COMPONENT_INFO_ [page 1418])

Parent topic: CLEANSE_COMPONENT_INFO_ [page 1418]

10.3.5 CLEANSE_INFO_CODES_

Provides non-summary table that includes a row for each significant information code generated by the cleanse process.

- Contains a row of statistics for each significant information code that is assigned during the cleanse process.
- Applicable for the Data Cleanse and Global Address Cleanse transforms
- Contains record-level statistics (non-summary)

Analyze this non-summary table to find potential suspect data in your data source. You can use the statistical information to view a distribution of information codes or a count of missing, suspect, or blank data.

Download the task localization file, `dqs_task_localization_<language>.csv` from the SAP Community Network website at http://scn.sap.com/docs/DOC-68523 to use for translating values in tables.

For descriptions of information codes see Global Address Cleanse information codes [page 809] and Data Cleanse information codes [page 833]

> **i Note**
>
> Not all existing information codes apply to data quality statistics.

Column	Data type definition	Description
OBJECT_KEY	INT	This unique internal ID is assigned at the time of execution for each of the objects created in Data Services. This is a foreign key to the JOB_KEY in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.
OBJECT_ID	NVARCHAR (255)	This ID is unique for each operation within a Data Services work flow. This is a foreign key to the OBJECT_ID in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.
TABLE_ID	INT	Set to 1.
ROW_ID	INT	An internal identification generated by the transform that uniquely identifies a row processed by that transform. Appears in each non-summary data quality statistics table and output by the transform. Use this field to join the non-summary data quality statistics tables to the transform output. Subsequently this can be joined to the input source using the input source primary key if it is included in the transform output.
ENTITY_ID	NVARCHAR (12)	Identification number that refers to entities such as person, firm, and address. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 1.
ENTITY_INSTANCE	INT	A number to describe which entity from the record the entry pertains to. For example the ENTITY_INSTANCE for address can be 1-2, and for person 1-6. The ENTITY_INSTANCE corresponds to the instance of the input mapping of an entity, and is therefore dependent on the order of input column mapping.

Column	Data type definition	Description
ENTITY_INSTANCE_OCCURRENCE	INT	A number to distinguish entities when multiple entities are encountered in the same entity instance. For example this number will distinguish between two person names when a dual name is input in a person column or between two addresses when a dual address is input in a single group of address columns.
DATA_SOURCE	NVARCHAR (255)	The name of the data source that the record comes from. The value is input from the column input mapped to DATA_SOURCE_ID and is simply passed through to the data quality statistics table.
INFO_CODE	NVARCHAR(10)	Indicates whether something potentially suspect about the data was encountered during the processing of the cleanse operation. Only records that get an information code are written to this table. Records that get multiple information codes will have multiple rows written to this table, one per information code. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 2.

Example

Use the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables to identify the unique key for the results in the data quality statistics tables for the query. The unique key of OBJECT_KEY and OBJECT_ID will be used to analyze the data in the data quality statistic tables. In this example this unique key is:

C.OBJECT_KEY = 412

AND

C.OBJECT_ID = “4f2ba82f-3e7c-44ce-8543-8f5844b44f60”

Sample Code

```
SELECT
    C.INFO_CODE AS “Info Code”,
    L.DESCRIPTION AS “Info Code Description”,
    COUNT (C.ROW_ID)  AS “Count”
FROM
    CLEANSE_INFO_CODES_  AS C,
    TASK_LOCALIZATION        AS L
WHERE
```

```
-- Get the statistics for the desired job and transform.
    C.OBJECT_KEY = 412   AND
    C.OBJECT_ID = "4f2ba82f-3e7c-44ce-8543-8f5844b44f60" AND
-- Get the description for the INFO_CODE
    L.LOC_TYPE_ID = 2 AND
    C.INFO_CODE = L.LOC_ID
GROUP BY
    C.INFO_CODE, L.DESCRIPTION
ORDER BY
    C.INFO_CODE;
```

Results:

Info Code	Info Code Description	Count
I161	No entry made for job title	2335
I171	Non-title data found together with the job title	339
I221	No entry made for person name	2365
I361	No entry made for organization name	2943
I362	No entry made for organization name	2943
I711	This is not an e-mail address	344
I722	No entry made for e-mail address	2943
I731	Non e-mail data found together with the e-mail address	36
I732	Non e-mail data found together with the e-mail address	57
I751	This is not a phone number	72
I761	No entry made for phone number	13
1030	No country identified	1060
2000	Unable to identify city, region, and/or postcode information	2
2020	Unable to identify postcode. Invalid city is preventing address cleansing	965
3000	City, region, and postcode are valid, but unable to identify the street address	2
3010	City, region, and postcode are valid, but unable to match street name to directory	10

Info Code	Info Code Description	Count
3030	House number has not been entered or is not in the directory	12
3090	An invalid or missing city is preventing address cleansing	1
3200	The building has not been entered or is not in the directory	2
4000	The secondary address information has not been entered or is not in the directory	8
4500	The organization has not been entered or is not in the directory	5
5000	The address is valid, but the postal authority classifies this address as undeliverable	2

10.3.6 CLEANSE_STATISTICS_

Provides a summary of statistics for the cleansing process.

The table contains summarized data for each unique entity ID for use in providing a high-level insight into what the software changed during the cleansing process. You can use this summary table to ensure that your data was cleansed as you expected.

Information in the table

Suspect Records:

For Global Address Cleanse, this is a count of records whose information code is less than 4000, plus 5000, 5010, 5040, and 5050. Users can locate these records by looking at the information code.

A user can locate these records by using the data quality statistics table CLEANSE_INFO_CODES_ (CLEANSE_INFO_CODES_ [page 1424]) to query for the information codes.

See Global Address Cleanse information codes [page 809] for descriptions of Global Address Cleanse information codes.

See Data Cleanse information codes [page 833] for descriptions of Data Cleanse information codes.

Blanks Records

For Global Address Cleanse, this is a count of records whose assignment information is set to “B”. Users can locate these records by using the data quality statistics table CLEANSE_ADDRESS_RECORD_INFO_ to query for the rows where ASSIGNMENT_INFORMATION LIKE “B”

A user can locate these records by using the data quality statistics table CLEANSE_INFO_CODES_ (CLEANSE_INFO_CODES_ [page 1424]) to query for the above information codes.

Significant Changes Records

A significant change is a change that was made during the cleansing process that a user may want to review to ensure the data was cleansed as expected.

Currently high significance changes are identified. An example of a high significant change is when a postcode1 for an address is deleted, or a change in the last name such as “Sanches” was input and output as “Sanchez”.

See the data quality statistics table, CLEANSE_CHANGE_INFO_, (CLEANSE_CHANGE_INFO_ [page 1414]) to identify all rows of the input data that were identified as significantly changed.

The reason for the change during the cleanse process is taken into consideration when determining the significance. For example:

- **Reference data:** this includes Global Address Cleanse and Geocoder reference data, and Data Cleanse cleansing package. An example of this reason is the correction of a misspelled city name.
- **Derived:** some elements may be added because they can be derived from the attributes of other elements. For example if the first name is John, a strong given name, a prename of Mr. can be derived.
- **Insufficient confidence:** when there is insufficient confidence in identifying an entity the entire contents may be deleted on output. For example if the input includes an address and the country of the address cannot be confidently identified the address may be deleted on output.

> **i Note**
>
> If an entity instance received multiple significant changes, it is counted only once.

Download the task localization file, `dqs_task_localization_<language>.csv` from the SAP Community Network website at http://scn.sap.com/docs/DOC-68523 to use for translating values in tables.

Column	Data type definition	Description
OBJECT_KEY	INT	This unique internal ID is assigned at the time of execution for each of the objects created in Data Services. This is a foreign key to the JOB_KEY in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.

Column	Data type definition	Description
OBJECT_ID	NVARCHAR(255)	This ID is unique for each operation within a Data Services work flow. This is a foreign key to the OBJECT_ID in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.
ENTITY_ID	NVARCHAR(12)	Identification number that refers to entities such as person, firm, and address. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 1.
ENTITY_INSTANCE	INT	A number to describe which entity from the record the entry pertains to. For example the ENTITY_INSTANCE for address can be 1-2, and for person 1-6. The ENTITY_INSTANCE corresponds to the instance of the input mapping of an entity, and is therefore dependent on the order of input column mapping.
NUM_RECORDS	INT	This is the count of records input to the cleanse process.
NUM_SUSPECTS	INT	This is the count of entities that have had a change in the data selected to be output that is suspect and it is recommended to review the cleansed output to ensure it is as expected. To identify the suspects the non-summary data quality statistics tables can be used.
NUM_BLANKS	INT	This is the count of entities identified as blank during the cleanse operation. To identify the blanks the non-summary data quality statistics tables can be used.

Column	Data type definition	Description
NUM_HIGH_SIGNIF-ICANT_CHANGES	INT	This is the count of entities identified as highly significant. To identify the significant the non-summary data quality statistics tables can be used. Note: **i Note** If an entity instance received multiple significant changes, it is counted only once.

Example

Use the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables to identify the unique key for the results in the data quality statistics tables for the query. The unique key of OBJECT_KEY and OBJECT_ID will be used to analyze the data in the data quality statistic tables. In this example the unique key is:

C.OBJECT_KEY = 412

AND

C.OBJECT_ID = “4f2ba82f-3e7c-44ce-8543-8f5844b44f60”

Sample Code

```
SELECT
    L.DESCRIPTION AS “Entity Cleansed”,
    C.ENTITY_INSTANCE “AS Entity Instance Cleansed”,
    C.NUM_RECORDS AS “Total Number of Rows”,
    C.NUM_SUSPECTS AS “Number of Suspects”,
    C.NUM_BLANKS AS “Number of Blanks”,
    C.NUM_HIGH_SIGNIFICANT_CHANGES AS “Number of Significant Changes”
FROM
    CLEANSE_STATISTICS_ AS C,
    TASK_LOCALIZATION    AS L
WHERE
-- Get the statistics for the desired job and transform.
    C.OBJECT_KEY = 412   AND
    C.OBJECT_ID = “4f2ba82f-3e7c-44ce-8543-8f5844b44f60” AND
-- Get the description for the ENTITY_ID
    L.LOC_TYPE_ID = 1 AND
    C.ENTITY_ID = L.LOC_ID
ORDER BY
    C.ENTITY_ID,
    C.ENTITY_INSTANCE;
```

RESULTS:

Entity Cleansed	Entity Instance Cleansed	Total Number of Rows	Number of Suspects	Number of Blanks	Number of Significant Changes
Address	1	3000	2021	36	132
Person	1	3000	625	236	296
Organization	1	3000	1560	256	57
Organization	2	3000	152	2205	23
Phone	1	3000	85	13	2
Phone	2	3000	569	1250	22
E-Mail	1	3000	380	22	234
E-Mail	2	3000	546	1552	59

10.3.7 GEOCODE_INFO_CODE_

Provides information about records in your data that were not assigned the highest point of longitude/latitude during the geocoding process.

- Contains a row of statistics for each applicable information code that is assigned during the cleanse process.
- Applicable for Geocoder Cleanse transform
- Contains record-level statistics (non-summary)

Analyze this non-summary table to track the reasons why records were not assigned the highest level of latitude and longitude by using the information code data.

Download the task localization file, `dqs_task_localization_<language>.csv` from the SAP Community Network website at http://scn.sap.com/docs/DOC-68523 to use for translating values in tables.

For descriptions of information codes see Data Cleanse information codes [page 833].

> **i Note**
>
> Not all existing information codes apply to data quality statistics.

Column name	Data type definition	Description
OBJECT_KEY	INT	This unique internal ID is assigned at the time of execution for each of the objects created in Data Services. This is a foreign key to the JOB_KEY in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.
OBJECT_ID	NVARCHAR(255)	This ID is unique for each operation within a Data Services work flow. This is a foreign key to the OBJECT_ID in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.
TABLE_ID	INT	Set to 1.
ROW_ID	INT	An internal identification generated by the transform that uniquely identifies a row processed by that transform. Appears in each non-summary data quality statistics table and output by the transform. Use this field to join the non-summary data quality statistics tables to the transform output. Subsequently this can be joined to the input source using the input source primary key if it is included in the transform output.
DATA_SOURCE	NVARCHAR(255)	The name of the data source that the record comes from. The value is input from the column input mapped to DATA_SOURCE_ID and is simply passed through to the data quality statistics table.
INFO_CODE	VARCHAR(10)	Indicates whether something occurred that prevented the geocode operation from assigning the highest level of latitude/longitude point to the address. Only records that get an information code are written to this table. This is a foreign key to the LOC_ID column in the TASK_LOCALIZATION table where the LOC_TYPE_ID = 11.

Example

Use the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables to identify the unique key for the results in the data quality statistics tables for the query. The unique key of OBJECT_KEY and OBJECT_ID will be used to analyze the data in the data quality statistic tables. In this example the unique key is:

C.OBJECT_KEY = 412

AND

C.OBJECT_ID = “4f2ba82f-3e7c-44ce-8543-8f5844b44f60”

Sample Code

```
SELECT
    C.INFO_CODE AS “Info Code”,
    L.DESCRIPTION AS “Info Code Description”,
    COUNT (C.ROW_ID)  AS “Count”
FROM
    GEOCODE_INFO_CODES_   AS C,
    TASK_LOCALIZATION     AS L
WHERE
-- Get the statistics for the desired job and transform.
    C.OBJECT_KEY = 412 AND
    C.OBJECT_ID =
“4f2ba82f-3e7c-44ce-8543-8f5844b44f60”
         AND
-- Get the description for the INFO_CODE.
    L.LOC_TYPE_ID = 11 AND
    C.INFO_CODE = L.LOC_ID
GROUP BY
    C.INFO_CODE, L.DESCRIPTION
ORDER BY
    C.INFO_CODE;
```

Results:

Info Code	Info Code Description	Count
001	Geocode reference data is not available for this country	64
005	The address entered does not match the Geocode reference data	320
006	The address entered matches multiple addresses in the Geocode reference data and is thus ambiguous	192
008	No entry made for address. This is required in order to match the Geocode reference data	1216

10.3.8 GEOCODE_STATISTICS_

Provides a statistical summary that includes information about the records in your data that were sent through the geocode process.

- Contains a summary row of statistics for the number of total rows and number of rows assigned a longitude/latitude during the geocode process
- Applicable for the Geocoder transform

The table contains summarized data for each Geocode transform to use in providing a high-level insight into the assignment ratio that was output from the geocode process.

Analyze this summary table to identity the assignment success for the highest level of latitude and longitude.

Download the task localization file, `dqs_task_localization_<language>.csv` from the SAP Community Network website at http://scn.sap.com/docs/DOC-68523 to use for translating values in tables.

Column name	Data type definition	Description
OBJECT_KEY	INT	This unique internal ID is assigned at the time of execution for each of the objects created in Data Services. This is a foreign key to the JOB_KEY in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.
OBJECT_ID	NVARCHAR(255)	This ID is unique for each operation within a Data Services work flow. This is a foreign key to the OBJECT_ID in the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables.
NUM_RECORDS	INT	This is the count of records input to the cleanse process.
NUM_ASSIGNED	INT	This is the count of records that receive latitude/longitude coordinates. The assignment level is not considered, only that a latitude and longitude is generated.

Example

Use the DQVW_AGR_RPTSTATS and DQVW_REPORTS_STAT tables to identify the unique key for the results in the data quality statistics tables for the query. The unique key of OBJECT_KEY and OBJECT_ID will be used to analyze the data in the data quality statistic tables. In this example the unique key is:

C.OBJECT_KEY = 412

AND

C.OBJECT_ID = “4f2ba82f-3e7c-44ce-8543-8f5844b44f60”

Sample Code

```
SELECT
    C.NUM_RECORDS AS “Total Number of Rows”,
    C.NUM_ASSIGNED AS “Number of Rows Assigned”
FROM
    GEOCODE_STATISTICS_  AS C
WHERE
-- Get the statistics for the desired job and transform.
    C.OBJECT_KEY = 412   AND
    C.OBJECT_ID =  '4b5cf77c-b4a1-479f-acf4-e03b64594a70'
ORDER BY
    C.OBJECT_ID;
```

Results:

Total Number of Rows	Number of Rows Assigned
3392	2816

11 Locales and multi-byte functionality

SAP Data Services supports the use of different locales in sources, the Job Server, and targets.

Data Services 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.

11.1 Terminology for locales and multibyte support

Become familiar with the terminology we use when we talk about locales and multibyte support.

The following table contains terms that we use in SAP Data Services regarding locales and multibyte support.

Term	Definition
Locale	Consists of three values related to world regions that control the format of data when it's 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 don't require that you set the Territory value for a locale. Database sources and targets might not need locale values specified.
Language	Specifies the locale value for a human language.
Territory	Specifies the locale value for a geographical location (usually the country/region) where a locale language is used. The paring of a language with a territory determines factors such as date format, time format, decimal separator, currency format, and so on. Data Services uses territory values to process the following data types: • date • datetime • decimal • double • int • interval • numeric • real • time • timestamp
Code page	A table of characters that associates each character with a numeric index (code point value). Data Services uses a code page value to transcode varchar data types. Most languages require their own code pages, although some code pages can represent multiple languages. Most code pages are compatible with `US-ASCII` for code points below 128. Multilanguage code pages mean, 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 doesn't validate that they are realistic.
Single-byte	An encoding or code page in which each character is represented by one byte.

Term	Definition
Multibyte	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 multibyte characters. Use multibyte code pages to process characters for those languages.
Encoding	The process of representing a code page character as one byte (single-byte encoding) or a sequence of bytes (multi-byte encoding).
Transcode	Converts data from one code page to another. To support ETL environments in which sources with different locales are processed in the same job, Data Services supports transcoding. Note that transcoding can impact performance.
Unicode	Provides a unique number for every character and a method to implement ISO/ISE 10646. Data Services supports UTF-8 and UTF-16 Unicode transformation formats.
UTF-8	Serializes a Unicode code point as a sequence of one to four bytes depending on the complexity of the character (single-byte characters use one byte and multibyte characters use up to four). Data Services allows you to select UTF-8 as a code page for the Job Server and connections to sources and targets.

UTF-16	In Data Services, standardizes each Unicode code point at two bytes for each character. Allows access to 63k characters as 16-bit units.
	
	Data Services supports UTF-16 for:
	
	• A Microsoft SQL Server database when its datastore code page is set to `utf8` or `utf16`
	• Data types for national character-sets 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 don't have to set locales for connections to database sources and targets.

Parent topic: Locales and multi-byte functionality [page 1437]

Related Information

Locales and language packs [page 1440]

11.2 Locales and language packs

Language packs contain locales for viewing the SAP Data Services user interface and other generated texts in languages other than English.

Language packs, other than English, which is the default language, are available for installation. However, you don't have to reinstall Data Services to obtain a language pack. You can install language packs over an existing Data Services installation.

After a language pack is installed, you can select the locale for the user interface and other generated text such as displayed data.

To set the locale, complete the following options:

- *Product locale:* Specifies the user interface language and all product messages.
- *Preferred viewing locale:* Specifies the locale in which to present the user data. For example, specify that the date formatting is presented in the preferred viewing locale.

Set the options in one of two locations. The location is based on your installation configuration:

- In Designer: *Tools* > *Options*
- In the Locale Selector utility

  Setting locales in Designer [page 1441]
  : If you selected to include SAP Data Services Designer when you installed Data Services, set locals in Designer.

  Setting locales in the Locale Selector [page 1442]
  : If you selected to install the *SAP Data Services* engine, but not *SAP Data Services Designer* when you installed Data Services, set locales using the SAP Data Services Locale Selector.

  Setting locales in UNIX or Linux [page 1443]
  : To modify locales for UNIX or Linux operating systems, edit the DSConfig.txt file.

  Impact of locale settings on Data Services components [page 1443]
  : Settings that you make in SAP Data Services Designer or the Locale Selector have a different impact on various Data Services components.

Parent topic: Locales and multi-byte functionality [page 1437]

Related Information

Terminology for locales and multibyte support [page 1437]
Locale support [page 1444]
Transcoding and code page support [page 1448]
Example locale settings [page 1455]
Multibyte and Unicode support [page 1457]
Supported locales and encodings [page 1463]
Setting locales in Designer [page 1441]
Setting locales in the Locale Selector [page 1442]

11.2.1 Setting locales in Designer

If you selected to include SAP Data Services Designer when you installed Data Services, set locals in Designer.

> **i Note**
>
> When you change the locale settings in Designer, Data Services changes the locale in the Locale Selector automatically.

To set the locale in Designer, log into Designer and perform the following steps:

1. Choose *Tools* > *Options*.
   The *Options* dialog box opens.
2. Expand the *Designer* node, and select *Language*.

   Options appear at right.
3. Select a value for the *Product Locale* and the *Preferred Viewing Locale* options.

   The default setting is English.
4. Select *OK*.

Task overview: Locales and language packs [page 1440]

Related Information

Setting locales in the Locale Selector [page 1442]
Setting locales in UNIX or Linux [page 1443]
Impact of locale settings on Data Services components [page 1443]

11.2.2 Setting locales in the Locale Selector

If you selected to install the *SAP Data Services* engine, but not *SAP Data Services Designer* when you installed Data Services, set locales using the SAP Data Services Locale Selector.

> **i Note**
>
> When you change the locale settings in the Locale Selector, Data Services changes the locale settings in the Designer *Options* dialog box automatically. This note is only relevent if you installed Designer.

To set a locale in the Locale Selector, perform the following steps:

1. Access the Locale Selector application based on your operating system:
   - **Windows:** From the Start menu, start to type "Locale Selector". Windows presents the best match, which is **Data Services Locale Selector**.
   - *UNIX/Linux:* Open a command line and type: `./start LocaleSelector.sh`

   The Locale Selector application opens.
2. Select a locale from the Viewing Locale dropdown list.

   The options for *Product Locale* and *Server Log Locale* are set to English and can't be changed.
3. Specify the Language, territory, and code page to use for the repository connection and for processing data:
   - *Use default database locale*: Uses the default locale for all the language, territory and code page.
   - *Use the following locale*: Enables *Language and Territory* and *Code Page* options. Select a language for each option, or leave them set to `<default>`.

     > **i Note**
     >
     > UTF-16 is supported as a code page but can't be selected in the *Locale Selector code page* dropdown list.
4. Select *OK* to save your settings and close the application.

Task overview: Locales and language packs [page 1440]

Related Information

Setting locales in Designer [page 1441]
Setting locales in UNIX or Linux [page 1443]
Impact of locale settings on Data Services components [page 1443]

11.2.3 Setting locales in UNIX or Linux

To modify locales for UNIX or Linux operating systems, edit the DSConfig.txt file.

To modify the product locales on a UNIX or Linux system, perform the following steps:

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 applicable locale.
3. Save and close the `DSConfig.txt` file.

Task overview: Locales and language packs [page 1440]

Related Information

Setting locales in Designer [page 1441]
Setting locales in the Locale Selector [page 1442]
Impact of locale settings on Data Services components [page 1443]

11.2.4 Impact of locale settings on Data Services components

Settings that you make in SAP Data Services Designer or the Locale Selector have a different impact on various Data Services components.

The following table lists components affected by your locale settings and how they're affected.

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 displays in the same language as the Designer's *Options* window *Product locale* setting.
Log files	Messages written to a log file are in the language set in the *Server Logs locale* option in the Locale Selector. Locale settings affect only the text of the log files. The date and timestamps remain the same.

Parent topic: Locales and language packs [page 1440]

Related Information

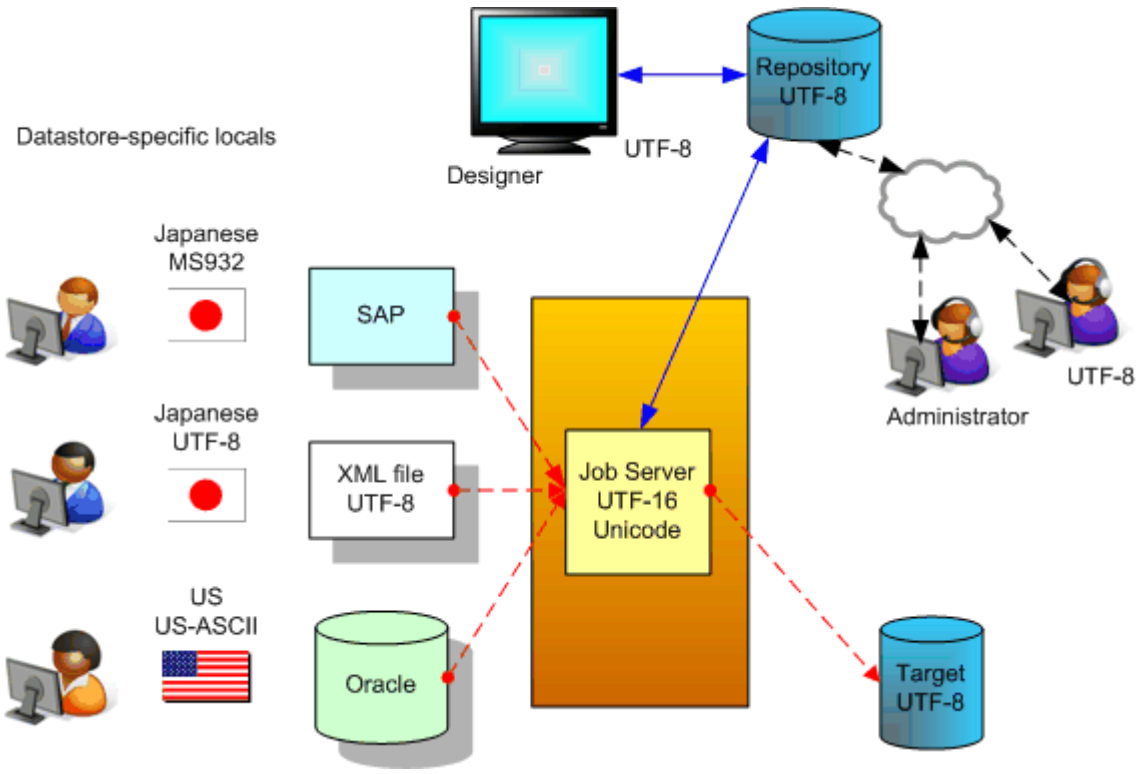
Setting locales in Designer [page 1441]
Setting locales in the Locale Selector [page 1442]
Setting locales in UNIX or Linux [page 1443]

11.3 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 the locals that Data Services supports, set locales for each of the following areas:

- Database (source, target, and repository) or application (SAP, PeopleSoft, Oracle, Siebel, JDE).
- Database client.
- Datastore connection to a database or application.
  Datastore locales must match the locales of source and target database clients. This allows the datastore to move data between Data Services and each database without possible data corruption. If the database locale differs from its database client locale, it is your responsibility to ensure that the database transcodes the data before it reaches or after it leaves Data Services.

  > **i Note**
  >
  > To avoid the necessity of setting locales for a database client and a Data Services datastore, you can process with or without UTF-16 Unicode.

  > **i Note**
  >
  > All adapter datastores are automatically set by Data Services to the code page UTF-8. They are handled the same way as XML message sources and targets.

- File format (flat file, XML Schema, and DTD): Match the file format locale to that of each source or target file.
- Job Server: Data Services uses the Job Server's locale for the engines it spawns. The Designer also uses the Job Server's locale as its repository connection locale. To avoid possible data corruption, in case the SAP Sybase repository or DB2 repository database codepage is not UTF-8, use the same locale settings for the repository, its client, and the Job Server.

  Locale selection [page 1445]

Parent topic: Locales and multi-byte functionality [page 1437]

Related Information

Terminology for locales and multibyte support [page 1437]
Locales and language packs [page 1440]
Transcoding and code page support [page 1448]
Example locale settings [page 1455]
Multibyte and Unicode support [page 1457]
Supported locales and encodings [page 1463]
Processing with and without UTF-16 Unicode [page 1449]
Guidelines for file format locales [page 1452]

11.3.1 Locale selection

Data Services automatically sets the locale for the Job Server, Designer, and Management console.

Job Server locale [page 1446]

Overriding the default Job Server locale [page 1447]

Designer locale [page 1447]

Management Console locale [page 1448]

Parent topic: Locale support [page 1444]

11.3.1.1 Job Server locale

The Job Server locale is set after installation to `<default>`. This default Job Server locale 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 locale 1	Datastore locale 2	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
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 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.

Datastore locales 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

Parent topic: Locale selection [page 1445]

Related Information

Overriding the default Job Server locale [page 1447]
Designer locale [page 1447]
Management Console locale [page 1448]
Locales and multi-byte functionality [page 1437]
Locale selection [page 1445]

11.3.1.2 Overriding the default Job Server locale

You can override the default locale for the Job Server by using the Data Services Locale Selector utility.

Choose *Start* > *Programs* > *SAP Data Services* `<x.x>` > *Data Services Locale Selector*.

For more information, see “Guidelines for setting locales”.

Task overview: Locale selection [page 1445]

Related Information

Job Server locale [page 1446]
Designer locale [page 1447]
Management Console locale [page 1448]

11.3.1.3 Designer locale

The Designer uses the Job Server locale to move data between the Designer and the repository.

The Designer expects to receive data from the repository in the Job Server's locale. Objects you create in the Designer are represented internally using a textural language (ATL) that is sent to and received from the repository's database client in the form of SQL statements. Use the same locale when installing the Job Server as you set for your repository and its client to support Data Services' internal language.

The Designer also has its own locale which is automatically set to that of its Microsoft Windows operating system locale. The Designer automatically transcodes input data from its locale to the Job Server locale when it interacts with the repository.

Parent topic: Locale selection [page 1445]

Related Information

Job Server locale [page 1446]
Overriding the default Job Server locale [page 1447]
Management Console locale [page 1448]
Locales and multi-byte functionality [page 1437]
Locale selection [page 1445]

11.3.1.4 Management Console locale

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.

Parent topic: Locale selection [page 1445]

Related Information

Job Server locale [page 1446]
Overriding the default Job Server locale [page 1447]
Designer locale [page 1447]
Locales and multi-byte functionality [page 1437]
Locale selection [page 1445]

11.4 Transcoding and code page support

SAP Data Services transcodes between different code pages automatically, when necessary, to support complex, multilanguage data management in a single job.

The code page that you select for each locale used by a job determines whether Data Services transcodes during job processing. Mismatched locale settings between Data Services and configured connections cause data corruption. Therefore, take care when you set connections between Data Services and its repository, sources, and targets. Also make sure that you use the same code page as the database middleware to which you connect.

Processing with and without UTF-16 Unicode [page 1449]

Minimize transcoding [page 1451]

Because transcoding impacts job performance, SAP Data Services takes advantage of minimizing the impact of transcoding equivalent code sets.

Parent topic: Locales and multi-byte functionality [page 1437]

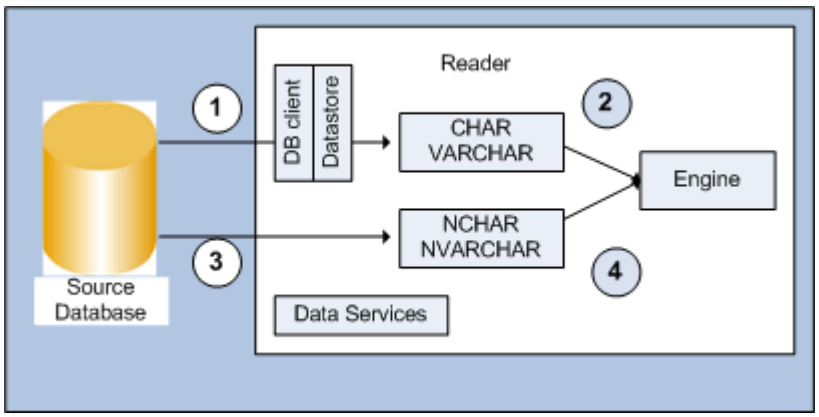
Related Information

Terminology for locales and multibyte support [page 1437]
Locales and language packs [page 1440]
Locale support [page 1444]
Example locale settings [page 1455]
Multibyte and Unicode support [page 1457]
Supported locales and encodings [page 1463]
Transcoding and round-trip conversion conflicts [page 1461]

11.4.1 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.

Parent topic: Transcoding and code page support [page 1448]

Related Information

Minimize transcoding [page 1451]
Column sizing and multibyte support [page 1462]

11.4.2 Minimize transcoding

Because transcoding impacts job performance, SAP Data Services takes advantage of minimizing the impact of transcoding equivalent code sets.

If a datastore or file format and the Job Server use a different locale, Data Services automatically transcodes the data, which supports a multilanguage enterprise environment. However, transcoding impacts job performance. To lessen the impact of transcoding, take the following actions:

- Use the same locale for all components.
- Use a single-byte code page if possible.

Data Services minimizes the impact of transcoding for equivalent code sets such that transcoding between the following code page pairs doesn't impact performance.

Superset	Subset
cp1252	ISO88591 (LATIN1)
cp1250	ISO88592
cp1251	ISO88595
cp1253	ISO88597
cp1254	ISO88599
cp1255	ISO88598
cp1256	ISO88596
cp1257	ISO88594

Parent topic: Transcoding and code page support [page 1448]

Related Information

Processing with and without UTF-16 Unicode [page 1449]
Supported locales and encodings [page 1463]

11.5 Guidelines for setting locales

11.5.1 Guidelines for Job Server locale

The repository connection and spawned engines use the Job Server locale settings.

Before you set the Job Server locale, read about specific guidelines:

- If you set the Job Server locale to `default` after installation, the Job Server takes its locale from the operating system of the host computer.
- If you process multibyte data with the operating system locale set to single-byte, set the Job Server locale to the same code page or a superset of the data code page to avoid data corruption.
- If you run your jobs in a multilanguage environment, SAP Data Services automatically sets the Job Server locale to a superset of all datastore and file format locales.
- If you don't have a multilanguage 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.
- The repository database client, which is on the same computer as Data Services Designer, must match the Job Server locale. The Designer uses the Job Server locale to ensure that it passes accurate data to the repository.

Override the default locale for the Job Server by using the Data Services Locale Selector.

Related Information

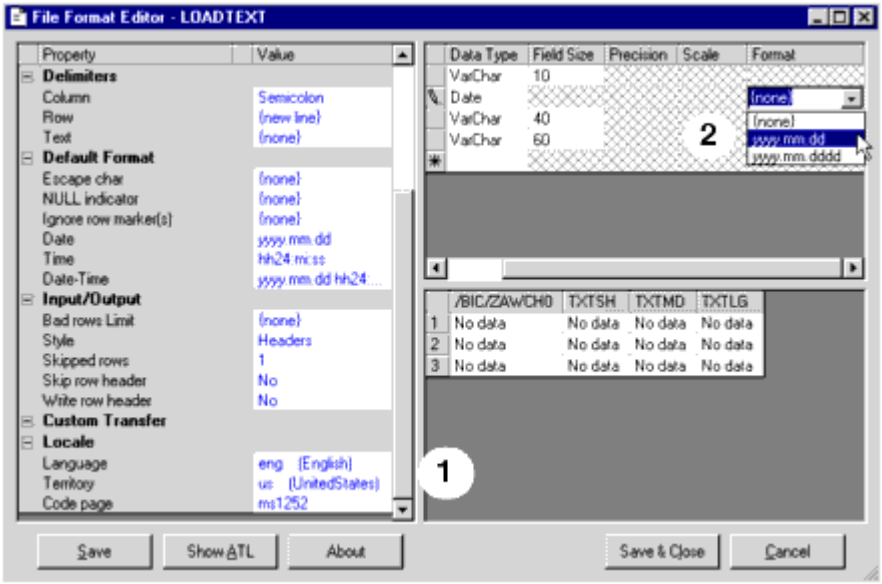
Minimize transcoding [page 1451]
Supported locales and encodings [page 1463]
Example locale settings [page 1455]

11.5.2 Guidelines for file format locales

SAP Data Services sets the locale for flat files to `default` automatically so the locale matches the locale 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 Information

File format [page 71]
Designer Guide: File Formats, Number formats

11.5.3 Guidelines for database, database client, and datastore locales

Set the database and database client using the specific database software.

Before setting the locale for your database, database client, and datastores, consider the following guidelines:

- Set a SAP Data Services datastore to the same locale as the application or database client to which it connects.
- Data Services sets each datastore locale automatically to `default` so that the locale matches the locale of the Job Server. However, if your sources or targets use different locales, manually modify the*Language* and *Code page* options under the *Advanced* options in the datastore editor.
- When you view table data with the View Data feature of the Data Services Designer, Data Services formats the values of numeric data type columns according to the number format of its locale territory.

> **Example**
>
> A datastore that is connected to an Oracle database has a datastore locale of `deu_de.cp1252`. With the Data Services locale set to `eng_us.cp1252`, View Data displays 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. 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 set to the code page UTF-8 automatically by Data Services. Adapter datastores are handled the same way as XML message sources and targets.

Related Information

Datastore [page 54]
Guidelines for XML encodings [page 1454]
to_char [page 1277]
to_decimal [page 1281]
to_decimal_ext [page 1282]

11.5.4 Guidelines for XML encodings

SAP Data Services uses specific guidelines for determining the encoding (code page) for XML Schemas or DTDs.

Data Services assumes that the default encoding for XML Schemas or DTDs used as sources is UTF-8. It ignores any header information. 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 runtime. If encoding information isn't specified, Data Services assumes it's UTF-8. If you edit the encoding for XML file sources, edit the encoding in the original file and then rerun the job.

For XML Schemas or DTDs used as targets, Data Services sets the default encoding 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 can't be changed. Data Services transcodes out-bound message configuration information (like global variable data) into UTF-8.

Related Information

Target XML files, messages, and templates [page 207]

11.5.5 Overriding the database client locale

By default, most database clients set their locale to the same locale used by the operating system. However, you can override the default setting by using either an environment variable or client configuration tool. We recommend using the methods below to override the client locale for different database types.

Database	Default setting	Environment variable	Driver setting
DB2	OS Locale	DB2CODEPAGE	N/A
Informix	OS Locale	CLIENT_LOCALE	Client Locale field
MySQL	Latin1	N/A	Character Set field
Netezza	N/A	N/A	ASCII (8-bit driver) UTF16 (Unicode driver)
Oracle	OS Locale	NLS_LANG	NLS_LANG
SAP ASE	OS Locale	LC_ALL	N/A
SAP Sybase IQ	OS Locale	N/A	Character Set field
Teradata	ASCII	N/A	Session Character Set field

> **i Note**
>
> For Microsoft SQL Server 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.

11.6 Example locale settings

The following examples illustrate setting up locales under different circumstances.

Example 1: Windows system set to Japanese [page 1456]

Example 2: Windows system set to English [page 1456]

Example 3: Multiple databases and different character sets [page 1457]

Parent topic: Locales and multi-byte functionality [page 1437]

Related Information

Terminology for locales and multibyte support [page 1437]
Locales and language packs [page 1440]
Locale support [page 1444]
Transcoding and code page support [page 1448]
Multibyte and Unicode support [page 1457]
Supported locales and encodings [page 1463]

11.6.1 Example 1: Windows system set to Japanese

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.

Parent topic: Example locale settings [page 1455]

Related Information

Example 2: Windows system set to English [page 1456]
Example 3: Multiple databases and different character sets [page 1457]
Locale support [page 1444]
Transcoding and code page support [page 1448]
Example locale settings [page 1455]

11.6.2 Example 2: Windows system set to English

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>`. You must set the Job Server's code page to Japanese (`shift_jis`) to avoid data corruption.

Parent topic: Example locale settings [page 1455]

Related Information

Example 1: Windows system set to Japanese [page 1456]
Example 3: Multiple databases and different character sets [page 1457]
Locale support [page 1444]
Transcoding and code page support [page 1448]
Example locale settings [page 1455]

11.6.3 Example 3: Multiple databases and different character sets

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.

Parent topic: Example locale settings [page 1455]

Related Information

Example 1: Windows system set to Japanese [page 1456]
Example 2: Windows system set to English [page 1456]
Locale support [page 1444]
Transcoding and code page support [page 1448]
Example locale settings [page 1455]

11.7 Multibyte and Unicode support

SAP Data Services supports various multi-byte code pages that are 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.

Parent topic: Locales and multi-byte functionality [page 1437]

Related Information

Terminology for locales and multibyte support [page 1437]
Locales and language packs [page 1440]
Locale support [page 1444]
Transcoding and code page support [page 1448]
Example locale settings [page 1455]
Supported locales and encodings [page 1463]

11.7.1 String functions and multibyte support

All SAP Data Services string functions support multibyte data.

For example, the sizes and offsets used as arguments or return values for the functions `INDEX`, `LENGTH`, `RPAD`, and `SUBSTR`, are expressed as number of characters, not number of bytes.

Data Services treats these functions the same way when the 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.

Parent topic: Multibyte and Unicode support [page 1457]

Related Information

Assign constant values to numeric data types [page 1459]
Byte Order Mark characters [page 1460]
Transcoding and round-trip conversion conflicts [page 1461]
Column sizing and multibyte support [page 1462]
Limitations of multi-byte support [page 1462]

11.7.2 Assign constant values to numeric data types

Use care when you assign a constant to a numeric variable or column in SAP Data Services to avoid unexpected results.

Adhere to specific guidelines when you work with assigning values to numeric data types.

Assign a value as a numeric directly

If a numeric value isn't within quotes, such as `$comm = 123.45`, then Data Services stores the value as a numeric. If the value is stored as numeric, then SAP Data Services Designer recognizes only a dot (".") as the decimal separator, regardless of the locale. However, during job execution, the Data Services engine automatically converts the value to appropriate decimal separator for the locale.

Correct and incorrect syntax

Syntax	Comment
`$VALUE = 123.45`	Correct syntax.
`$VALUE = 123,45`	Wrong syntax. Use a comma "," as a decimal separator inside Data Services only if the operating system's decimal separator is a comma "," and the value is within quotes.

Assign a value in string format

If a numeric value is stored within quotes, such as `$comm = '123,45'`, then the Designer stores the value in a string format. During job execution, 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 based on the locale specified for the Job Server.

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.

Correct and incorrect syntax

Syntax	Comment
`$VALUE = 123,45`	Correct syntax.
`$VALUE = 123.45`	Wrong syntax. If the operating system's decimal separator is a comma "," and the value is within quotes, then Data Services tries to process a dot "." as a thousands separator.

If the Job Server locale is set to English regional settings (`eng_us.cp1252`), then Data Services uses a dot (".") as the decimal separator when converting a value from a string to an appropriate numeric data type.

Correct and incorrect syntax

Syntax	Comment
`$VALUE = 123.45`	Correct syntax.
`$VALUE = 123,45`	Wrong syntax. If the operating system's decimal separator is a dot "." and the value is within quotes, then Data Services tries to process a comma "," as a thousands separator.

> **i Note**
>
> Using an incorrect decimal separator can generate incorrect results. Similarly, using a thousands separator can generate incorrect results. Therefore, SAP recommends that you don't use a thousands separator when you convert a value from string to numeric.

Parent topic: Multibyte and Unicode support [page 1457]

Related Information

String functions and multibyte support [page 1458]
Byte Order Mark characters [page 1460]
Transcoding and round-trip conversion conflicts [page 1461]
Column sizing and multibyte support [page 1462]
Limitations of multi-byte support [page 1462]

11.7.3 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

Parent topic: Multibyte and Unicode support [page 1457]

Related Information

String functions and multibyte support [page 1458]
Assign constant values to numeric data types [page 1459]
Transcoding and round-trip conversion conflicts [page 1461]
Column sizing and multibyte support [page 1462]
Limitations of multi-byte support [page 1462]

11.7.4 Transcoding and round-trip conversion conflicts

If SAP Data Services encounters a round-trip conversion conflict, it transcodes to the first codepoint match in the target code page.

When Data Services transcodes to the first code point match, it does so in ascending order of the hexadecimal values.

> **Example**
>
> There are two different Japanese characters mapped to different hexadecimal code points in the `shift_jis` code page: `EEFA` and `FFE4`. The Japanese characters are then mapped to the same UTF-16 code point: `FFE4`. When Data Services transcodes back from UTF-16 to `shift_jis`, it converts the characters to code point `EEFA`.

Parent topic: Multibyte and Unicode support [page 1457]

Related Information

String functions and multibyte support [page 1458]
Assign constant values to numeric data types [page 1459]
Byte Order Mark characters [page 1460]
Column sizing and multibyte support [page 1462]
Limitations of multi-byte support [page 1462]

11.7.5 Column sizing and multibyte support

Even though SAP Data Services represents column size in number of characters, it internally allocates enough memory to store multibyte characters.

The number of bytes per character can vary from one code page to another. For example, the "£Á" which represents a special "A" in the Korean ksc-5601 code page, needs 2 bytes to represent the character, while the UTF-8 code page needs 3 bytes to represent the same character.

Code pages and byte size

Code page	Value	Hex values	Bytes
ksc-5601	special A	`A3 C1`	2
UTF-8	special A	`EF BC A1`	3

If the datastore code page is different from the Job Server code page, then Data Services must transcode the page. Transcoding requires extra space allocation.

Parent topic: Multibyte and Unicode support [page 1457]

Related Information

String functions and multibyte support [page 1458]
Assign constant values to numeric data types [page 1459]
Byte Order Mark characters [page 1460]
Transcoding and round-trip conversion conflicts [page 1461]
Limitations of multi-byte support [page 1462]

11.7.6 Limitations of multi-byte support

There are several limitations to SAP Data Services' support for multi-byte characters:

- The software supports a variety of single- and multi-byte code pages, but it does not support UCS-4, or SAP application blended code pages. In addition, the software does not support Surrogate Pairs. EBCIDIC is supported for COBOL files only.
  The software does not support EBCDIC code pages for datastores as they are not ASCII- or UTF-8 - compatible. This is not a problem for users targeting or sourcing data from IBM systems if the engine is running on a non-IBM platform. The IBM system will transparently convert its data to/from EBCDIC when communicating with a foreign architecture.
- Each data flow can process in different locales. However, if you need to change the default locale setting, you need to use a different installation of the Job Server on a different computer, because the automatic locale setting will be disabled.
- The software does not yet fully address all formatting issues. For instance, it supports the "dot" and "comma" currency formats used for most European currencies, but does not support the "tick" and "space" currency format used in Switzerland.

Parent topic: Multibyte and Unicode support [page 1457]

Related Information

String functions and multibyte support [page 1458]
Assign constant values to numeric data types [page 1459]
Byte Order Mark characters [page 1460]
Transcoding and round-trip conversion conflicts [page 1461]
Column sizing and multibyte support [page 1462]

11.8 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.

> **i Note**
>
> The combination of language code `zho` and territory `cn` maps to Simplified Chinese, while the combination of `zho` and `tw` maps to Traditional Chinese.

Supported languages

SAP Data Services supports all three-letter language abbreviations specified in the ISO 639-2/T standard.

Supported territories

SAP Data Services supports all two-letter territory abbreviations specified in the ISO 3166-1 standard.

Supported code pages

SAP Data Services supports the following code pages:

	Description			
Code page	**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
cp850	Latin-1 (Western European languages)			
	WINDOWS-850		Microsoft code page 850	2.0
cp1250	Latin-2 (Central Europe)			
	WINDOWS-1250		Microsoft code page 1250	2.1
cp1251	Cyrillic (Slavic)			
	WINDOWS-1251		Microsoft code page 1251	2.1
cp1252	Latin-1 (ANSI), ISO 8859-1 plus Microsoft extensions			
	WINDOWS-1252		Microsoft code page 1252	2.1
cp1253	Greek			
	WINDOWS-1253		Microsoft code page 1253	2.1
cp1254	Latin-5 (Turkish), ISO 8859-9 plus Microsoft extensions			
	WINDOWS-1254		Microsoft code page 1254	2.1
cp1255	Hebrew			
	WINDOWS-1255		Microsoft code page 1255	2.1
cp1256	Arabic			
	WINDOWS-1256		Microsoft code page 1256	2.1
cp1257	Baltic Rim			
	WINDOWS-1257		Microsoft code page 1257	2.1
cp1258	Vietnamese			
	WINDOWS-1258		Microsoft code page 1258	2.1
cp936	Simplified Chinese, GB 2312-80 plus Microsoft extensions. User-defined range added to match Windows 2000 SP4.			
	iso2022cn	Yes	Microsoft code page 936	3.0
euc-jp	Japanese Extended UNIX Code (incl. JIS X 0212)			
	EUC-JP	Yes	Japanese EUC (JIS X 0201-1976, JIS X 0208-1990, JIS X 0212-1990)	2.1

	Description			
Code page	**XML Encoding**	**Multi-byte**	**Vendor**	**Unicode Ver.**
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
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	

	Description			
Code page	XML Encoding	Multi-byte	Vendor	Unicode Ver.
utf-8	UTF-8 encoding of Unicode			
	UTF-8	Yes		2.1

Parent topic: Locales and multi-byte functionality [page 1437]

Related Information

Terminology for locales and multibyte support [page 1437]
Locales and language packs [page 1440]
Locale support [page 1444]
Transcoding and code page support [page 1448]
Example locale settings [page 1455]
Multibyte and Unicode support [page 1457]

12 Python

Python is installed with your SAP Data Services installation; therefore, you're equipped with everything you need to begin coding.

Python is an open-source, object-oriented scripting language. Data Services even has its own Python expression editor, which you can access from the Match and User-Defined transforms.

By using Python with Data Services, you can customize transforms to meet your specific needs during processing.

SAP Data Services supports the Python 3.x programming language for writing expressions with the User-Defined and Match transforms.

> **! Restriction**
>
> Python 3.x isn't backwards compatible. Because of this incompatibility, you have to update any 2.x Python code being used in your User-Defined or Match transforms before you can run data flows in Data Services version 4.2.13 or higher.
>
> To update Python 2.x code to 3.x code, we recommend using the 2to3 conversion utility that ships with Data Services. The utility is located in `%LINK_DIR%\Data Services\DataQuality\python\Tools\scripts`. For detailed information about this utility, see the Python documentation.
>
> In the Match Best Record transform, you must also edit each job that contains a standard strategy. To edit each job, navigate to the *Best Record Editor* and reselect an option from the *Best Record Strategy* drop-down. Once you save and rerun the job, Data Services automatically updates the 2.x Python code to the new 3.x Python code.

If you want more information about the Python language, see the Python web site at www.python.org .

12.1 Python module and supported transforms

SAP Data Services has a Python module that you access using the Expression editor in the supported transforms.

Data Services supports Python in the User-Defined and the Match transforms. Access the classes in the Python module, and build expressions using the Expression editor.

Data Services supports the following classes in the Python module:

- FLDataCollection class
- FLDataManager class
- FLDataRecord class
- FLProperties class
- FLPythonString class

Each of these classes has one or more methods.

Parent topic: Python [page 1467]

Related Information

12.2 Third-party Python libraries

Python expressions require third-party libraries that ensure your Python expressions run correctly.

You must install all third-party Python libraries to the appropriate dynamic library path for your operating system so that SAP Data Services can resolve all dependencies.

To use third-party Python libraries, SAP recommends that you install the module or package in `%LINK_DIR%\DataQuality\python\Lib\site-packages` (Windows) or `$LINK_DIR/DataQuality/python/Lib/site-packages` (Linux, AIX, and Solaris).

If the Python library contains binary components (`.dll` on Windows and `.so` on Unix), they must be in the library path (`PATH` on Windows and `LD_LIBRARY_PATH` on Solaris or Linux).

Parent topic: Python [page 1467]

Related Information

Python module and supported transforms [page 1468]
Python processing modes in User-Defined transform [page 1469]
Clean up Python new memory references [page 1470]
Using Mapped_Name in Python [page 1472]
Using Python escape character in paths [page 1472]
Create expressions with the Python Expression editor [page 1473]
Built-in objects for Python expressions [page 1479]
Python defined classes and methods [page 1480]
Python examples [page 1496]

12.3 Python processing modes in User-Defined transform

The Python processing modes determine how SAP Data Services applies the Python expression.

You can run the User-Defined transform in two processing modes:

- *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.

  > **i Note**
  >
  > Much of the Python coding is done internally so you don't have to worry about properly importing the Python module. Most of the Python examples in this section don't include any imported syntax.

- *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.

Parent topic: Python [page 1467]

Related Information

Python module and supported transforms [page 1468]
Third-party Python libraries [page 1469]
Clean up Python new memory references [page 1470]
Using Mapped_Name in Python [page 1472]
Using Python escape character in paths [page 1472]
Create expressions with the Python Expression editor [page 1473]
Built-in objects for Python expressions [page 1479]
Python defined classes and methods [page 1480]
Python examples [page 1496]

12.4 Clean up Python new memory references

Python methods can create new memory references, which you need to clean up periodically to avoid possible memory leaks.

New memory references result from new objects that refer to a Python method. New objects are returned when you call certain Python methods. For example, the `NewDataRecord` method returns a new object of type `FlDataRecord`. 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 also create a new memory reference. You must clean new memory references to avoid memory leaks.

> ⚠ Caution
>
> If you allow new memory references to accumulate in your scripts, and you run a project that contains the script, you risk a memory leak during processing.

> Example
>
> To retrieve the value of a field, you use the `GetField` method and save the return value as a variable. `GetField` returns the content of the field as a string value. Therefore, include a `del` command to delete the variable at the end of the script as follows:
>
> ```
> Master = SRC.GetField('input.code')
> del Master
> ```

Example

You use the `NewDataRecord` method, but you don't use `Collection.AddRecord(newRec)`. To delete the memory allocation, use the `DeleteDataRecord` method before the `del` command. The following is the expression:

```
newRec = DataManager.NewDataRecord(1)
DataManager.DeleteDataRecord(newRec)
del newRec
```

Further, if you use `Collection.AddRecord(newRec)`, you don't have to use the `DeleteDataRecord` method before you use the `del` command, because you don't own the record.

```
Collection.AddRecord(newRec)
del newRec
```

Example

The following expression gets the first record from a collection and deletes it.

```
newRec = DataManager.NewDataRecord(0)
Collection.GetRecord(newRec, 1)
Collection.DeleteRecord(newRec)
del newRec
```

Parent topic: Python [page 1467]

Related Information

Python module and supported transforms [page 1468]
Third-party Python libraries [page 1469]
Python processing modes in User-Defined transform [page 1469]
Using Mapped_Name in Python [page 1472]
Using Python escape character in paths [page 1472]
Create expressions with the Python Expression editor [page 1473]
Built-in objects for Python expressions [page 1479]
Python defined classes and methods [page 1480]
Python examples [page 1496]

12.5 Using Mapped_Name in Python

To use a shorter name for a long field name in Python expressions, create an alias for the field name using `Mapped_Name`.

When you set up fields in the User-Defined transforms, and you select to use the Python Per Record processing mode, use the `Mapped_Name` to specify a shorter name for fields with long names. Then, when writing your Python scripts, use the `Mapped_Name` for the field name instead of the actual field name to simplify typing the script.

In some transforms, you can specify the same `Mapped_Name` for the field for both input and output. However, when you use the `Mapped_Name` in Python, use unique values for the field on input and output. If you use the same value on both input and output, SAP Data Services issues an error.

When you use the `GetField` or `SetField` methods, make sure that you enter the `Mapped_Name` correctly in the Python code and map the input field in the transform. If you don't use the `Mapped_Name` correctly, and you don't map the field, Data Service issues the following error:

```
FlDataRecord::GetField() error: Invalid field name MAPPED_RECNO.
```

Parent topic: Python [page 1467]

Related Information

Python module and supported transforms [page 1468]
Third-party Python libraries [page 1469]
Python processing modes in User-Defined transform [page 1469]
Clean up Python new memory references [page 1470]
Using Python escape character in paths [page 1472]
Create expressions with the Python Expression editor [page 1473]
Built-in objects for Python expressions [page 1479]
Python defined classes and methods [page 1480]
Python examples [page 1496]

12.6 Using Python escape character in paths

Because the backslash (\) is an escape character in Python, use caution when you write any expressions in Python that references a path.

The backslash character can also appear in substitution parameters or custom options in expressions. Additionally, if your substitution parameter refers to a path, the backslash may cause the same issue.

Avoid the potential for problems with the backslash character in the following ways:

- Use forward slashes.
- Use a double backslash in the path, where the first backslash is the escape character for the second backslash, which is the actual backslash in the path.
- Enclose the substitution value with an “r”, which indicates a raw string. For example:

```
r'[$$TEST]'
```

Parent topic: Python [page 1467]

Related Information

Python module and supported transforms [page 1468]
Third-party Python libraries [page 1469]
Python processing modes in User-Defined transform [page 1469]
Clean up Python new memory references [page 1470]
Using Mapped_Name in Python [page 1472]
Create expressions with the Python Expression editor [page 1473]
Built-in objects for Python expressions [page 1479]
Python defined classes and methods [page 1480]
Python examples [page 1496]

12.7 Create expressions with the Python Expression editor

Access the Python Expression editor from the User-Defined and the Match transforms.

The Python Expression editor is similar to the Smart editor. The editor helps you create Python expressions by providing basic programming features, such as keyword highlighting, auto-completion, auto-indentation, and code tool tips.

Opening the Python Expression editor in the User-Defined transform [page 1474]
: Access the Python Expression editor after you select the processing mode in the User-Defined transform editor.

Opening the Python Expression editor from the Match transform [page 1475]
: Access the Python Expression editor through the *Match Editor* options.

Writing Python code [page 1476]
: In the Python Expression editor, create and edit Python code in the editor pane.

Validating Python syntax [page 1477]
: The Python Expression editor has a validation feature to make sure the syntax is correct.

Fixing syntax errors in Python [page 1478]
: After you select the *Valdidate* button, SAP Data Services displays any errors in the bottom section of the Python Expression editor.

Use Find and Replace in Python [page 1478]
The Find and Replace buttons in the Python Expression Editor help you quickly find text in Python code.

Parent topic: Python [page 1467]

Related Information

Python module and supported transforms [page 1468]
Third-party Python libraries [page 1469]
Python processing modes in User-Defined transform [page 1469]
Clean up Python new memory references [page 1470]
Using Mapped_Name in Python [page 1472]
Using Python escape character in paths [page 1472]
Built-in objects for Python expressions [page 1479]
Python defined classes and methods [page 1480]
Python examples [page 1496]
Smart Editor and the function wizard [page 1061]

12.7.1 Opening the Python Expression editor in the User-Defined transform

Access the Python Expression editor after you select the processing mode in the User-Defined transform editor.

Before you use the Python Expression editor, define your input and output fields for the User-Defined transform in the transform editor.

To access the Python Expression editor, open the User-Defined transform editor in the data flow and perform the following steps:

1. Open the *Options* tab and select *Edit Options*.

   Alternately, select the User-Defined transform in the data flow and choose *Tools* > *User-Defined Editor*.

   The User-Defined editor opens.
2. Select the applicable processing mode to use: *Per Record* or *Per Collection*.
3. Select Python Expression editor in the *Option Editor* pane, and select *Launch Python Editor*.

Task overview: Create expressions with the Python Expression editor [page 1473]

Related Information

Opening the Python Expression editor from the Match transform [page 1475]
Writing Python code [page 1476]
Validating Python syntax [page 1477]
Fixing syntax errors in Python [page 1478]
Use Find and Replace in Python [page 1478]
Smart Editor and the function wizard [page 1061]

12.7.2 Opening the Python Expression editor from the Match transform

Access the Python Expression editor through the *Match Editor* options.

Before you use the Python Expression editor, define your input and output fields for the Match transform in the transform editor.

1. Select the Match transform icon in your data flow and select *Tools* > *Match Editor*
2. Add a best record operation to your Match transform and select the appropriate option values.
3. Select *Yes* in the Custom column of the *Best record action fields* table.

   The Python code becomes editable.
4. Select *Edit Python*.

   The *Python Editor for best Record - Smart Editor* opens.

Task overview: Create expressions with the Python Expression editor [page 1473]

Related Information

Opening the Python Expression editor in the User-Defined transform [page 1474]
Writing Python code [page 1476]
Validating Python syntax [page 1477]
Fixing syntax errors in Python [page 1478]
Use Find and Replace in Python [page 1478]
Smart Editor and the function wizard [page 1061]

12.7.3 Writing Python code

In the Python Expression editor, create and edit Python code in the editor pane.

The Python expression that you create depends on what you need to do with the Match or User-Defined transform.

The Python Expression editor includes keyword highlighting, auto-completion, auto-indentation, and code tool tips. As you type, the Python Expression editor highlights the correct Python syntax. It also auto-completes:

- The Python objects, functions, classes, and methods
- The SAP Data Services generated variables

Python API

The Python API tab lists the objects, functions, classes, and methods that are available for the specific transform and processing mode. When you select an item in the tab, information about it appears in the help area.

Input and output fields

The I/O Fields tab displays the input fields and output fields that have been mapped in the User-Defined transform. You can also add, delete, and edit the properties of user-defined input and output fields from this tab by right-clicking *Input Fields* or *Output Fields* and selecting *Insert*, *Delete*, or *Properties*.

The field variable name is the Mapped_Name option. For example, in a transform you may have an input field as follows:

- *Mapped_Name:* BEST_PRIMNAME1_IN

You could also have an output field as follows:

- *Mapped_Name:* BEST_PRIMNAME1_OUT

The Python Expression editor works in this way for both Per Record and Per Collection processing modes.

Custom options

For the User-Defined transform, you can create custom options that are used as variables within the transform (for example, a file path). You create the custom option in the User-Defined editor, and it is then displayed in the Variables tab of the Python Expression editor. Like substitution parameters, custom options are assigned $$ as a prefix and are enclosed in brackets (for example, [$$PATH]).

If you have both custom options and substitution parameters in your data flow, substitution parameters take precedence over custom options.

Parent topic: Create expressions with the Python Expression editor [page 1473]

Related Information

Opening the Python Expression editor in the User-Defined transform [page 1474]
Opening the Python Expression editor from the Match transform [page 1475]
Validating Python syntax [page 1477]
Fixing syntax errors in Python [page 1478]
Use Find and Replace in Python [page 1478]
Smart Editor and the function wizard [page 1061]
Python examples [page 1496]

12.7.4 Validating Python syntax

The Python Expression editor has a validation feature to make sure the syntax is correct.

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

Parent topic: Create expressions with the Python Expression editor [page 1473]

Related Information

Opening the Python Expression editor in the User-Defined transform [page 1474]
Opening the Python Expression editor from the Match transform [page 1475]
Writing Python code [page 1476]
Fixing syntax errors in Python [page 1478]
Use Find and Replace in Python [page 1478]

12.7.5 Fixing syntax errors in Python

After you select the *Valdidate* button, SAP Data Services displays any errors in the bottom section of the Python Expression editor.

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.

Task overview: Create expressions with the Python Expression editor [page 1473]

Related Information

Opening the Python Expression editor in the User-Defined transform [page 1474]
Opening the Python Expression editor from the Match transform [page 1475]
Writing Python code [page 1476]
Validating Python syntax [page 1477]
Use Find and Replace in Python [page 1478]

12.7.6 Use Find and Replace in Python

The Find and Replace buttons in the Python Expression Editor help you quickly find text in Python code.

Instead of browsing through lines of code, use the *Find* and—or *Replace* buttons to search for specific text and replace found text with other text. The Expression Editor highlights the search term in the script.

To customize your search, select the options *Match case* or *Match whole word only*.

Parent topic: Create expressions with the Python Expression editor [page 1473]

Related Information

Opening the Python Expression editor in the User-Defined transform [page 1474]
Opening the Python Expression editor from the Match transform [page 1475]
Writing Python code [page 1476]

Validating Python syntax [page 1477]
Fixing syntax errors in Python [page 1478]

12.8 Built-in objects for Python expressions

To help write Python expressions, use the built-in objects that SAP Data Services provides.

Write Python expressions in the User-Defined transform. The User-Defined transform supports two processing modes:

- Per Record
- Per Collection

The following table lists the built-in objects for Python expressions, and whether you use it in Per Record or Per Collection mode.

> **i Note**
>
> Because Python is case-sensitive, ensure that you use the exact capitalization as in the following table.

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
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

Parent topic: Python [page 1467]

Related Information

Python module and supported transforms [page 1468]
Third-party Python libraries [page 1469]
Python processing modes in User-Defined transform [page 1469]
Clean up Python new memory references [page 1470]
Using Mapped_Name in Python [page 1472]
Using Python escape character in paths [page 1472]
Create expressions with the Python Expression editor [page 1473]
Python defined classes and methods [page 1480]
Python examples [page 1496]

12.9 Python defined classes and methods

When you write Python expressions, use the Python defined classes and methods.

The following table lists every SAP Data Services defined class and the class supported methods. The table indicates whether you can use the classes and methods in the User-Defined transform or 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
FIPython String	GetBuffer			Yes
	Set Buffer			Yes

Python class: FIDataCollection [page 1481]

The `FIDataCollection` class methods manipulate entire collections of data or records.

Python class: FIDataManager [page 1487]
: Use `FlDataManager` class methods when you want to create new records.

Python class: FIDataRecord [page 1490]
: Use Data Services-defined `FlDataRecord` class methods to manipulate existing individual records.

Python class: FIProperties [page 1492]
: Use the `FIProperties` class to access various properties of the system in which the Python expression is running.

Python class: FIPythonString [page 1494]
: Use the `FlPythonString` methods to customize your data processing.

Parent topic: Python [page 1467]

Related Information

Python module and supported transforms [page 1468]
Third-party Python libraries [page 1469]
Python processing modes in User-Defined transform [page 1469]
Clean up Python new memory references [page 1470]
Using Mapped_Name in Python [page 1472]
Using Python escape character in paths [page 1472]
Create expressions with the Python Expression editor [page 1473]
Built-in objects for Python expressions [page 1479]
Python examples [page 1496]

12.9.1 Python class: FIDataCollection

The `FIDataCollection` class methods manipulate entire collections of data or records.

Use FIDataCollection class methods to:

- Manipulate entire collections of data or records
- When adding new records to a collection that did not exist before
- Helpful during real-time matching process

FIDataCollection: AddRecord [page 1482]
: To add records to new collections, use the `AddRecord` method.

FIDataCollection: DeleteRecord [page 1483]
: To remove a specific record from a collection, use the `DeleteRecord` method.

FIDataCollection: GetRecord [page 1484]
: To retrieve a value of a reccord in a specified index position, use the `GetRecord` method.

FIDataCollection: Size [page 1485]
: To count the number of recorrds in a collection, use the `Size` method.

FIDataCollection: Truncate [page 1486]
: To quickly remove all records from a collection without deleting the collection, use the `Truncate` method.

Parent topic: Python defined classes and methods [page 1480]

Related Information

Python class: FIDataManager [page 1487]
Python class: FIDataRecord [page 1490]
Python class: FIProperties [page 1492]
Python class: FIPythonString [page 1494]

12.9.1.1 FIDataCollection: AddRecord

To add records to new collections, use the `AddRecord` method.

Syntax

```
AddRecord(<record>)
```

Description

Adds the record to the new collection.

> **Note**
> For every NewDataRecord(), you can call AddRecord() only once.

> **Note**
> After you call AddRecord(), don't call DeleteDataRecord().

Parameters

The `AddRecord` method has the parameter described in the following table.

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, 'brian', 'boyd', '123 main st', '83301'], \
[2, 'bryan', 'boyde', '456 first st', '83302'], \
[3, 'brina', 'boyle', '789 last ave', '83303']
for rec in aDup:
    newrecord = DataManager.NewDataRecord(1)
    newrecord.SetField('ID',str(rec[0]))
    newrecord.SetField('FIRST_NAME',str(rec[1]))
    newrecord.SetField('LAST_NAME',str(rec[2]))
    newrecord.SetField('ADDRESS',str(rec[3]))
    newrecord.SetField('POSTCODE1',str(rec[4]))
    Collection.AddRecord(newrecord)
```

12.9.1.2 FIDataCollection: DeleteRecord

To remove a specific record from a collection, use the `DeleteRecord` method.

Syntax

```
DeleteRecord(<record>)
```

Description

The `DeleteRecord` method removes the specified record from the collection.

Parameters

The following table describes the parameter for the `DeleteRecord` method.

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 the following code example, a new record (newRec) is created and then deleted.

```
newRec = DataManager.NewDataRecord()
Collection.DeleteRecord(newRec)
```

12.9.1.3 FIDataCollection: GetRecord

To retrieve a value of a reccord in a specified index position, use the `GetRecord` method.

Syntax

```
GetRecord(<record>, <index>)
```

Description

The `GetRecord` method retrieves the value of a record in a collection in the specified index position.

Parameters

The following table describes the parameters in the `GetRecord` method.

Parameter	Description
record	Substitute the name of the record object. This parameter is a variable that you must define; for example, `record = DataManager.NewDataRecord()`.
index	Substitute the numerical index value of the record in the collection.

Return value

The `GetRecord` method 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)
```

12.9.1.4 FIDataCollection: Size

To count the number of recorrds in a collection, use the `Size` method.

Syntax

```
Size()
```

Description

The `Size` method counts the number of records in the collection.

Parameters

None.

Return value

The `Size` method returns an integer that refers to the number of records in the collection.

Example

The following example retrieves the number of records in the collection.

```
collectionSize = Collection.Size()
```

12.9.1.5 FIDataCollection: Truncate

To quickly remove all records from a collection without deleting the collection, use the `Truncate` method.

Syntax

```
Truncate()
```

Description

`Truncate` removes all records from a collection, but does not delete the collection.

Parameters

None.

Return value

None.

Example

The following example deletes all records from a collection instead of deleting a record one by one.

```
Collection.Truncate()
```

12.9.2 Python class: FIDataManager

Use `FlDataManager` class methods when you want to create new records.

FIDataManager: DeleteDataRecord [page 1487]
: To delete the memory of a record object allocated using `NewDataRecord()`, use the `DeleteDataRecord` method in the `FlDataManager` class.

FIDataManager: NewDataRecord [page 1488]
: To create a new record object, use the `NewDataRecord` method.

Parent topic: Python defined classes and methods [page 1480]

Related Information

Python class: FIDataCollection [page 1481]
Python class: FIDataRecord [page 1490]
Python class: FIProperties [page 1492]
Python class: FIPythonString [page 1494]

12.9.2.1 FIDataManager: DeleteDataRecord

To delete the memory of a record object allocated using `NewDataRecord()`, use the `DeleteDataRecord` method in the `FlDataManager` class.

Syntax

```
DeleteDataRecord(<record>)
```

Description

Deletes the memory of a record object allocated using `NewDataRecord()`.

> **Note**
>
> Don't call `DeleteDataRecord()` after calling `AddRecord()`.

Parameters

The following table lists the parameters for the `DeleteDataRecord` method.

Parameter	Description
record	Substitute the name of the record object you want to delete.

Return value

None.

Example

You create a new record object (`newRec`). Then, to delete the memory allocated to the data collection, use `DeleteDataRecord`.

> **Caution**
>
> To avoid a possible memory leak from the Python expression, you must use the `DeleteDataRecord()` method when you use the `NewDataRecord()` method.

```
newRec = DataManager.NewDataRecord()
DataManager.DeleteDataRecord(newRec)
```

12.9.2.2 FIDataManager: NewDataRecord

To create a new record object, use the `NewDataRecord` method.

> **Syntax**
>
> ```
> NewDataRecord()
> ```

Description

The `NewDataRecord` method creates a new record object. Add the `NewDataRecord` method at the beginning of the expression, if applicable to the specific expression.

> ⚠ Caution
>
> To avoid a possible memory leak with the Python expression, don't use the`NewDataRecord` method in a loop.

Parameters

None.

For exceptions, see the example.

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 `FlDataRecord`.

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('NAME', 'test')

DataManager.DeleteDataManager(newRecord)
```

In the following example, records are read from a database and then added to the original collection. Therefore, the `NewDataRecord()` method requires a numeric parameter of 1.

```
newRecord = DataManager.NewDataRecord(1)
#get the records from the database (excluded from example)
#populate the record
newRecord.SetField('NAME', 'test')
#add record to the collection
Collection.AddRecord(newRecord)
del newRecord
```

> ⚠ Caution
>
> Make sure that you clean up memory references.

12.9.3 Python class: FIDataRecord

Use Data Services-defined `FlDataRecord` class methods to manipulate existing individual records.

FIDataRecord: GetField [page 1490]
: To retrieve the contents of a specified input field, use the `GetField` method.

FIDataRecord: SetField [page 1491]
: To store a value in a specific field, use the `SetField` method.

Parent topic: Python defined classes and methods [page 1480]

Related Information

Python class: FIDataCollection [page 1481]
Python class: FIDataManager [page 1487]
Python class: FIProperties [page 1492]
Python class: FIPythonString [page 1494]

12.9.3.1 FIDataRecord: GetField

To retrieve the contents of a specified input field, use the `GetField` method.

> Syntax
>
> ```
> fieldVal = GetField('<fieldName>')
> ```

Description

The `GetField` method retrieves the contents of the specified input field. Use the `GetField` method with defined input fields only.

Parameters

The following table describes the parameter in the `GetField` method.

Parameter	Description
fieldName	In the Python Expression editor, use one of the input field variables. If you use this method with the Best Record operation of the Match transform, replace this parameter with the `Mapped_Name` that you want to retrieve.

Return value

Returns a new string with the contents of the specified field.

Example

```
if newRecord.GetField('POSTCODE1') == '54601'...
```

12.9.3.2 FIDataRecord: SetField

To store a value in a specific field, use the `SetField` method.

Syntax

```
SetField('<fieldName>', 'value')
```

Description

The `SetField` method stores a value in the specified field.

Parameters

The `SetField` method has the parameters described in the following table.

Parameter	Description
fieldName	In the Python Expression editor, use one of the input or output field variables, which use the `Mapped_Name`.
value	Specifies the value to store in the field.

Return value

None.

Example

Store "Current Resident" in the field named `NAME_SUBSTITUTION`.

```
newRecord.SetField('NAME_SUBSTITUTION', 'Current Resident')
```

12.9.4 Python class: FIProperties

Use the `FIProperties` class to access various properties of the system in which the Python expression is running.

The `FIProperties` class can access the runtime parameters listed in the following table in the SAP 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.

FIProperties: GetProperty [page 1493]
To return the value in a property, use the `GetProperty` method.

Parent topic: Python defined classes and methods [page 1480]

Related Information

Python class: FIDataCollection [page 1481]
Python class: FIDataManager [page 1487]
Python class: FIDataRecord [page 1490]
Python class: FIPythonString [page 1494]

12.9.4.1 FIProperties: GetProperty

To return the value in a property, use the `GetProperty` method.

Syntax

```
var1 = GetProperty(<PropertyName>)
```

Description

The `GetProperty` method returns the value of a given property, which is specified as an input parameter.

Parameters

The `GetProperty` method has one parameter described in the following table.

Parameter	Description
PropertyName	Specifies the environment variable 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('JOB_ID')
#Set the Job Id value into JOB_ID_OUT field
record.SetField('JOB_ID_OUT',str(propValue))
del propValue
```

12.9.5 Python class: FIPythonString

Use the `FlPythonString` methods to customize your data processing.

With the `FlPythonString` methods, you can create a Best Record operation in the Match transform.

FIPythonString: GetBuffer [page 1494]
The `GetBuffer` method returns the specified string.

FIPythonString: SetBuffer [page 1495]

Parent topic: Python defined classes and methods [page 1480]

Related Information

Python class: FIDataCollection [page 1481]
Python class: FIDataManager [page 1487]
Python class: FIDataRecord [page 1490]
Python class: FIProperties [page 1492]

12.9.5.1 FIPythonString: GetBuffer

The `GetBuffer` method returns the specified string.

Syntax

```
GetBuffer()
```

Description

The `GetBuffer` method returns the specified string.

Parameters

None.

Return value

The Unicode character string.

Example

In the following example, *getstr* holds the value of the Unicode character string.

```
getstr = STR.GetBuffer()
```

12.9.5.2 FIPythonString: SetBuffer

Syntax

```
SetBuffer('<stringValue>')
```

Description

The `SetBuffer` method sets character buffer to the object.

Parameters

The following table describes the parameter for the `SetBuffer` method.

Parameter	Description
stringValue	Specifies the string that you want to buffer.

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('input.code')
DESTINATION = DST.GetField('input.code')
# if the master is not empty and the subordinate is
if len(SOURCE.strip()) != 0 and len(DESTINATION.strip()) == 0:
   RET.SetBuffer('T')
else:
   RET.SetBuffer('F')
# delete temporary variables
del SOURCE
del DESTINATION
```

Best Record Action:

```
# store master
SOURCE = SRC.GetField('input.code')
# return master
RET.SetBuffer(SOURCE)
# delete temporary variables
del SOURCE
```

12.10 Python examples

To get you started in writing Python expressions in SAP Data Services, use the Python examples in this section.

The examples in this section are grouped by the type of action performed on data. To apply the examples to your data, modify the expression to fit the type of data and names of fields that you're using.

Even though the transforms that support Python are the User-Defined and Match transforms, you can also perform some of the tasks in the examples in the Query transform.

Python example: Formatting data [page 1497]
To format data using Python methods, use the `SetField` and `GedField` methods from the `FIDataRecord` class.

Python example: Splitting data [page 1499]
To split input data fields in specific ways, use various Python expressions.

Python example: Best Record [page 1501]
Use Python methods for the Best Record operation in the Match transform.

Python example: Assigning source attributes [page 1501]
To assign a source to records in a collection, use various Python methods, including `GetField` and `SetField`.

Parent topic: Python [page 1467]

Related Information

Python module and supported transforms [page 1468]
Third-party Python libraries [page 1469]
Python processing modes in User-Defined transform [page 1469]
Clean up Python new memory references [page 1470]
Using Mapped_Name in Python [page 1472]
Using Python escape character in paths [page 1472]
Create expressions with the Python Expression editor [page 1473]
Built-in objects for Python expressions [page 1479]
Python defined classes and methods [page 1480]

12.10.1 Python example: Formatting data

To format data using Python methods, use the `SetField` and `GedField` methods from the `FIDataRecord` class.

Use the examples in the following table to help set up data formatting in the User-Defined transform.

Example use	Sample Python code
Input data to the User-Defined transform doesn't have data-source identification. Append a field mapped to the source and populate it with **TRC** for all records.	`record.SetField('SOURCE','TRC')`

Input data has a name field mapped to **name** in the User-Defined transform. Change the name data case to upper and put the data into a new name field that is mapped to **uppername**.	`name = record.GetField('name')`
	`uppername = name.upper()`
	`record.SetField('uppername',str(uppername))`
	`del name`
	`del uppername`

Example use	Sample Python code

Input data has an account type indicator field, mapped to **account_type**. Account_type 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 don't contain B or b, output the contents to the new name field.
	`account_type = record.GetField('acct_type')`
	`customer_name = record.GetField('cust')`
	`if account_type.strip().upper() == 'B':`
	`  record.SetField('firm',str(customer_name))`
	`else:`
	`  record.SetField('name',str(customer_name))`
	`del account_type`
	`del customer_name`

The input field that contains firm data is mapped to **firm**. In the User-Defined transform, populate a two-character field mapped to **firm_length** that contains the number of characters in the firm name, padded with zeros.	`field = record.GetField('firm')`
	`firm_length = field.strip().zfill(2)`
	`record.SetField('firm_length',str(firm_length))`
	`del field`
	`del firm_length`

Input data that has some records that don't have a name in the field is 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.
The mapped name for the input field name is **name_in**. In the output field section, the mapped name for the same field is **name_out**.
	`name_in = record.GetField('name')`
	`if len(name_in.strip()) == 0:`
	`  record.SetField('name','Valued Customer')`
	`del name_in`

Input data has 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** field
• The first three characters of the **street** field
	`zip = record.GetField('zip')`
	`street = record.GetField('street')`
	`breakgroupid = zip[0:3] + street[0:3]`
	`record.SetField('breakgroupid',str(breakgroupid))`
	`del zip`
	`del street`
	`del breakgroupid`

Input data has a field mapped to **groupnumber**. In the User-Defined transform, append a field mapped to **groupnumberzeropad**, and populate it with the group number, padded with zeros to 10 characters in length.	`groupnumber = record.GetField('groupnumber')`
	`groupnumberzeropad =`
	`groupnumber.strip().zfill(10)`
	`record.SetField('groupnumberzeropad',str(groupnum`
	`berzeropad))`
	`del groupnumber`
	`del groupnumberzeropad`

Example use	Sample Python code

In the User-Defined transform, append a field mapped to **recordnum**, and populate it with the record number.	`dct = locals()`
	`if 'COUNTER' in dct:`
	`  dct['COUNTER'] = dct['COUNTER'] + 1`
	`else:`
	`  dct['COUNTER'] = 1`
	`record.SetField('recordnum',str(dct['COUNTER']))`

In the User-Defined transform, append a field mapped to **recordnum**, and populate it with the record number, zero padded to 10 characters in length.	`dct = locals()`
	`if 'COUNTER' in dct:`
	`  dct['COUNTER'] = dct['COUNTER'] + 1`
	`else:`
	`  dct['COUNTER'] = 1`
	`recordnum = str(dct['COUNTER']).zfill(10)`
	`record.SetField('recordnum',str(recordnum))`
	`del recordnum`

Data is input in user_group and user_code fields. In the User-Defined transform:
• If the contents of **user_code** is A, B, C, D, E, F, G, or H, output **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.
	`user_code = record.GetField('user_code')`
	`uga = 'A,B,C,D,E,F,G,H'`
	`ugb = 'I,J,K,L,M,N,O,P'`
	`if uga.find(user_code.strip().upper()) > -1:`
	`  record.SetField('user_group','UserGroupA')`
	`elif ugb.find(user_code.strip().upper()) > -1:`
	`  record.SetField('user_group','UserGroupB')`
	`del user_code`
	`del uga`
	`del ugb`

12.10.2 Python example: Splitting data

To split input data fields in specific ways, use various Python expressions.

Use the following examples in the User-Defined transform to split your data in a specific way without changing how it's routed.

Example use	Sample Python code

Input data is mapped to an **account** field with the following format: name / 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 goes to **name**
• The contents after the slash goes to **firm**
 	
The syntax **str** specifies the type of split, and the syntax **3-part** specifies how to split.
	`from flscansplit import ScanSplit`
	`account = record.GetField('account')`
	`name = ScanSplit(account, 'str', '3-part', ['/'])[0]`
	`firm = ScanSplit(account, 'str', '3-part', ['/'])[2]`
	`record.SetField('name',str(name))`
	`record.SetField('firm',str(firm))`
	`del account`
	`del name`
	`del firm`

Input data goes to an **input.account** field, mapped to **account**. The field has an individual'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.
	`from flscansplit import ScanSplit`
	`account = record.GetField('account')`
	`type = ['JT/WROS', 'JT WROS', 'JTWROS', 'JT/TEN', 'JT TEN', 'JTTEN', 'JT/TIC', 'JT TIC', 'JTTIC', 'TEN COM', 'TEN/COM', 'TENCOM']`
	`account_name = ScanSplit(account, 'str', 'before',type)[0]`
	`account_type = ScanSplit(account, 'str', 'before',type)[1]`
	`record.SetField('account_name',str(account_name))`
	`record.SetField('account_type',str(account_type))`
	`del account`
	`del type`
	`del account_name`
	`del account_type`

ScanSplit method

Another option for scan values is to create an external file in a text editor and saved locally with extension “py”. Import the file before the method. Then in the `ScanSplit` method, use the variable in the file in place of the actual scan values.

Example

To accomplish the same **account_name** and **account_type** fields specified in the second example in the table, create a file called `suffixes.py` with the following contents:

```
type = ['JT/WROS', 'JT WROS', 'JTWROS', 'JT/TEN', 'JT TEN', 'JTTEN', 'JT/TIC', 'JT TIC', 'JTTIC', 'TEN COM', 'TEN/COM', 'TENCOM']
```

Then, complete the following expression in the User-Defined transform.

```
from flscansplit import ScanSplit
from suffixes import *
account = record.GetField('account')
account_name = ScanSplit(account, 'str', 'before',type)[0]
account_type = ScanSplit(account, 'str', 'before',type)[1]
record.SetField('account_name',str(account_name))
record.SetField('account_type',str(account_type))
del account
del account_name
```

```
del account_type
```

12.10.3 Python example: Best Record

Use Python methods for the Best Record operation in the Match transform.

The following table contains an example of Python expressions in the Best Record operation in the Match transform.

Example use	Sample Python code

Input data contains data in a field named **gen.phone**, which 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.
 	
The Best Record strategy returns a **True** when the source is populated and the destination is empty. Otherwise, the strategy returns a **False**. At the end, the `GetField` method places the source data into the destination field when the Best Record strategy returns a **True**.
	Best Record strategy:
	
	`Source = SRC.GetField('gen.phone')`
	`Destination = DST.GetField('gen.phone')`
	`if len(Source.strip()) > 0 and`
	`len(Destination.strip()) == 0:`
	`  RET.SetBuffer('T')`
	`else:`
	`  RET.SetBuffer('F')`
	`del Source`
	`del Destination`
	
	Best Record Action:
	
	`RET.SetBuffer(SRC.GetField('gen.phone')`

12.10.4 Python example: Assigning source attributes

To assign a source to records in a collection, use various Python methods, including `GetField` and `SetField`.

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

Input data has a field mapped to **SOURCE_IN**. Use a Python expression to assign source attributes in the **SOURCE_IN** field as follows:
• When the source is **CRM** or **LEADS**, assign source attributes.
• When the source is **DoNotMarket**, assign list attributes.
	`SOURCE_IN = SRC.GetField('SOURCE_IN')`
	`if SOURCE_IN.strip() == 'CRM':`
	`  SOURCE_TYPE_OUT = 'N'`
	`  DRIVER_ORDER_OUT = '020'`
	`  BEST_RECORD_PRIORITY_OUT = '010'`
	`  INCLUDE_IN_SOURCE_COUNT_OUT = 'Y'`
	`  APPLY_BLANK_PENALTY_OUT = 'Y'`
	`  PERFORM_DATA_SALVAGE_OUT = 'N'`
	`  PROTECT_UNIQUE_ID_OUT = 'Y'`
	`elif SOURCE_IN.strip() == 'Leads':`
	`  SOURCE_TYPE_OUT = 'N'`
	`  DRIVER_ORDER_OUT = '010'`
	`  BEST_RECORD_PRIORITY_OUT = '020'`
	`  INCLUDE_IN_SOURCE_COUNT_OUT = 'Y'`
	`  APPLY_BLANK_PENALTY_OUT = 'Y'`
	`  PERFORM_DATA_SALVAGE_OUT = 'N'`
	`  PROTECT_UNIQUE_ID_OUT = 'N'`
	`elif SOURCE_IN.strip() ==`
	`'DoNotMarket':`
	`  SOURCE_TYPE_OUT = 'S'`
	`  DRIVER_ORDER_OUT = '000'`
	`  BEST_RECORD_PRIORITY_OUT = '000'`
	`  INCLUDE_IN_SOURCE_COUNT_OUT = 'Y'`
	`  APPLY_BLANK_PENALTY_OUT = 'N'`
	`  PERFORM_DATA_SALVAGE_OUT = 'N'`
	`  PROTECT_UNIQUE_ID_OUT = 'N'`
	`record.SetField('SOURCE_TYPE_OUT',str(`
	`SOURCE_TYPE_OUT))`
	`record.SetField('DRIVER_ORDER_OUT',st`
	`r(DRIVER_ORDER_OUT))`
	`record.SetField('BEST_RECORD_PRIORITY_`
	`OUT',str(BEST_RECORD_PRIORITY_OUT))`
	`record.SetField('INCLUDE_IN_SOURCE_COU`
	`NT_OUT',str(INCLUDE_IN_SOURCE_COUNT_OU`
	`T))`
	`record.SetField('APPLY_BLANK_PENALTY_O`
	`UT',str(APPLY_BLANK_PENALTY_OUT))`
	`record.SetField('PERFORM_DATA_SALVAGE_`
	`OUT',str(PERFORM_DATA_SALVAGE_OUT))`
	`record.SetField('PROTECT_UNIQUE_ID_OUT`
	`',str(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`

13 Reserved words

There are specific words that you shouldn't use for naming objects because they have special meaning for SAP Data Services.

Do not use reserved words as names for work flows, data flows, transforms, or other design elements that you create. Additionally, avoid using the reserved words as user names when you create a Data Services repository. The words are reserved with any combination of upper- and lower-case letters.

If you have to use reserved words, enter them with double quotation marks as shown in the following example:

```
"PRIMARY"
```

The following table lists the reserved words.

Reserved words

_AL_DEFINE	INTEGER	BY
_AL_IFDEF	IS	CASE
_AL_METADATA_ELEMENT	JOB SERVER	CHAR
_AL_TRAN_FUNCTION	LOCAL	CONCAT
_MEMORY	LEFTOUTERJOIN	CREATE
_SAP_INNER_JOIN	LOAD	DATAFLOW
ABAP_PROGRAM	LOOKUP	DATE
ACTAGUICOMMENT	MOD	DECIMAL
ALL	NOT	DEFAULT
AL_NESTED_TABLE	NUMERIC	DISTINCT
AL_REAL_TIME_DATAFLOW	ON	DOMAIN
AL_REPO_FUNCTION	ORDER	EMBEDDED_DATAFLOW
AL_UNNEST	OUTPUT	END
AL_UNSPECIFIED_PARAMAND	PARALLEL	ERROR
AS	PLAN	ERROR_STEP
BEGIN	PSFT_TREE	FIRM_NOISE_WORD
BULK	READ	FOREIGN

CALL	REFERENCES	FUNCTION
CATCH	RETURNS	FUNC_CHAR
CHARACTER	SAP_TREE	FUNC_DS
CONVERT	SESSION	GLOBAL
CUSTOM	SYSTEM	IN
DATABASE	TABLE	INT
DATASTORE	TRANSFORM	INTERVAL
DATETIME	TRY	KEY
DECLARE	VARCHAR	LIKE
DESC	VIEW	LONG
DISTINCT_KEY	WHERE	NULL
DOUBLE	_AL_ELSE	OR
ELSE	_AL_MESSAGE	OUT
EMBEDDED_DATAFLOW_RT	_AL_STORED_PROCEDURE	PIPE
END_TRY	_FUNC_TABLE	PRIMARY
ERROR_CONDITION	_RFC_FUNCTION	REAL
FILE	_SAP_LEFT_OUTER_JOIN	RETURN
FLOWOUTPUT	ACTA	ROW
FROM	ALGUICOMMENT	SELECT
FUNC_ANY	AL_NEST	SET
FUNC_COL	AL_PROJECT	SYSTEM_PROFILE
FUNC_NUM	AL_RELATION	TIME
GENERATED	AL_RFC_SCHEMA_GROUP	TRANSFORM_SCHEMA_MAPPING
GROUP	AL_UNNEST_SCHEMA_GROUP	VARIABLE
HAVING	AND	VOID
IF	ASC	WHILE
INPUT	BEGIN_SCRIPT	

14 Glossary

access server	A real-time, request-reply message broker that collects message requests, routes them to a real-time service, and delivers a message reply within the user-specified time.
adapter	An external Data Services interface that is created using the Software Developer Kit or is prebuilt and purchased from SAP.
Administrator	A browser-based application running on the Data Services Management Console that allows you to schedule, execute, monitor batch jobs, and so on.
after-image	The values in an UPDATE row after the row changes, used for log-based changed-data capture (CDC) jobs.
alias	An alternate form or name.
ANKLink	An NCOALink option that provides additional data about moves that occurred in the previous months 19 through 48.
annotation	A note that you can attach to a workspace diagram to describe or explain job, work flow, or data flow.
application	A software program.
association matching	A method that uses the results of two or more match sets to find matches that could not be found within a single match set.
audit point	The object in a data flow where audit statistics are collected.
batch job	A set of objects that you can schedule and execute together.
batch project	A job that executes at a specific time and ends after all the data in the specific source is processed.
before-image	The values in an UPDATE row before the row changes, used for log-based changed-data capture (CDC) jobs.
best record	A record created by consolidating the most complete, accurate, and up-to-date data elements from matching records.
best record priority	A value used to designate data from a particular source as having more or less importance than other data.
Binary Large Object	A field whose data consists of such objects as bitmap graphics, images, OLE objects, and metafiles. See also blob.
blank penalty	A setting that assigns a lower priority to records in which a particular field is blank.
blob	A field whose data consists of such objects as bitmap graphics, images, OLE objects, and metafiles. See also Binary Large Object.
blueprint	A sample job that has already been set up to handle a common business problem.
break group	A subset of records that are more likely to match, and which consist of driver and passenger records. Fields commonly used for creating break groups are postcodes,

	account or Social Security numbers, or the first two positions (or characters) of the street name.
break key	A user-defined field used to create a break group.
bulk loading	The moving of large amounts of data into a database to achieve optimal performance.
CASS	A USPS certification that requires software vendors to go through a series of tests to prove that their software correctly codes addresses according to USPS requirements, and produces the required USPS reports. See also Coding Accuracy Support System.
CDC	The process of retrieving changes made to a production data source. This process consolidates units of work, ensures, data is synchronized with the original source, and reduces data volume in a warehouse environment. See also changed-data capture.
CDC checkpoint	A software feature that lets you restrict changed-data capture (CDC) subscription reads.
CDC datastore	A datastore that allows you to limit extracted data to changed data only, and connects a changed-data capture (CDC) table in a source databse to Data Services.
CDC subscription	An option on a source CDC table that defines the start and end of your data set, thereby allowing different data flows to extract data from the same table without corrupting data extracted by other data flows.
central repository	A storage mechanism that contains all information normally found in a local repository (definitions for each object in an application), but is optional and is shared by multiple users, who can check objects in and out of the repository.
changed-data capture	The process of retrieving changes made to a production data source. This process consolidates units of work, ensures, data is synchronized with the original source, and reduces data volume in a warehouse environment. See also CDC.
City directory	A file that is used by the USA Regulatory Address Cleanse transform when processing data from the U.S., and contains a table of city names, states, and ZIP Codes, organized by state and city.
classification	An indicator of the types of situations that apply to a word.
client/server	A distributed technology approach where the processing is divided by function. The server performs shared functions (such as managing communications and providing database services), while the client performs individual user functions.
Coding Accuracy Support System	A USPS certification that requires software vendors to go through a series of tests to prove that their software correctly codes addresses according to USPS requirements, and produces the required USPS reports. See also CASS.
Common Warehouse Model	A specification that enables interchange of data warehouse metadata between tools, platforms, and repositories in distributed heterogeneous environments. See also CWM.
compare buffer	A part of memory reserved for processing break groups (one break group at a time) in the Match or Associate transform. A larger buffer typically helps improve performance.
component	A major piece of the software.

conditional	A single-use object, available in work flows, that allows you to branch the execution logic based on the results of an expression. The conditional takes the form of an if/then/else statement.
connection string	A string version of the initialization properties needed to connect to a database, also known as a "DSN-less" connection. With a connection string you can easily store connection information or pass it between applications.
content type	The type of data in a field in your data source; helps you map your fields when you set up downstream transforms.
contribution value	A value you assign to a match criteria that represents the importance (or weight) you place on that criteria's data.
custom ABAP program	Software that extracts data from an SAP application using custom logic that is not currently supported by Data Services ABAP generation logic, and generates a data set that you use as a source in a data flow or an ABAP data flow.
custom function	A script you create to evaluate or make calculations on input values and produce a return value.
CWM	A specification that enables interchange of data warehouse metadata between tools, platforms, and repositories in distributed heterogeneous environments. See also Common Warehouse Model.
data collection	A collection of information that is sent between transforms.
data flow	A reusable object containing steps to define the transformation of information from source to target.
data record	A row of data that is constructed at run time.
Data Services engine	The core process that reads job information from the Data Services repository and sets up run-time processes that execute the job. The run-time processes extract, transform, and load relational and hierarchical data. The Job Server starts the Data Services engine to execute batch or real-time jobs.
Data Services repository	The database that contains information about a Data Services application. The repository contains information about defined reusable objects, and the metadata for sources and targets, transforms, and functions. The repository also contains the job history and runtime statistics information.
Data Services service	The process that ensures that the Access Server and the Job Server are running.
data set	Rows of data with a defined schema.
data source name	A parameter that provides connectivity for a Windows user to a database through an Open Database Connectivity (ODBC) driver. See also DSN.
data transformation	The phase of the data movement process that occurs between extraction and loading.
data transport	A step in an ABAP data flow that defines a target to store the data set extracted during the flow. You can locate the target file on the SAP Application server or in a location accessible to both the SAP Application server and to Data Services across a network.
data type	The format used to store a value, which can imply a default format for displaying and entering the value.

data validation	The process of defining rules to which correct data should conform. In Data Services you define these rules in the Validation transform.
database link	A communication path from one database server to another.
Database Management System	A software application that builds and maintains database tables. See also DBMS.
DataConnector	An operator instance used to read data files generated by Data Services when performing bulk loading using the Teradata Warehouse Builder.
datastore	A logical channel connecting Data Services to a source or target application.
datastore configuration	The definition of a connection to a particular database from a single datastore.
DBMS	A software application that builds and maintains database tables. See also Database Management System.
debug mode	A state of operation that allows you to diagnose errors while executing a job using the interactive debugging features in the Designer.
degree of parallelism	A property of a data flow that defines how many times each transform defined in the data flow replicates for use on a parallel subset of data. See also DOP.
delimiter	A character sequence used to separate column, row, and text data. To separate columns, a delimiter can be a tab, semicolon, comma, space, or any character sequence. To separate rows of data, a delimiter can be a {new line} or any other character sequence. To denote the start and end of a character string, a delimiter can be single quotation marks ('), double quotation marks ("), or {none}.
Delivery Point Barcode	A form of Postnet barcode, consisting of 62 bars and based on the combination of ZIP Code, ZIP+4, DPBC, and a check digit. See also DPBC.
Delivery Point Validation	A technology that assists you in validating the accuracy of your address information with the USA Regulatory Address Cleanse transform. With DPV you can identify addresses that are undeliverable as addressed and whether an address is a Commercial Mail Receiving Agency (CMRA). See also DPV.
Designer	A graphical user interface that allows you to design and test Data Services jobs.
diacritical character	A character that contains an accent, dieresis (umlaut), tilde, cedilla, or other distinguishing marks (for example, ä or Ç). You can choose to have standardized data with these types of characters. The application uses the Latin-1 code page for assigning these accents.
dictionary	Relational database that contains a lexicon of words and phrases that the data cleansing packages and the Data Cleanse transform use to identify, parse, and standardize data.
directional	A component of the address line that indicates direction, such as North in "211 North 115th St".
disambiguation	The process of resolving ambiguity.
discrete field	Input or output data that has separate fields for each piece of information, such as addresses and names.

discrete line format	Input source format in which pieces of data are parsed down to nearly the most distinct level. For example, a “first name” field would be discrete, whereas a “name” field that could contain first, middle, or last name information would not be discrete.
DOP	A property of a data flow that defines how many times each transform defined in the data flow replicates for use on a parallel subset of data. See also degree of parallelism.
DPBC	A form of Postnet barcode, consisting of 62 bars and based on the combination of ZIP Code, ZIP+4, DPBC, and a check digit. See also Delivery Point Barcode.
DPV	A technology that assists you in validating the accuracy of your address information with the USA Regulatory Address Cleanse transform. With DPV you can identify addresses that are undeliverable as addressed and whether an address is a Commercial Mail Receiving Agency (CMRA). See also Delivery Point Validation.
drill down	To explore detailed data that was used in creating a summary level of data. How far you drill down depends on the granularity of the data in the warehouse.
driver record	A record that drives the comparison process. Driver records are part of a break group and are compared with passenger records to determine matches.
DSN	A parameter that provides connectivity for a Windows user to a database through an Open Database Connectivity (ODBC) driver. See also data source name.
dual address	A dual address occurs when a record contains two address lines. Two combinations are typical: • PO box and street address: 1000 Main Street, Suite 51 / PO Box 2342 • Rural route or Highway Contract and street address: RR 1 Box 345 / 12784 Old Columbus Road
dual names	Two names included on an address line, such as John and Jane Doe.
Early Warning System	A solution for matching valid delivery points that have been created between updates to the national ZIP+4 directory. EWS uses four months of rolling data found in an intermediate directory that is updated weekly with data from the USPS. See also EWS.
eLOT	A sorting sequence for US mail in which ZIP+4 codes are arranged in the order that they are served by the mail carrier. Compare with Line of Travel (LOT). See also Enhanced Line of Travel.
embedded data flow	A data flow that is called from inside another data flow.
Enhanced Line of Travel	A sorting sequence for US mail in which ZIP+4 codes are arranged in the order that they are served by the mail carrier. Compare with Line of Travel (LOT). See also eLOT.
enterprise application	Software that enables businesses to execute and optimize business and IT strategies in domains like Enterprise Resourcew Planning (ERP), Customer Relationship Management (ERM), or Supply Chain Management (SCM).
enterprise resource planning system	An enterprise application from which Data Services can extract data. See also ERP system.
ERP system	An enterprise application from which Data Services can extract data. See also enterprise resource planning system.
EWS	A solution for matching valid delivery points that have been created between updates to the national ZIP+4 directory. EWS uses four months of rolling data found in an intermediate directory that is updated weekly with data from the USPS. See also Early Warning System.

exception	An error that occurs while executing a job.
expression	A combination of variables, parameters, constants, and functions linked by operation symbols and any required punctuation that describe a rule for calculating a value.
fault code	A numeric value that is assigned to a record after the USA Regulatory Address Cleanse transform validation process that signifies that the particular record was not successfully validated. Each numeric value represent a different type of fault.
file format	A description of how data is or should be organized in a file Data Services reads from or loads to. A file format can be specific to a single file or generic for many files.
Forward Sortation Area	The first three characters of a Canadian alphanumeric postal code, such as K1A in the postal code for Canada Post's Ottawa headquarters, K1A 0B1. See also FSA.
FSA	The first three characters of a Canadian alphanumeric postal code, such as K1A in the postal code for Canada Post's Ottawa headquarters, K1A 0B1. See also Forward Sortation Area.
function	A program that operates on values that are passed to it.
functional area	A virtual group of jobs that relate to the same business function, such as Human Resources or Customers.
gather	To recombine terms, such as alphanumeric terms that you would look up together in the dictionary.
gender	A code that indicates the likelihood of a record being a certain sex. This code is derived from the name and has five possible values: strong male, strong female, weak male, weak female, ambiguous, and unassigned.
generated field	A field that is produced on output by a transform, such as a postcode field generated by the Global Address Cleanse transform.
global suggestion list	A method of completing and populating addresses with minimal data, or offering suggestions for possible matches. This address-entry system is ideal in call center environments or any transactional environment where data cleansing is necessary at the point of entry, and a human being is available to choose one of the suggestions.
host name	A computer's network name (or IP address).
hybrid format	An arrangement for records in which some fields are discrete whereas others are in a multiline format.
impact and lineage analysis	A category of reports on the Management Console that provides end-to-end impact and lineage analysis of Data Services tables and columns and SAP BusinessObjects Enterprise objects such as business views and reports.
import	The process of acquiring information for the Data Services repository.
input source	The records in a database or file that you want the application to process.
interactive debugger	A Designer feature that allows you to step through the data of a job one row at a time using filters and breakpoints on a line.
interface	A type of interaction with Data Services which is either internal (allows you to create datastore connections to natively- supported applications), or external (allows Data Services to to communicate with information exchange technologies such as Web Services and MQ queries).

intersource match	A match between records of different sources.
job	The unit of work that can be scheduled independently for execution by the Administrator. Jobs are special work flows that can be scheduled for execution, but cannot be called by other work flows or jobs.
Job Server	The Data Services software that receives requests from the Designer and the Administrator to start and stop jobs.
join rank	A value given to or calculated for all data sets in a data flow by which the application determines which source to read first when assembling the data set in a join.
LACS	A database of addresses that have been permanently converted, usually due to 911 emergency system implementation. The changes often consist of conversion from rural-style addressing to standardized, city-style addressing, or renumbering of existing city-style addresses. See also Locatable Address Conversion System.
lastline	The address information that contains components such as the locality, region, postcode, and sometimes country name.
LDU	The last three characters of a Canadian alphanumeric postal code. Compare with Forward Sortation Area (FSA). See also Local Delivery Unit.
legacy system	An information or transaction processing system used to store data such as bank balances, inventories, payroll, and manufacturing parts.
license-controlled feature	A feature that is enabled or disabled depending on the product license.
line	The visual connection between objects in a workspace diagram which shows the left-to-right flow path for data moving between those objects during job execution.
Line of Travel	A sorting sequence for US mail in which ZIP+4 codes are arranged in the order that they are served by the mail carrier. See also LOT.
linked datastores	The datastores in a database link relationship.
Local Delivery Unit	The last three characters of a Canadian alphanumeric postal code. Compare with Forward Sortation Area (FSA). See also LDU.
locale	The identification of a record's world region, which controls the format of data when it is stored, processed, or displayed.
locality	A part of the address line of a record that usually refers to the city or town, but which in some countries, such as the United Kingdom, can extend to include the district.
Locatable Address Conversion System	A database of addresses that have been permanently converted, usually due to 911 emergency system implementation. The changes often consist of conversion from rural-style addressing to standardized, city-style addressing, or renumbering of existing city-style addresses. See also LACS.
lookup table	A table that contains data that other tables can reference with lookup functions that return one or more output columns.
LOT	A sorting sequence for US mail in which ZIP+4 codes are arranged in the order that they are served by the mail carrier. See also Line of Travel.
mail piece unit	A version identifier for printers, representing the unique characteristics of a portion of a mailing.

mapped field	A data-quality-specific field used to tell a data quality transform how to process the data in that field.
master record	The first record in a match group.
match	A pair or group of records that are found to be identical, based on the criteria you set.
match criteria	The options and rules you define for how a match key is used to find records in your data.
match group	A collection of records, consisting of a master and subordinate records, that are found to be matching with each other.
match set	A group of options and rules used to perform comparisons on data.
memory datastore	A datastore connection to an in-memory database.
memory table	An internal table used to store a data set in memory while a job runs.
Meta Integration Model Bridge	A Windows stand-alone utility that converts metadata models among design tool formats. See also MIMB.
metadata	Information acquired and maintained to describe tables in source and target databases.
Metropolitan Statistical Area	An aggregation of counties by the US Office of Management and Budget used for statistical purposes. See also MSA.
MIMB	A Windows stand-alone utility that converts metadata models among design tool formats. See also Meta Integration Model Bridge.
MSA	An aggregation of counties by the US Office of Management and Budget used for statistical purposes. See also Metropolitan Statistical Area.
multiline	A database record format in which address data is not consistently located in the same arrangement in all records.
multiline field	Input or output data that has certain address and name data combined in one field.
NANP	A system for structuring telephone numbers that is shared by 19 North American countries including the United States and Canada. See also North American Numbering Plan.
NCOALink	A USPS product consisting of a secured database of approximately 160 million change of address (COA) records consisting of the names and addresses of individuals, families, and businesses who have filed a change of address with the USPS.
nested data	Information in one table that is related to a single row in another table.
noise word	A word that is meaningless to the matching process.
normal source	An origin of records that the application should consider to be good, eligible records.
North American Numbering Plan	A system for structuring telephone numbers that is shared by 19 North American countries including the United States and Canada. See also NANP.
object	Any item that you create in the Data Services Designer.
object definition	The options that describe the operation of an object, which are viewable in the workspace when you open the object.

object dependent	The state of being associated beneath another object. Any object under the highest level object in the hierarchy is object dependent.
object library	A directory management system that provides access to reusable objects.
object version	An instance of an object. Each time you add or check in an object to the central repository, Data Services creates a new version of the object.
operation code	A flag associated with a row in a data set that indicates the status of the data in the row, such as INSERT, UPDATE, DELETE, and NORMAL.
operational dashboard	A type of report on the Management Console that visually displays the status and performance of job and data flow executions.
Option Editor	A feature in Data Services' Data Quality transform editor through which you can change the value for each option within the transform.
Option Explorer	A pane in the Associate, Match, and User-Defined transform editors which shows a list of the option groups within a transform.
option group	A set of choices that control various business rules for a transform.
other source	In a Match transform, data that should be treated as transparent, such as seed sources, and as such are not counted in determining how to characterize a match group.
parameter	A value passed to a work flow or data flow when that flow is called.
partition	The division of table data into sets based on criteria such as a range or list of values in each row.
passenger record	A row of data in a break group that is compared against the driver record.
pattern file	A plain text file that contains user-defined patterns and is used by the Data Cleanse transform, and can be edited by any text editing program.
PMB	A postal delivery location similar to a post-office box but which is hosted by a private company. See also Private Mail Box.
postal code	A system of letters and/or digits used for sorting mail, such as the numeric ZIP Code used in the U.S. and the alphanumeric FSA LDU system used in Canada.
postcode move	A valid postcode that has been split or moved, so only a portion of the the area that had been covered by the one postcode now has two or more postcodes, including the original one, for the same area.
postcode1	The postal code or five-digit ZIP Code (USA).
postcode2	The secondary part of a postal code, such as the "4051" in the United States postcode "54601-4051".
primary entry	A word or phrase in the dictionary that the data cleansing packages and Data Cleanse transform use to identify, parse, and standardize data.
Private Mail Box	A postal delivery location similar to a post-office box but which is hosted by a private company. See also PMB.
project	The highest-level object in the Designer window, which provides you with a way to organize the other objects you create in Data Services.

projection	An operation within a SELECT statement that the software can push to the database; the subset of columns that you map on the Mapping tag in the query editor.
property	An item of information that describes an object, such as its name, description, or date on which it was created.
query transform	A data transformation object that creates a data set that satisfies conditions you specify.
real-time job	A job that executes on-demand as a "request-response" system.
reference file	A file of address data that Data Services can use to match, assign, standardize, and verify addresses.
relational data	A data set in which data in each column contains a scalar value.
Remote Function Call server	A server that allows third-party programs, including SAP Applications and SAP NetWeaver Business Warehouse, to schedule and initiate Data Services jobs and return the results to Data Services. See also RFC server.
Remote Function Call server interface	The node on the Administrator application of the Data Services Management Console where you configure SAP connections to load data into or read data from an SAP NetWeaver Business Warehouse system. See also RFC server interface.
repository	A set of tables that hold user-created and predefined system objects, source and target metadata, and transformation rules.
request/acknowledge operation	An operation that executes a remote HTTP service in the Request Acknowledge mode, wherein acknowledgement is sent only if the operation is successful.
request/reply operation	An operation that sends a request and then awaits notice of the request's result.
reusable object	An object that can be defined, stored, and reused independent of other objects, and is accessible from the object library.
RFC server	A server that allows third-party programs, including SAP Applications and SAP NetWeaver Business Warehouse, to schedule and initiate Data Services jobs and return the results to Data Services. See also Remote Function Call server.
RFC server interface	The node on the Administrator application of the Data Services Management Console where you configure SAP connections to load data into or read data from an SAP NetWeaver Business Warehouse system. See also Remote Function Call server interface.
rule file	A text file that controls how the application parses data.
rule matching	The process of comparing token classifications against defined rules.
sample size	The number of rows to display in the View Data feature.
sampling rate	The number of rows processed after which Data Services writes information to the monitor log file and updates job events.
sampling rows	The parameter that specifies the frequency with which the Management Console Profiler samples data, beginning with the first row of the specified number of sampling rows.

script	A step in a job or work flow that allows you to calculate values to pass to other parts of the job or work flow by calling functions, executing if-then-else statements, and assigning values to variables.
secondary information	Data that helps the application determine how to process a string in various scenarios.
segment	The format with which the data records of IDocs are interpreted.
server group	An association of Job Servers on different computers that can automatically measure resource availability, and distribute batch jobs or part of a job to the Job Server with the lightest load at run time.
similarity score	A percentage that indicates how much two fields or values are considered alike, which is calculated by the application after the comparison process.
single use object	An object that is defined only within the context of one job or one data flow.
smart editor	A flexible Data Services tool used for creating scripts, expressions, and custom functions without having to type the names of existing elements like column, function, and variable names.
snowbird	An informal term for a person with multiple residences who typically changes where he or she resides according to the season.
source group	A collection of data that you can use to prepare a second set of match statistics, combining the statistics for two or more regular sources.
source record	A row that contains the data you want to use for updating or creating your best record.
standards	Business rules that define how Data Cleanse will apply capitalization or other output formatting to data.
star schema	A database design used to format data in a data mart, and which is based on a single fact table to which any number of dimensional tables may be joined.
step	An object that is part of the definition of a work or data flow, which is represented by an icon in the flow diagram, and is connected to other steps to indicate the flow of data through the data flow, or the order of execution in the work flow.
street address	A postal delivery location that consists of a street name and house number.
subordinate record	Any record in a match group other than the master record.
substitution parameter	A text string "alias" that you can use within your job and transforms, and is defined in a substitution parameter configuration. At runtime, that parameter is replaced with its value anywhere it is used in your job.
substitution parameter configuration	The definition of the substitution parameters used throughout your job in a particular run-time environment.
suggestion list	A group of potential matches presented to the user for selection of the correct one.
suppression source	An origin of data that contains records of information that should be excluded from other output destinations.
table	Database information that is organized into rows and columns that the software reads data from or loads data into.

target	The object into which the application loads extracted and transformed data in a data flow.
TDPID	The server name Data Services uses when loading with the bulk loader option. See also Teradata Director Program ID.
Teradata Director Program ID	The server name Data Services uses when loading with the bulk loader option. See also TDPID.
territory	The locale value for a geographical location (usually the country) where a local language is used.
thread	The instance of the program running on behalf of a process.
tokenization	The creation of tokens, which assigns meaning to each piece of word that results from hyphenation in the Data Cleanse dictionary.
transform	A step in a data flow that acts on a data set, and is available through the object library in three cateogories: Data Integrator, Data Quality, and Platform.
try/catch block	A combination of a try object and one or more catch objects that define alternate execution paths in case an error occurs during the execution of a job.
unique identifier	In a Data Quality transform, an ID that is unique to a record or group of matching records, and is sequential, static, and will not change when records are updated or re-processed through the application.
unique record	A record that does not have any matching or subordinate records and so does not belong to any match group after the matching process is complete.
web service request	Any message sent from a web client that requires processing by a real-time job.
web services	A platform on which multiple applications can communicate with each other even though constructed in different languages and on different platforms.
weighted scoring	A method of comparison that lets you use values to place more or less importance on various match criteria during the matching process.
work flow	A reusable object that contains steps defining the order of job execution.
workspace	The area on the Designer window where you can manipulate system objects and graphically assemble data movement processes.
Z4Change	A directory of all U.S. ZIP Codes and ZIP+4 Codes, in which those codes that have changed in the last 12 months are flagged.
ZCF	A directory that is used by the USA Regulatory Address Cleanse transform when processing data from the U.S., and contains a table of city names, states, and ZIP Codes, organized by ZIP Code. See also ZIP City File.
ZIP City File	A directory that is used by the USA Regulatory Address Cleanse transform when processing data from the U.S., and contains a table of city names, states, and ZIP Codes, organized by ZIP Code. See also ZCF.
ZIP plus 4	A postal code that consists of both the USPS's 5-digit ZIP Code and the 4-digit add-on code.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

THE BEST RUN SAP