



Informatica® Data Integration - Free & PayGo
April 2023

Transformations

© Copyright Informatica LLC 2022, 2023

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC.

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation is subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License.

Informatica, Informatica Cloud, Informatica Intelligent Cloud Services, PowerCenter, PowerExchange, and the Informatica logo are trademarks or registered trademarks of Informatica LLC in the United States and many jurisdictions throughout the world. A current list of Informatica trademarks is available on the web at <https://www.informatica.com/trademarks.html>. Other company and product names may be trade names or trademarks of their respective owners.

Portions of this software and/or documentation are subject to copyright held by third parties. Required third party notices are included with the product.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

Publication Date: 2023-04-04

Table of Contents

Preface	8
Informatica Resources.	8
Informatica Documentation.	8
Informatica Intelligent Cloud Services web site.	8
Informatica Intelligent Cloud Services Communities.	8
Informatica Intelligent Cloud Services Marketplace.	8
Data Integration connector documentation.	9
Informatica Knowledge Base.	9
Informatica Intelligent Cloud Services Trust Center.	9
Informatica Global Customer Support.	9
 Chapter 1: Transformations.....	 10
Active and passive transformations.	10
Transformation types.	11
Incoming fields.	12
Field name conflicts.	12
Field rules.	13
Data object preview.	17
Variable fields.	18
Transformation caches.	19
Cache types.	19
Cache files.	19
Cache size.	20
Optimizing the cache size.	20
Expression macros.	21
Macro types.	21
Macro input fields.	21
Vertical macros.	21
Horizontal macros.	26
Hybrid macros.	30
File lists.	31
Manually created file lists.	31
File list commands.	32
Using a file list in a Source transformation.	33
Using a file list in a Lookup transformation.	34
Multibyte hierarchical data.	34
 Chapter 2: Source transformation.....	 35
Source object.	35
File sources.	36

Database sources.	38
Database source properties.	38
Related objects.	39
Advanced relationships.	42
Custom queries.	42
Source filtering and sorting.	43
Multibyte hierarchical data.	43
Source fields.	44
Editing transformation data types.	45
Chapter 3: Target transformation.	46
Target object.	46
Database targets.	47
Database target properties.	48
Database targets created at run time.	49
Update columns for relational targets.	50
Target update override.	50
Multibyte hierarchical data.	52
Target fields.	52
Target transformation field mappings.	53
Configuring a Target transformation.	54
Chapter 4: Aggregator transformation.	56
Group by fields.	56
Sorted data.	57
Aggregate fields.	58
Aggregate functions.	58
Nested aggregate functions.	58
Conditional clauses.	59
Advanced properties.	59
Chapter 5: Expression transformation.	61
Expression fields.	61
Expression editor.	62
Transformation language components for expressions.	62
Expression syntax.	63
String and numeric literals.	63
Adding comments to expressions.	63
Reserved words.	64
Advanced properties.	65
Chapter 6: Filter transformation.	66
Filter conditions.	66

Advanced properties.	67
Chapter 7: Input transformation.	68
Input fields.	68
Chapter 8: Joiner transformation.	69
Join condition.	69
Join type.	70
Advanced properties.	70
Creating a Joiner transformation.	71
Joiner transformation example.	72
Chapter 9: Lookup transformation.	75
Lookup object.	76
Lookup object properties.	77
Custom queries.	78
Lookup condition.	78
Lookup return fields.	79
Advanced properties.	80
Lookup SQL overrides.	83
Guidelines for overriding the lookup query.	84
Lookup source filter.	85
Dynamic lookup cache.	86
Static and dynamic lookup comparison.	86
Dynamic cache updates.	87
Inserts and updates for insert rows.	87
Dynamic cache and lookup source synchronization.	88
Dynamic cache and target synchronization.	89
Field mapping.	89
Ignore fields in comparison.	90
Dynamic lookup query overrides.	90
Persistent lookup cache.	91
Rebuilding the lookup cache.	91
Unconnected lookups.	92
Configuring an unconnected Lookup transformation.	93
Calling an unconnected lookup from another transformation.	93
Connected Lookup example.	94
Dynamic Lookup example.	94
Unconnected Lookup example.	95
Chapter 10: Mapplet transformation.	98
Mapplet transformation configuration.	98
Selecting a mapplet.	99

Mapplet transformation field mappings.	99
Mapplet parameters.	100
Mapplet transformation output fields.	101
Mapplet transformation names.	101
Synchronizing a mapplet.	102
Chapter 11: Normalizer transformation.....	103
Normalized fields.	103
Occurs configuration.	103
Unmatched groups of multiple-occurring fields.	104
Generated keys.	104
Normalizer field mapping.	105
Normalizer field mapping options.	105
Advanced properties.	106
Target configuration for Normalizer transformations.	106
Normalizer field rule for parameterized sources.	106
Mapping example with a Normalizer and Aggregator.	107
Chapter 12: Output transformation.....	110
Output fields.	110
Field mapping.	110
Chapter 13: Rank transformation.....	112
Ranking string values.	112
Rank caches.	113
Defining a Rank transformation.	114
Rank transformation fields.	114
Defining rank properties.	115
Defining rank groups.	116
Advanced properties.	117
Rank transformation example.	118
Chapter 14: Router transformation.....	120
Working with groups.	121
Guidelines for connecting output groups.	121
Group filter conditions.	121
Configuring a group filter condition.	122
Advanced properties.	123
Router transformation examples.	123
Chapter 15: Sequence Generator transformation.....	125
Sequence Generator transformation uses.	125
Sequence Generator output fields.	126

Sequence Generator properties.	127
Disabling incoming fields.	128
Sequence Generator transformation rules and guidelines.	129
Chapter 16: Sorter transformation.	130
Sort conditions.	130
Sorter caches.	130
Advanced properties.	131
Sorter transformation example.	132
Chapter 17: SQL transformation.	135
Stored procedure or function processing.	135
Connected or unconnected SQL transformation for stored procedure processing.	137
Unconnected SQL transformations.	137
Calling an unconnected SQL transformation from an expression.	138
Invoking a stored procedure before or after a mapping run.	139
Query processing.	139
Static SQL queries.	140
Dynamic SQL queries.	141
Passive mode configuration.	142
SQL statements that you can use in queries	143
Rules and guidelines for query processing.	143
SQL transformation configuration.	144
Configuring the SQL type.	145
SQL transformation field mapping.	146
SQL transformation output fields.	147
Advanced properties.	149
Chapter 18: Union transformation.	152
Comparison to Joiner transformation.	152
Planning to use a Union transformation.	153
Input groups.	153
Output fields.	154
Field mappings.	154
Advanced properties.	155
Union Transformation example.	155
Index.	159

Preface

Refer to *Transformations* for information about the transformations that you can include in mappings and mapplets. Learn how to transform your data when you move it from source to target.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit <https://docs.informatica.com>.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Intelligent Cloud Services web site

You can access the Informatica Intelligent Cloud Services web site at <http://www.informatica.com/cloud>. This site contains information about Informatica Cloud integration services.

Informatica Intelligent Cloud Services Communities

Use the Informatica Intelligent Cloud Services Community to discuss and resolve technical issues. You can also find technical tips, documentation updates, and answers to frequently asked questions.

Access the Informatica Intelligent Cloud Services Community at:

<https://network.informatica.com/community/informatica-network/products/cloud-integration>

Developers can learn more and share tips at the Cloud Developer community:

<https://network.informatica.com/community/informatica-network/products/cloud-integration/cloud-developers>

Informatica Intelligent Cloud Services Marketplace

Visit the Informatica Marketplace to try and buy Data Integration Connectors, templates, and mapplets:

<https://marketplace.informatica.com/>

Data Integration connector documentation

You can access documentation for Data Integration Connectors at the Documentation Portal. To explore the Documentation Portal, visit <https://docs.informatica.com>.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit <https://search.informatica.com>. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Intelligent Cloud Services Trust Center

The Informatica Intelligent Cloud Services Trust Center provides information about Informatica security policies and real-time system availability.

You can access the trust center at <https://www.informatica.com/trust-center.html>.

Subscribe to the Informatica Intelligent Cloud Services Trust Center to receive upgrade, maintenance, and incident notifications. The [Informatica Intelligent Cloud Services Status](#) page displays the production status of all the Informatica cloud products. All maintenance updates are posted to this page, and during an outage, it will have the most current information. To ensure you are notified of updates and outages, you can subscribe to receive updates for a single component or all Informatica Intelligent Cloud Services components. Subscribing to all components is the best way to be certain you never miss an update.

To subscribe, go to <https://status.informatica.com/> and click **SUBSCRIBE TO UPDATES**. You can then choose to receive notifications sent as emails, SMS text messages, webhooks, RSS feeds, or any combination of the four.

Informatica Global Customer Support

You can contact a Customer Support Center by telephone or online.

For online support, click **Submit Support Request** in Informatica Intelligent Cloud Services. You can also use Online Support to log a case. Online Support requires a login. You can request a login at <https://network.informatica.com/welcome>.

The telephone numbers for Informatica Global Customer Support are available from the Informatica web site at <https://www.informatica.com/services-and-training/support-services/contact-us.html>.

CHAPTER 1

Transformations

Transformations are a part of a mapping that represent the operations that you want to perform on data. Transformations also define how data enters each transformation.

Each transformation performs a specific function. For example, a Source transformation reads data from a source, and an Expression transformation performs row-level calculations.

Active and passive transformations

A transformation can be active or passive.

An active transformation can change the number of rows that pass through the transformation. For example, the Filter transformation is active because it removes rows that do not meet the filter condition.

A passive transformation does not change the number of rows that pass through the transformation.

You can connect multiple branches to a downstream passive transformation when all transformations in the branches are passive.

You cannot connect multiple active transformations or an active and a passive transformation to the same downstream transformation or transformation input group. You might not be able to concatenate the rows. An active transformation changes the number of rows, so it might not match the number of rows from another transformation.

For example, one branch in a mapping contains an Expression transformation, which is passive, and another branch contains an Aggregator transformation, which is active. The Aggregator transformation performs aggregations on groups, such as sums, and reduces the number of rows. If you connect the branches, Data Integration cannot combine the rows from the Expression transformation with the different number of rows from the Aggregator transformation. Use a Joiner transformation to join the two branches.

Transformation types

After you add a transformation to a mapping, you can define transformation details. Each transformation type has a unique set of options that you can configure.

The following table provides a brief description of each transformation:

Transformation	Description
Source	Reads data from a source.
Target	Writes data to a target.
Aggregator	An active transformation that performs aggregate calculations on groups of data.
Expression	A passive transformation that performs calculations on individual rows of data.
Filter	An active transformation that filters data from the data flow.
Input	A passive transformation that passes data into a mapplet. Can be used in a mapplet, but not in a mapping.
Joiner	An active transformation that joins data from two sources.
Lookup	Looks up data from a lookup object. Defines the lookup object and lookup connection. Also defines the lookup condition and the return values. A passive lookup transformation returns one row. An active lookup transformation returns more than one row.
Mapplet	Inserts a mapplet into a mapping or another mapplet. A mapplet contains transformation logic that you can create and use to transform data before it is loaded into the target. Can be active or passive based on the transformation logic in the mapplet.
Normalizer	An active transformation that processes data with multiple-occurring fields and returns a row for each instance of the multiple-occurring data.
Output	A passive transformation that passes data from a mapplet to a downstream transformation. Can be used in a mapplet, but not in a mapping.
Rank	An active transformation that limits records to a top or bottom range.
Router	An active transformation that you can use to apply a condition to incoming data.
Sequence Generator	A passive transformation that generates a sequence of values.
Sorter	A passive transformation that sorts data in ascending or descending order, according to a specified sort condition.
SQL	Calls a stored procedure or function or executes a query against a database. Passive when it calls a stored procedure or function. Can be active or passive when it processes a query.
Union	An active transformation that merges data from multiple input groups into a single output group.

Incoming fields

An incoming field is a field that enters a transformation from an upstream transformation.

By default, a transformation inherits all incoming fields from an upstream transformation. However, you might want to change the default. For example, you might not need all of the fields from an upstream transformation, or you might need to rename fields from an upstream transformation.

A field rule defines how data enters a transformation from an upstream transformation. You can create field rules to specify which incoming fields to include or exclude and to rename incoming fields as required.

A field name conflict occurs when fields come from multiple transformations and have the same name. To resolve a field name conflict caused by fields from an upstream transformation, you can create a field name conflict resolution to rename incoming fields in bulk.

The following list shows the order of events that take place as fields enter and move through a transformation:

1. Field name conflict resolution rules run, if any are present.
2. Field rules run as fields from upstream transformations enter a transformation.
3. Depending on the transformation type, new fields might be added to a transformation. For example, in a Lookup transformation, fields can enter the transformation from a lookup object.

Field name conflicts

The Mapping Designer generates a field name conflict error when you validate a mapping that has fields with matching names from different transformations. When a field name conflict occurs, you need to ensure that each field has a unique name.

To resolve a field name conflict, you can create a field rule to rename fields. If you create a field rule to resolve a field name conflict, you create the field rule in the upstream transformation.

Alternatively, field name conflict error messages contain a link that you can use to create a field name conflict rule to resolve the field name conflict. A field name conflict rule renames all of the fields from the upstream transformation, not just the fields that cause a conflict.

Field name conflict rules take effect before field rules take effect. Field name conflict rules are only applicable to incoming fields from upstream transformations. Field name conflicts that occur after incoming fields first enter a transformation cannot be corrected by field name conflict rules. For example, you cannot use field name conflict rules to correct field name conflicts that occur due to field rules or activities such as lookup fields. Instead, modify the field rules or transformations that cause the conflict.

Creating a field name conflict resolution

You can resolve a field name conflict by renaming all of the fields coming from an upstream transformation in bulk using the **Resolve Field Name Conflict** dialog box, which you access from a field name conflict error message.

1. Click the link in the error message to access the **Resolve Field Name Conflict** dialog box.
2. Select the upstream transformation that contains the fields you want to rename in bulk.
3. In the **Bulk Rename Options** column, specify whether you want to rename by adding a prefix or by adding a suffix.
4. Enter the text to add to the field names, then click **OK**.

Field rules

Configure a field rule based on incoming fields from an upstream transformation. Then configure the field selection criteria and naming convention for the fields.

When you configure a field rule, you perform the following steps:

1. Choose the incoming fields that you want to include or exclude. To improve processing time and keep a clean set of data, you can include only the incoming fields that you need.
2. Configure the field selection criteria to determine which incoming fields apply to the rule. If you use the Named Fields selection criteria, you can use a parameter for the incoming fields.
3. Optionally, choose to rename the fields. To distinguish fields that come from different sources or to avoid field name conflicts, you can rename incoming fields. If you use the pattern option, you can create a parameter to rename fields in bulk.
4. Verify the order of execution. If you configure multiple rules, you can change the order in which the mapping task applies them.

Note: You cannot configure field rules on Source transformations or Mapplet transformations that contain sources.

Step 1. Choose incoming fields

When you configure a field rule, you indicate whether the rule includes or excludes the incoming fields that you specify.

The include/exclude operator works in conjunction with field selection criteria to determine which incoming fields a field rule affects.

For example, you want a transformation to exclude all binary fields. You select the exclude operator to indicate that the incoming fields that meet the field selection criteria do not pass into the current transformation. Then you specify the binary data type for the field selection criteria.

Step 2. Configure field selection criteria

When you configure a field rule, you specify the field selection criteria to determine which incoming fields apply to the field rule.

You can choose one of the following field selection criteria:

All Fields

Includes all of the incoming fields. You can rename the incoming fields in bulk when you use this option in combination with the **Includes** operator.

Named Fields

Includes or excludes the incoming fields that you specify. Use the **Named Fields** selection criteria to specify individual incoming fields to rename or to include or exclude from the incoming transformation. When you enter the field selection criteria details, you can review all of the incoming fields and select the fields to include or exclude. You can add a field that exists in the source if it does not display in the list. You can also create a parameter to represent a field to include or exclude.

Fields by Data Type

Includes or excludes incoming fields with the data types that you specify. When you enter the field selection criteria details, you can select the data types that you want to include or exclude.

Fields by Text or Pattern

Includes or excludes incoming fields by prefix, suffix, or pattern. You can use this option to select fields that you renamed earlier in the data flow. When you enter the field selection criteria details, you can select a prefix, suffix, or pattern, and define the rule to use.

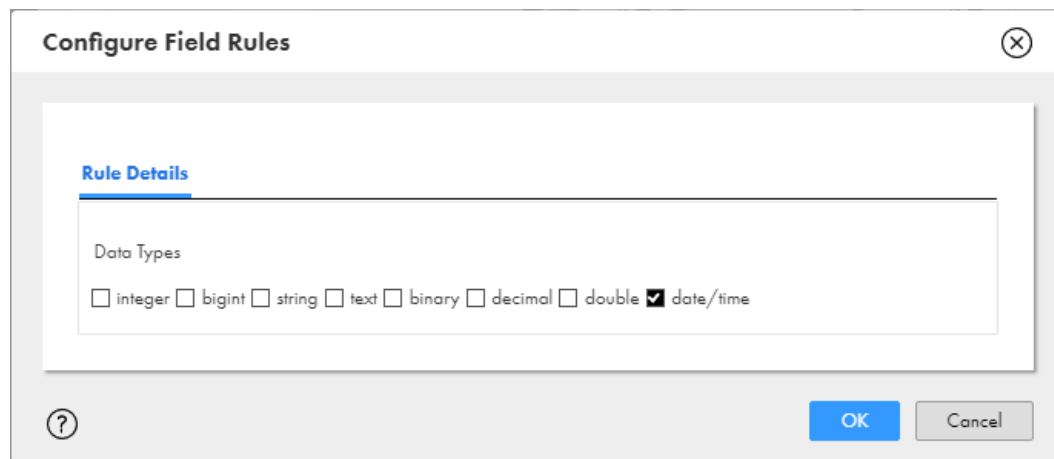
When you select the prefix option or suffix option, you enter the text to use as the prefix or suffix. For example, to find all fields that start with the string, "Cust," enter `Cust` as the prefix.

When you select the pattern option, you can enter a regular expression or you can use a parameter for the pattern. The expression must use perl compatible regular expression syntax. For example, to find all fields that start with the strings "Cust" or "Addr," enter the pattern `Cust.*|Addr.*`. To find all fields that contain the string "Cust" or "CUST" anywhere in the field name, enter the pattern `.*Cust.*|.*CUST.*`. For more information about perl compatible regular expression syntax, see the help for the `REG_EXTRACT` function in *Function Reference*.

The following image shows the selection of the **Fields by Data Types** field selection criteria:



The following image shows the selection of the **date/time** data type for the field selection criteria details:



Step 3. Rename fields

Rename fields to avoid field name conflicts or to clarify field origins in complex mappings. You can rename fields as part of a field rule in a transformation. After you specify the field selection criteria for a field rule, you specify how to rename the selected fields.

You can rename fields individually or in bulk. When you rename fields individually, you select the fields you want to rename from a list of incoming fields. Then you specify the name for each of the selected fields.

When you rename in bulk, you can rename all fields by adding a prefix, suffix, or pattern. When you rename fields with a prefix or suffix, you enter the text string to use as a prefix or suffix. For example, you can specify to rename all fields as `FF_<field name>`.

When you rename fields by pattern, you enter a regular expression to represent the pattern or use a parameter to define the pattern in the task. You can create a simple expression to add a prefix or suffix to all field names or you can create an expression to replace a particular pattern with particular text.

To replace a pattern with text use a regular expression in the following syntax, where a forward slash separates the pattern to match and the text the pattern will be replaced with:

<pattern to match>/<replacement text>

The following table provides a few examples of using regular expressions when you rename fields in bulk:

Goal	Expression
Replace all occurrences of Inc with LLC.	Inc/LLC
Replace occurrences of Inc that occur at the end of a field name with LLC.	Inc\$/LLC
Replace occurrences of Loc that occur at the beginning of a field name with Branch.	^Loc/Branch
Remove all occurrences of A/C.	A\C(.*)/\$1 Note: When a character in a field name is a regular expression metacharacter, escape the character with a backslash to show that it is a literal. In this example, the forward slash is a metacharacter.
Add a prefix of FF and a suffix of _in to all fields.	FF_\$0_in

The following image shows the **Configure Field Rules** dialog box with the Pattern bulk renaming option selected and a pattern specified to use:

Configure Field Rules

Field Selection Criteria: All Fields

Bulk Rename Options:

☒ Pattern:

☐ Parameter:

Carefully construct field renaming rules to ensure that the rules do not introduce issues such as field name conflicts. If a field renaming rule causes field name conflicts, you can edit the rule.

Tip: If the upstream transformation is a source where you cannot rename in bulk, you can add an Expression transformation to rename the fields.

Step 4. Verify order of rule execution

If you create multiple field rules, confirm that the rules run in a logical order.

To review the order in which the rules run, you can view the rules in the **Field Rules** area. The mapping task runs the rules in the order in which the rules appear. If the order of the field rules is incorrect, you can rearrange the order.

You also can preview the incoming fields for the transformation based on the rules that you have created in the Preview Fields table. The Preview Fields table lists all included and excluded fields. For example, if you create a field rule that excludes binary fields, the **Excluded Fields** list shows the binary fields as excluded from the transformation.

If the Source transformation in the mapping uses a connection parameter or a data object parameter, the Preview Fields table does not display the transformation incoming fields.

The following image shows the Preview Fields table:

General

All incoming fields are included by default. You can configure field rules to exclude incoming fields. You can also rename incoming fields to avoid field name conflicts. The Preview Fields table lists all included and excluded fields.

Incoming Fields

Field Rules

Target

Operator

Field Selection Criteria

Detail

Actions

Include

All Fields

All Fields | [Rename...](#)

Target Fields

Exclude

Fields by Data Types

Excluded: [Fields of 1 Data Types](#)

Field Mapping

Included Fields

Excluded Fields

Field Name ^

Type

Precision

Scale


Origin

Address1

string

255

0


 Boston_Customers.csv

Address2

string

255

0


 Boston_Customers.csv

Address3

string

255

0


 Boston_Customers.csv

City

string

255

0


 Boston_Customers.csv

City2

string

255

0

 Boston_Customers.csv

Field rule configuration example

You need to create a mapping to gather revenue data from multiple sales locations.

You learn that multiple fields from the upstream transformation have the same names as fields in a source transformation. To avoid field name conflicts, you decide to change the field names for all incoming fields. You decide to rename the fields so that the source is distinguishable throughout the mapping.

To increase performance, you want to ensure that the data set only includes required data. You determine that information regarding transaction dates is not required, so you decide that the date fields are not necessary for the mapping.

To change the names of all of the incoming fields, you create a field rule to rename all fields with the `SalesForce_` prefix.

To exclude date fields, you create a rule to exclude fields with a date/time data type.

You review the order in which the rules appear. You realize that you want the rule to rename the fields to run after the rule to exclude the date/time fields. You move the rule to remove date/time fields so that it appears before the renaming rule.

Creating a field rule

Configure field rules on the **Incoming Fields** tab of the **Properties** panel in the Mapping Designer.

1. On the **Incoming Fields** tab, in the **Field Rules** area, insert a row for the rule based on the order in which the rules must run. In the **Actions** column for a rule that you want to run before or after the new rule, select either **Insert above** or **Insert below**.
2. To specify whether the rule includes or excludes fields, from the **Operator** column, choose either **Include** or **Exclude**.
3. In the **Field Selection Criteria** column, choose one of the following methods:
 - To rename all of the incoming fields in bulk, select **All Fields**.
 - To apply the rule to the fields that you specify, select **Named Fields**.
 - To apply the rule based on the data type of each field, select **Fields by Data Types**.
 - To apply the rule to all fields that contain a specific prefix, suffix, or pattern, select **Fields by Text or Pattern**.
4. To provide the field selection details, in the **Detail** column, click the **Configure** or **Rename** link. The **Rename** link appears if the field selection criteria is **All Fields**.
5. In the **Configure Field Rules** dialog box, select the fields to apply to the rule, based on the chosen field selection criteria. Alternatively, click **Parameters** to add a parameter so fields can be selected in the mapping task.
6. To rename fields, click the **Rename Fields** tab and choose to rename fields individually or in bulk.
If you want to rename all fields, you must rename in bulk. If you want to rename fields in bulk by pattern, you can create a parameter to specify the pattern in the mapping task.
7. To ensure that the field rules run in a logical order, in the **Field Rules** area, review the order in which the rules display. In the **Included Fields** and **Excluded Fields** lists, review the results of the rules. Move field rules to the appropriate location if required.
8. To delete a rule, in the **Actions** column, select **Delete**.

Data object preview

When you add a Source, Target, or Lookup transformation to a mapping and you select a single object as the source, target, or lookup object, you can preview the data.

This data preview feature is different from mapping data preview. A mapping data preview lets you see data that changed as a result of processing the mapping logic. For information on mapping data preview, see *Mappings*.

To preview the data object, open the **Source**, **Target**, or **Lookup Object** tab of the **Properties** panel, and click **Preview Data**.

When you preview data, Data Integration displays the following information:

- The first 10 rows with the fields in native order.
- The rows with their fields in alphabetical order if you enable the **Display source fields in alphabetical order** option.

If the source or lookup object is a flat file, you can also configure the formatting options. The following table describes the formatting options for flat files:

Property	Description
Flat File Type	File type. Select Delimited . Fixed-width files are not used.
Delimiter	Delimiter character for delimited files.
Text Qualifier	Character to qualify text for delimited files.
Escape Character	Escape character for delimited files.
Field Labels	For delimited files, determines whether the task generates the field labels or imports them from the source file. If you import them from the source file, enter the row number that contains the field labels.
Fixed Width File Format	This option is not used.

Note: Other formatting options might be available based on the connection type. For more information, see the help for the appropriate connector.

Variable fields

A variable field defines calculations and stores data temporarily. You can use variable fields in the Expression and Aggregator transformations.

You might use variables to perform the following tasks:

- Temporarily store data.
- Simplify complex expressions.
- Store values from prior rows.
- Compare values.

For example, you want to generate a mailing list by concatenating first and last names, and then merging the name with the address data. To do this, you might create a variable field, `FullName`, that concatenates the `First` and `Last` fields. Then, you create an expression field, `NameAddress`, to concatenate the `FullName` variable field with the `Address` field.

The results of a variable field does not pass to the data flow. To use data from a variable field in the data flow, create an expression field for the variable field output. In the preceding example, to pass the concatenated first and last name to the data flow, create a `FullName_out` expression field. And then, use the `FullName` variable field as the expression for the field.

Transformation caches

Data Integration allocates cache memory for Aggregator, Joiner, Lookup, Rank, and Sorter transformations in a mapping.

You can configure the cache sizes for these transformations. The cache size determines how much memory Data Integration allocates for each transformation cache at the start of a mapping run.

If the cache size is larger than the available memory on the machine, Data Integration cannot allocate enough memory and the task fails.

If the cache size is smaller than the amount of memory required to run the transformation, Data Integration processes some of the transformation in memory and stores overflow data in cache files. When Data Integration pages cache files to the disk, processing time increases. For optimal performance, configure the cache size so that Data Integration can process the transformation data in the cache memory.

By default, Data Integration automatically calculates the memory requirements at run time based on the maximum amount of memory that it can allocate. After you run a mapping in auto cache mode, you can tune the cache sizes for each transformation.

Cache types

Aggregator, Joiner, Lookup, and Rank transformations require an index cache and a data cache. Sorter transformations require one cache.

The following table describes the type of information that Data Integration stores in each cache:

Transformation	Cache types
Aggregator	Index. Stores group values as configured in the group by fields. Data. Stores calculations based on the group by fields.
Joiner	Index. Stores all master rows in the join condition that have unique keys. Data. Stores master source rows.
Lookup	Index. Stores lookup condition information. Data. Stores lookup data that is not stored in the index cache.
Rank	Index. Stores group values as configured in the group by fields. Data. Stores row data based on the group by fields.
Sorter	Sorter. Stores sort keys and data.

Cache files

When you run a mapping, Data Integration creates at least one cache file for each Aggregator, Joiner, Lookup, Rank, and Sorter transformation. If Data Integration cannot run a transformation in memory, it writes the overflow data to the cache files.

For Aggregator, Joiner, Lookup, and Rank transformations, Data Integration creates index and data cache files to run the transformation. For Sorter transformations, Data Integration creates one sorter cache file. By default, Data Integration stores cache files in the directory entered in the Secure Agent \$PMCCacheDir property for the Data Integration Server. You can change the cache directory on the **Advanced** tab of the transformation properties. If you change the cache directory, verify that the directory exists and contains enough disk space for the cache files.

Cache size

Cache size determines how much memory Data Integration allocates for each transformation cache at the start of a mapping run. You can configure a transformation to use auto cache mode or use a specific value.

Auto cache

By default, a transformation cache size is set to Auto. Data Integration automatically calculates the cache memory requirements at run time. You can also define the maximum amount of memory that Data Integration can allocate in the advanced session properties when you configure the task.

Data Integration allocates more memory to transformations with higher processing times. For example, Data Integration allocates more memory to the Sorter transformation because the Sorter transformation typically takes longer to run.

In transformations that use a data and an index cache, Data Integration also allocates more memory to the data cache than to the index cache. It allocates all of the memory for the Sorter transformation to the sorter cache.

Specific cache size

You can configure a specific cache size for a transformation. Data Integration allocates the specified amount of memory to the transformation cache at the start of the mapping run. Configure a specific value in bytes when you tune the cache size.

You can use session logs to determine the optimal cache size. When you configure the cache size to use the value specified in the session log, you can ensure that no allocated memory is wasted. However, the optimal cache size varies based on the size of the source data. Review the mapping logs after subsequent mapping runs to monitor changes to the cache size.

To define specific cache sizes, enter the cache size values on the **Advanced** tab in the transformation properties.

Optimizing the cache size

For optimal mapping performance, configure the cache sizes so that Data Integration can run the complete transformation in the cache memory.

1. On the **Advanced** tab of the transformation properties, set the tracing level to verbose initialization.
2. Run the task in auto cache mode.
3. Analyze the transformation statistics in the session log to determine the cache sizes required for optimal performance.

For example, you have a Joiner transformation called "Joiner." The session log contains the following text:

```
CMN_1795 [2023-01-06 16:16:59.026] The index cache size that would hold [10005]
input rows from the master for [Joiner], in memory, is [8437760] bytes
CMN_1794 [2023-01-06 16:16:59.026] The data cache size that would hold [10005] input
rows from the master for [Joiner], in memory, is [103891920] bytes
```

The log shows that the index cache size requires 8,437,760 bytes and the data cache requires 103,891,920 bytes.

4. On the **Advanced** tab of the transformation properties, enter the value in bytes that the session log recommends for the cache sizes.

Expression macros

An expression macro is a macro that you use to create repetitive or complex expressions in mappings.

You can use an expression macro to perform calculations across a set of fields or constants. For example, you might use an expression macro to replace null values in a set of fields or to label items based on a set of sales ranges.

In an expression macro, one or more input fields represent source data for the macro. An expression represents the calculations that you want to perform. And an output field represents the results of the calculations.

At run time, the task expands the expression to include all of the input fields and constants, and then writes the results to the output fields.

You can create expression macros in Expression and Aggregator transformations but you cannot combine an expression macro and an in-out parameter in an Expression transformation.

Macro types

You can create the following types of macros:

Vertical

A vertical macro expands an expression vertically. The vertical macro generates a set of similar expressions to perform the same calculation on multiple incoming fields.

Horizontal

A horizontal macro expands an expression horizontally. The horizontal macro generates one extended expression that includes a set of fields or constants.

Hybrid

A hybrid macro expands an expression both vertically and horizontally. A hybrid macro generates a set of vertical expressions that also expand horizontally.

Macro input fields

A macro input field is a field that represents input that you want to use in the expression macro. The input can be fields or constants. All expression macros require a macro input field.

A macro input field in a vertical macro represents a set of incoming fields.

A macro input field in a horizontal macro can represent a set of incoming fields or a set of constants. You can create a multiple macro input fields in a horizontal macro to define multiple sets of constants.

Macro input fields use the following naming convention: %<macro_field_name>%.

For example, you want to apply an expression to a set of address fields. You create a macro input field named %AddressFields% and define a field rule to indicate the incoming fields to use. When you configure the expression, you use %AddressFields% to represent the incoming fields.

Vertical macros

Use a vertical macro to apply a macro expression to a set of incoming fields.

The macro input field in a vertical macro represents the incoming fields. The expression represents the calculations that you want to perform on all incoming fields. And the macro output field represents a set of

output fields that passes the results of the calculations to the rest of the mapping. You configure the macro expression in the macro output field.

The macro output field represents the output fields of the macro, but the names of the output fields are not explicitly defined in the mapping. To include the results of a vertical macro in the mapping, configure a field rule in the downstream transformation to include the output fields that the macro generates.

To write the results of a vertical macro to the target, link the output fields to target fields in the Target transformation.

When the task runs, the task generates multiple expressions to perform calculations on each field that the macro input field represents. The task also replaces the macro output field with actual output fields, and then uses the output fields to pass the results of the calculations to the rest of the mapping.

Note: The macro output field does not pass any data.

Example

The following vertical macro expression trims leading and trailing spaces from fields that the %Addresses% macro input field represents:

```
LTRIM(RTRIM(%Addresses%))
```

At run time, the task generates the following set of expressions to trim spaces from the fields that %Address % represents:

```
LTRIM(RTRIM(Street))  
LTRIM(RTRIM(City))  
LTRIM(RTRIM(State))  
LTRIM(RTRIM(ZipCode))
```

Configuring a vertical macro

You can configure a vertical macro on the **Expression** tab of the Expression transformation or the **Aggregate** tab of the Aggregator transformation.

1. Create a macro input field to define the incoming fields to use.
2. Create a macro output field to define the datatype and naming convention for the output fields.
3. In the macro output field, configure the macro expression. Include the macro input field in the macro expression.
4. In the downstream transformation, configure a field rule to include the results of the macro in the mapping.

Macro input fields for vertical macros

Use a macro input field to represent the incoming fields that you want to use in a vertical macro.

When you create a macro input field, define a name for the macro input field, and then use field rules to define the incoming fields that you want to use. At run time, the macro input field expands to represent the selected fields.

You can use the following field rules when you configure a macro input field:

- All Fields
- Named Fields
- Fields by Text or Pattern

The following image shows a Named Fields field rule that includes the Q1 to Q4 fields:

Select inputs as fields or constant values that you want to pass into the macro input variable.

Named Fields

Choose an input option: ☒ Fields: ☐ Constants ?

▼ Incoming Fields (4 of 12 selected)

<input checked="" type="checkbox"/> Field Name ▲	Origin
<input type="checkbox"/> phone	users.csv
<input checked="" type="checkbox"/> Q1	users.csv
<input checked="" type="checkbox"/> Q2	users.csv
<input checked="" type="checkbox"/> Q3	users.csv
<input checked="" type="checkbox"/> Q4	users.csv

► Parameters (0 of 0 selected)

OK Cancel

Macro output fields for vertical macros

A macro output field represents the output fields that the task generates at run time for a vertical macro. You also define the expression that you want to use in the macro output field.

When you configure a macro output field, you select the macro input field to use and define a naming convention for the output fields. You can customize a prefix or suffix for the naming convention. By default, the macro output field uses the following naming convention for output fields: <macro_input_field>_out.

You can define the data type, precision, and scale of the output fields. Or, you can configure the macro output field to use the datatype, precision, and scale of the incoming fields. Use the datatype of incoming fields when the incoming fields include more than one datatype and when the expression does not change the datatype of incoming data.

At run time, the task generates output fields based on the macro output field configuration. The task creates an output field for each incoming field that the macro input field represents, and then writes the results of the expression to the output fields.

For example, the following image shows a macro output field that creates output fields based on the incoming fields that %QuarterlyData% represents:

New Field

Create new output field, variable field, input macro field or output macro field.

Field Type: Output Macro Field ▼

Input Macro Field:* %QuarterlyData% ▼

Output Macro Field: %QuarterlyData%_out

Suffix: _out

Prefix:

Type:* Input Field Type ▼

Precision:* 0

Scale: 0

? OK Cancel

If the %QuarterlyData% macro input field represents the Q1 to Q4 fields, then the task creates the following output fields at run time: Q1_out, Q2_out, Q3_out, Q4_out. The output fields have the same datatype as the incoming fields.

Note that you cannot define the precision and scale after you select the Input Field Type datatype.

Field rules for vertical macro output fields

To use the results of a vertical macro in a mapping, configure a field rule to include the output fields in the downstream transformation or parameterize the target field mapping.

Because an expression macro represents fields that are not explicitly defined until run time, you need to configure the downstream transformation to include the output fields of a vertical macro. There are two ways to do this:

- Create named fields in the downstream transformation. On the **Incoming Fields** tab, create a named field rule and create a new incoming field for each output field of the vertical macro. You can use these fields in downstream transformations.
- Alternatively, if your Target transformation is directly downstream from the macro, completely parameterize the target field mapping. When you configure the mapping task, Data Integration creates the macro output fields in the target. Map the incoming fields to the target fields.

Example

A macro input field named %InputDates% represents the following source fields for a macro that converts the data to the Date data type:

```
OrderDate
ShipDate
PaymentReceived
```


The macro output field uses the default naming convention: <macro input field>_out. To use the Date fields that the macro generates, create a Named Field rule in the downstream transformation. Create the following fields:

```
OrderDate_out
ShipDate_out
PaymentReceived_out
```

Configure the field rule to include the fields that you create.

After you create the field rule, you can use the fields in expressions and field mappings in downstream transformations.

Vertical macro example

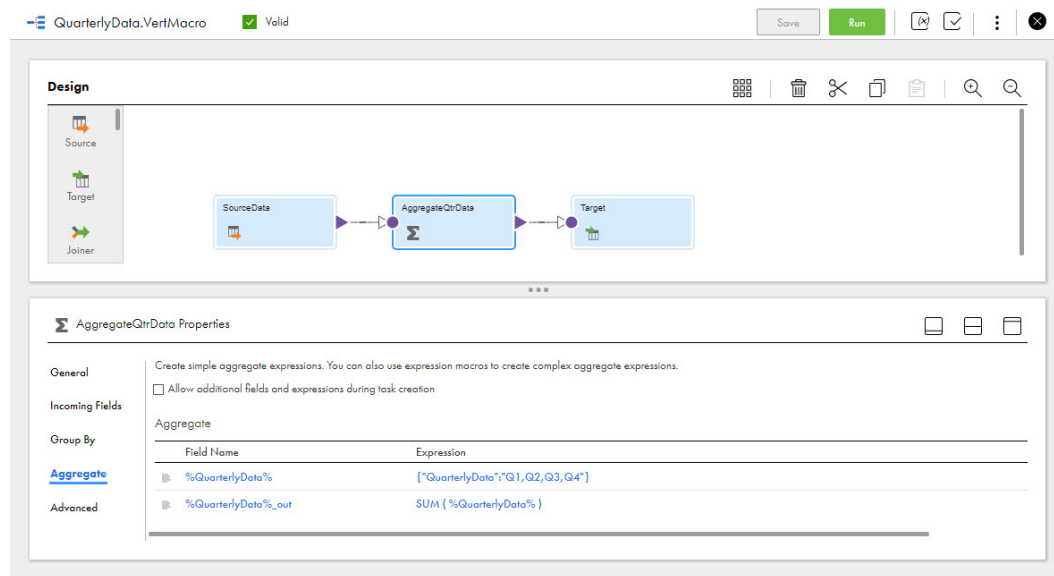
To find the annual sum of quarterly data for each store, you might use a vertical expression macro in an Aggregator transformation.

The Aggregator transformation uses the store ID field as the group by field. A %QuarterlyData% macro input field reads sales data from the following input fields: Q1, Q2, Q3, and Q4.

A %QuarterlyData%_out macro output field is based on the %QuarterlyData% macro input field. To find the sum of sales for each quarter, the macro output field includes the following expression: SUM(%QuarterlyData%).

In the Target transformation, a field rule includes the following output fields in the incoming fields list: Q1_out, Q2_out, Q3_out, Q4_out. In the target field mapping, the Qx_out fields are mapped to the target.

The following image shows the vertical expression macro in an Aggregator transformation:



When the task runs, the expression expands vertically, as follows:

```
SUM (Q1)
SUM (Q2)
SUM (Q3)
SUM (Q4)
```

The task groups the data by store when it performs the aggregation and writes the results to the target.

Horizontal macros

Use a horizontal macro to generate a single complex expression that includes a set of incoming fields or a set of constants.

In a horizontal macro, a macro input field can represent a set of incoming fields or a set of constants.

In a horizontal macro, the expression represents calculations that you want to perform with the incoming fields or constants. The expression must include a horizontal expansion function.

A horizontal macro expression produces one result, so a transformation output field passes the results to the rest of the mapping. You configure the horizontal macro expression in the transformation output field.

The results of the expression pass to the downstream transformation with the default field rule. You do not need additional field rules to include the results of a horizontal macro in the mapping.

To write the results of a horizontal macro to the target, connect the transformation output field to a target field in the Target transformation.

Example

For example, a horizontal macro can check for null values in the fields represented by the %AllFields% macro input field. When a field is null, it returns 1. And then, the %OPR_SUM% horizontal expansion function returns the total number of null fields.

The following expression represents the calculations in the macro:

```
%OPR_SUM[ IIF(ISNULL(%AllFields%), 1, 0) ]%
```

At run time, the application expands the expression horizontally as follows to include the fields that %AllFields% represents:

```
IIF(ISNULL (AccountID, 1,0))+IIF(ISNULL(AccountName, 1, 0))+IIF(ISNULL(ContactName, 1, 0))+IIF(ISNULL(Phone, 1, 0))+IIF(ISNULL(Email, 1, 0)...
```

Horizontal expansion functions

Use a horizontal expansion function to create an expression in an expression macro.

Horizontal expansion functions use the following naming convention: %OPR_<function_type>%. Horizontal expansion functions use square brackets ([]) instead of parentheses.

In the Field Expression dialog box, the functions appear in the Horizontal Expansion group of the functions list.

You can use the following horizontal expansion functions:

%OPR_CONCAT%

Uses the CONCAT function and expands an expression in an expression macro to concatenate multiple fields. %OPR_CONCAT% creates calculations similar to the following expression:

```
FieldA || FieldB || FieldC...
```

%OPR_CONCATDELIM%

Uses the CONCAT function and expands an expression in an expression macro to concatenate multiple fields, and adds a comma delimiter. %OPR_CONCATDELIM% creates calculations similar to the following expression:

```
FieldA || ", " || FieldB || ", " || FieldC...
```

%OPR_IIF%

Uses the IIF function and expands an expression in an expression macro to evaluate a set of IIF statements. %OPR_IIF% creates calculations similar to the following expression:

```
IIF(<field> >= <constantA>, <constant1>,  
    IIF(<field> >= <constantB>, <constant2>,  
        IIF(<field> >= <constantC>, <constant3>, 'out of range')))
```

%OPR_SUM%

Uses the SUM function and expands an expression in an expression macro to return the sum of all fields. %OPR_SUM% creates calculations similar to the following expression:

```
FieldA + FieldB + FieldC...
```

For more information about horizontal expansion functions, see *Function Reference*.

Configuring a horizontal macro

You can configure a horizontal macro on the **Expression** tab of the Expression transformation or the **Aggregate** tab of the Aggregator transformation.

Configure a horizontal macro based on whether you want to use incoming fields or constants in the macro expression.

1. Create one or more macro input fields:
 - To use incoming fields, create a macro input field to define the incoming fields to use.
 - To use constants, create a macro input field for each set of constants that you want to use.
2. Create a transformation output field.
3. In the transformation output field, configure the macro expression. Use a horizontal expansion function and include the macro input fields.
4. To include the results of a horizontal macro in the mapping, use the default field rule in the downstream transformation. You can use any field rule that includes the transformation output field.
5. To write the results of a horizontal macro to the target, connect the transformation output field to a target field in the Target transformation.

Macro input fields for incoming fields in horizontal macros

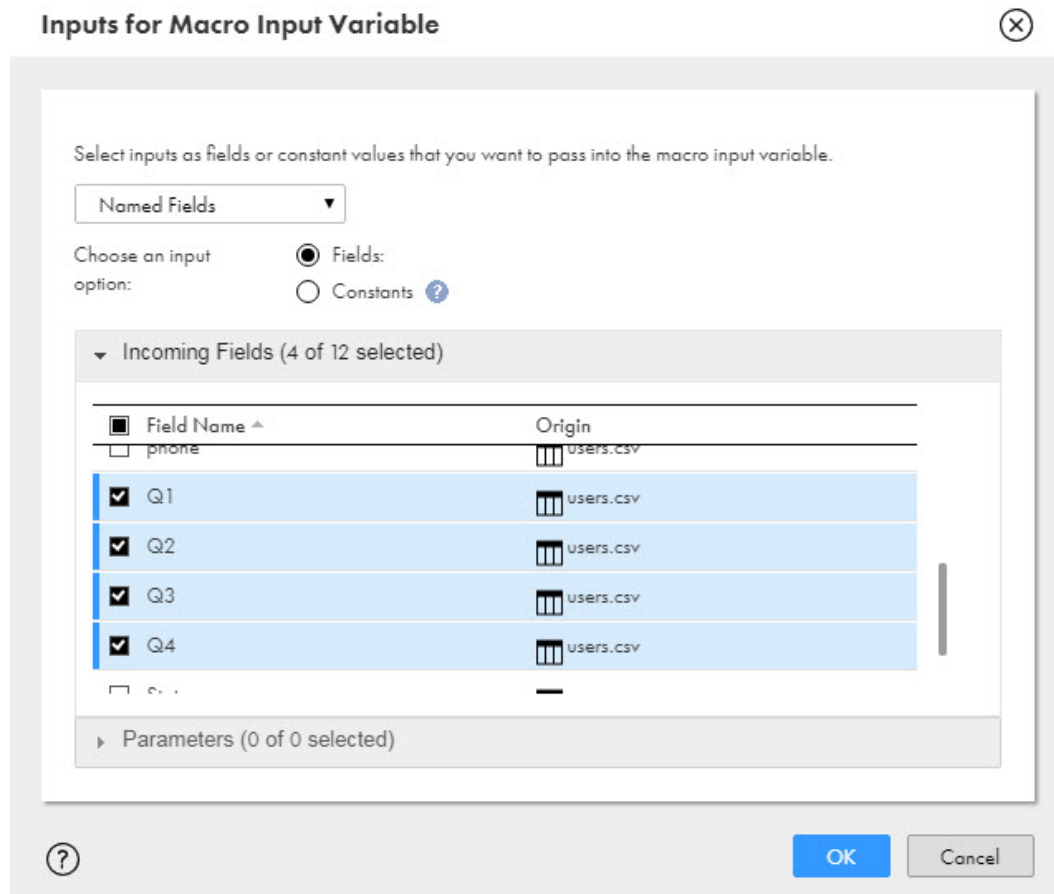
You can use a macro input field to represent the incoming fields that you want to use in a horizontal macro.

When you create a macro input field, define a name for the macro input field, and then use field rules to define the incoming fields that you want to use. At run time, the macro input field expands to represent the selected fields.

You can use the following field rules when you configure a macro input field:

- All Fields
- Named Fields
- Fields by Text or Pattern

The following image shows a Named Fields field rule that includes the Q1 to Q4 fields:



Macro input fields for constants in horizontal macros

You can use a macro input field to represent the constants that you want to use in a horizontal macro. You can also create multiple macro input fields to represent corresponding sets of constants.

When you create a macro input field, define a name for the macro input field, and then define the constants that you want to use. At run time, the macro input field expands to represent the constants and uses them in the listed order.

When you create multiple macro input fields with corresponding sets of constants, the task evaluates each set of constants in the listed order.

The following image shows a macro input field that represents constants:

Inputs for Macro Input Variable

Select inputs as fields or constant values that you want to pass into the macro input variable.

All Incoming Fields ▼

Choose an input option:

☐ Fields:

☒ Constants ?

Constants

Value
50000
100000
150000

?

OK Cancel

At run time, the macro input field expands and uses the constants in the following order: 50000, 100000, 150000.

Transformation output field configuration for horizontal macros

Use a transformation output field to define the expression for a horizontal macro and to pass the results to the rest of the mapping.

When you create a transformation output field, you define the name and datatype for the field. You also configure the expression for the macro. In the expression, include a horizontal expansion function and any macro input fields that you want to use.

The default field rule passes the transformation output field to the downstream transformation. You can use any field rule that includes the transformation output field to pass the results of a horizontal macro to the mapping.

Horizontal macro example

To create categories for employees based on salary ranges, you might create a horizontal macro that defines the minimum and maximum values for each range and corresponding job categories.

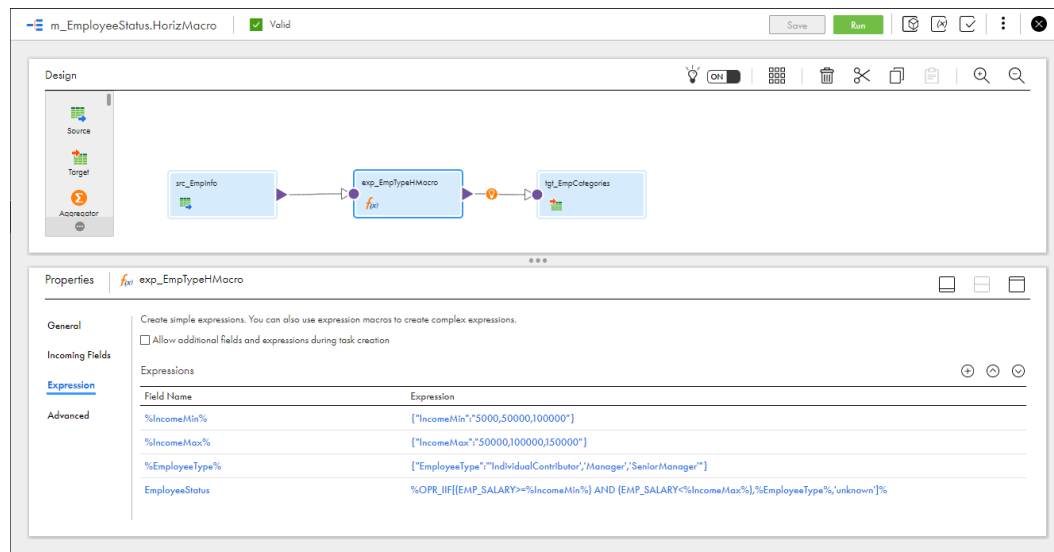
In an Expression transformation, macro input fields define the constants to use in the expression. %IncomeMin% defines the low end of each salary range and %IncomeMax% defines the high end of each salary range. %EmployeeType% lists the job category that corresponds to each range.

The EmployeeStatus transformation output field passes the results to the mapping and includes the following horizontal macro expression:

```
%OPR_IIF[ (EMP_SALARY>=%IncomeMin%) AND (EMP_SALARY<%IncomeMax%), %EmployeeType%, 'unknown' ]%
```

In the Target transformation, the default field rule includes the EmployeeStatus transformation output field in the incoming fields list. In the target field mapping, the EmployeeStatus is mapped to the target.

The following image shows the horizontal macro in an Expression transformation:



The horizontal macro expression expands as follows when you run the task:

```
IIF(Salary>=5000 AND Salary<50000), 'IndividualContributor',  
IIF (Salary>=50000 AND Salary<100000), 'Manager',  
IIF (Salary>=100000 AND Salary<150000), 'SeniorManager', 'unknown')))
```

Note that the expression uses the first value of each macro input field in the first IIF expression and continues with each subsequent set of constants.

Hybrid macros

A hybrid macro expands an expression both vertically and horizontally. A hybrid macro generates a set of vertical expressions that also expand horizontally.

Configure a hybrid macro based on your business requirements. Use the configuration guidelines for vertical and horizontal macros to create a hybrid macro.

Example

For example, the following expression uses the %OPR_IIF% horizontal expansion function to convert the format of the date fields represented by the %dateports% macro input field to the 'mm-dd-yyyy' format:

```
%OPR_IIF[IsDate(%dateports%,%fromdateformat%),To_String(To_Date(%dateports  
%,%fromdateformat%), 'mm-dd-yyyy'), %dateports%]%
```

The %fromdateformat% macro input field defines the different date formats used in the date fields: mm/dd/yy and mm/dd/yyyy.

At run time, the application expands the expression vertically and horizontally, as follows:

```
IIF(IsDate(StartDate, 'mm/dd/yy'), To_String(To_Date(StartDate, 'mm/dd/yy'), 'mm-dd-yyyy'),  
IIF(IsDate(StartDate, 'mm/dd/yyyy'), To_String(To_Date(StartDate, 'mm/dd/yyyy'), 'mm-dd-
```

```

yyyy'), StartDate))

IIF(IsDate(EndDate, 'mm/dd/yy'), To_String(To_Date(EndDate, 'mm/dd/yy'), 'mm-dd-yyyy'),
    IIF(IsDate(EndDate, 'mm/dd/yyyy'), To_String(To_Date(EndDate, 'mm/dd/yyyy'), 'mm-dd-
yyyy'), EndDate))

```

The expression expands vertically to create an expression for the StartDate and EndDate fields that %dateports% represents. The expression also expands horizontally to use the constants that %fromdateformat% represents to evaluate the incoming fields.

File lists

You can use a file list as a source for flat file connections. A file list is a file that contains the names and directories of each source file that you want to use in a mapping. Use a file list to enable a task to read multiple source files for one source object in a mapping.

For example, you might want to use a file list if your organization collects data for multiple locations that you want to process through the same mapping.

You configure a source object such as a Source transformation or Lookup transformation to read the file list. You can also write the source file name to each target row. When you run a mapping that uses a file list, the task reads rows of data from the different source files in the file list.

Use the following rules and guidelines when you create a file list:

- Each file in the list must use the user-defined code page that is configured in the connection.
- Each file in the list must share the same file properties as configured in the connection.
- If you do not specify a path for a file, the task assumes that the file is in the same directory as the file list.
- Each path must be local to the Secure Agent.

You can create a file list manually or you can use a command to create the file list.

Manually created file lists

To create a file list manually, create the list in a text editor and save it as a text file. Place the text file in a directory that is local to the Secure Agent.

When you create the file list, enter one file name or one file path and file name on each line. Data Integration extracts the field names from the first file in the file list.

The following example shows a valid Windows file list:

```

customers_canada.dat
C:\Customer_Accounts\customers_us.dat
C:\Customer_Accounts\customers_uk.dat
C:\Customer_Accounts\customers_india.dat

```

In the previous example, Data Integration extracts the field names from the customers_canada.dat file which resides in the same folder as the file list.

File list commands

You can use a command to generate a list of source files for a mapping. You can use a valid DOS or UNIX command, batch file, or shell script. Data Integration reads each file in the list when the task runs.

Use a command to generate a file list when the list of source files changes often or when you want to generate a file list based on specific conditions. For example, you can use a command to generate a file list from all files in a directory or based on the file names.

Use the following guidelines when you generate a file list through a command:

- You must enter Windows commands that use parameters such as `"/b"` in a batch file.
- You must enter fully qualified file paths in each command, batch file, and shell script.
- You cannot use an in-out parameter for the file list command.

UNIX Example with Shell Script

You need to extract data from parts lists that are stored on a Linux machine. The parts lists are text files that are stored in the `/home/dsmith/flatfile/parts` directory.

The following table shows the command that you enter in the Source transformation and the contents of the corresponding shell script:

Command	Shell Script (parts.sh)
<code>/home/dsmith/flatfile/parts/parts.sh</code>	<pre>cd /home/dsmith/flatfile/parts ls *.txt</pre>

Windows Example with Batch File

You need to extract data from sales records that are stored on a Windows machine. The sales record files are stored in the `C:\SalesRecords` directory and use the naming convention `SalesRec_??-??-2017.txt`.

The following table shows the command that you enter in the Source transformation and the corresponding batch file contents:

Command	Batch File (SalesSrc.bat)
<code>C:\SalesRecords\SalesSrc.bat</code>	<pre>@echo off cd C:\SalesRecords dir /b SalesRec_??-??-2017.txt</pre>

Example without Shell Script or Batch File

You can also generate a file list through a command instead of through a batch file or shell script. For example, the following command generates a file list that contains one file named `source.csv`:

```
echo C:\sources\source.csv
```


Command sample file

When you generate a file list through a command, you select a sample file that Data Integration uses to extract the field names. Data Integration does not extract data from the sample file unless the sample file is included in the generated file list.

If a file in the generated file list does not contain all fields in the sample file, Data Integration sets the record values for the missing fields to null. If a file in the file list contains fields that are not in the sample file, Data Integration ignores the extra fields.

For example, the sample file that you select contains the fields CustID, NameLast, and NameFirst. One file in the generated file list does not contain the NameFirst field. When Data Integration reads data from the file, it sets the first names for each record in the file to null.

Another file in the generated file list contains the fields CustID, NameLast, NameFirst, and PhoneNo. Data Integration does not import records for the PhoneNo field because the field is not in the sample file. If you want to import the phone numbers, either select a sample file that contains the PhoneNo field, or add a field for the phone numbers in the transformation.

Using a file list in a Source transformation

To use a file list in a Source transformation, create the text file, batch file, or shell script that creates the file list. Then configure the Source transformation to use the file list.

1. Create the text file, batch file, or shell script that creates the file list and install it locally to the Secure Agent.
2. In the Mapping Designer, select the Source transformation.
3. On the **Source** tab, select a flat file connection.
4. To use a manually created file list, perform the following steps:
 - a. In the **Source Type** field, select **File List**.
 - b. In the **Object** field, select the text file that contains the file list.
 - c. On the **Fields** tab, verify the incoming fields for the Source transformation.
Data Integration extracts source fields from the first file in the file list. If the source fields are not correct, you can add or remove fields.
5. To use a file list that is generated from a command, perform the following steps:
 - a. In the **Source Type** field, select **Command**.
 - b. In the **Sample Object** field, select the sample file from which Data Integration extracts source fields.
You can use one of the files you use to generate the file list as the sample file or select a different file.
 - c. In the **Command** field, enter the command that you use to generate the file list, for example, `/home/dsmith/flatfile/parts/parts.sh`.
 - d. On the **Fields** tab, verify the incoming fields for the Source transformation.
If the source fields are not correct, you can add or remove fields, or click the **Source** tab and select a different sample file.
6. Optionally, to write the source file name to each target row, click the **Fields** tab, and enable the **Add Currently Processed Filename field** option.
The `CurrentlyProcessedFileName` field is added to the fields table.

Using a file list in a Lookup transformation

To use a file list in a Lookup transformation, create the text file, batch file, or shell script that creates the file list. Then configure the Lookup transformation to use the file list.

1. Create the text file, batch file, or shell script that creates the file list and install it locally to the Secure Agent.
2. In the Mapping Designer, select the Lookup transformation.
3. On the **Lookup Object** tab, select a flat file connection.
4. To use a manually created file list, perform the following steps:
 - a. In the **Source Type** field, select **File List**.
 - b. In the **Lookup Object** field, select the text file that contains the file list.
 - c. On the **Return Fields** tab, verify the return fields for the Lookup transformation.

Data Integration extracts the return fields from the first file in the file list. If the return fields are not correct, you can add or remove fields.
5. To use a file list that is generated from a command, perform the following steps:
 - a. In the **Source Type** field, select **Command**.
 - b. In the **Lookup Object** field, select the sample file from which Data Integration extracts return fields.

You can use one of the files you use to generate the file list as the sample file or select a different file.
 - c. In the **Command** field, enter the command that you use to generate the file list, for example, `/home/dsmith/flatfile/parts/parts.sh`.
 - d. On the **Return Fields** tab, verify the return fields for the Lookup transformation.

If the return fields are not correct, you can add or remove fields, or click the **Lookup Object** tab and select a different sample file.

Multibyte hierarchical data

If a mapping includes a transformation that processes hierarchical data and the data uses multibyte characters, you must configure your system to use UTF-8.

On Windows, create the `INFA_CODEPAGE=UTF-8` environment variable in Windows System Properties.

On Linux, set the `LC_LOCALE` variable to UTF-8.

CHAPTER 2

Source transformation

A Source transformation extracts data from a source. When you add a Source transformation to a mapping, you define the source connection, source objects, and source properties related to the connection type. For some connection types, you can use multiple source objects within a Source transformation.

You can use a Source transformation to read data from the following source types:

- File. The Source transformation can read data from a single source file or a file list.
- Database. The Source transformation can read data from a single source table or multiple source tables.
- Data Integration connectors. The Source transformation can read data from a single source object, a multi-group source object, or multiple source objects based on the connection type.

For more information about sources for individual connectors, see the **Connectors** section of the online help.

You can use one or more Source transformations in a mapping. If you use two Source transformations in a mapping, you can use a Joiner transformation to join the data. If you use multiple Source transformations with the same structure, you can use a Union transformation to merge the data into a single pipeline.

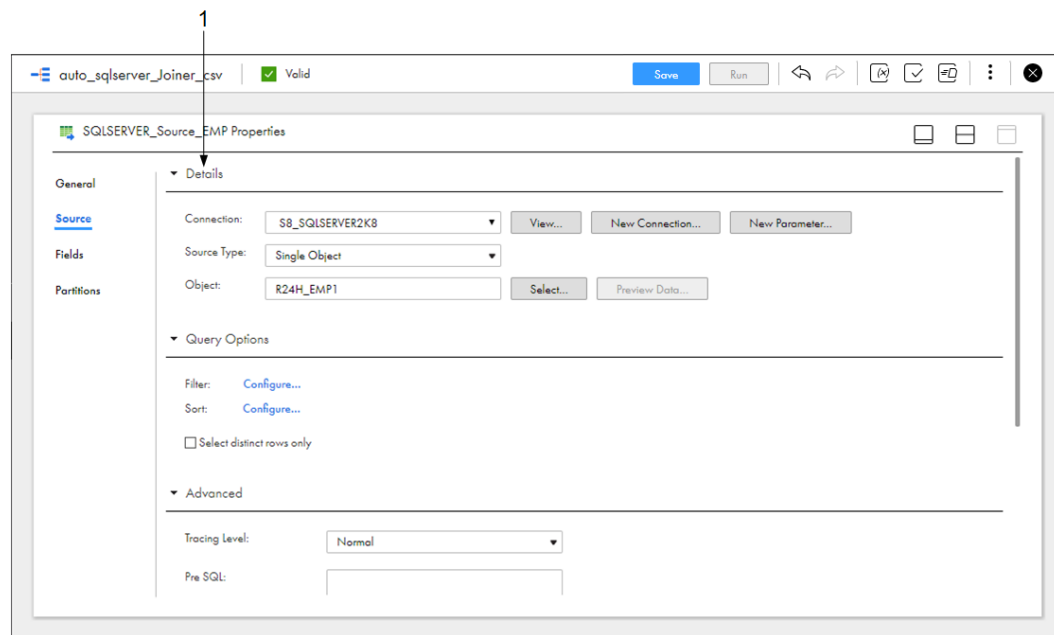
In a Source transformation, the source properties that appear vary based on the connection type. For example, when you select a Salesforce connection, you can use multiple related source objects and configure the Salesforce API advanced source property. In contrast, when you select a flat file connection, you specify the file type and configure the file formatting options.

Source object

Select the source object for the Source transformation on the **Source** tab of the Properties panel.

The properties that you configure for the source object vary based on the connection type and the mapping type.

The following image shows the **Source** tab for a relational source:



1. Source details where you configure the source connection, source type, and source object.

You can select a source in the following ways:

Select the connection and source object.

In the **Details** area, select the source connection, source type, and source object. For some source types, you can select multiple source objects. You can also create a new connection.

The source type varies based on the connection type. For example, for relational sources you can select a single object, multiple related objects, or an SQL query. For flat file sources, you can select a single object, file list, or file list command.

Use a parameter.

You can use input parameters to define the source connection and source object when you run the mapping task. For more information about parameters, see *Mappings*.

File sources

When you configure a file source, you specify the connection, source type, and formatting options. Configure file source properties on the **Source** tab of the **Properties** panel.

The following table describes the file source properties:

Property	Description
Connection	Name of the source connection.
Source Type	Source type. The source type can be single object, file list, command, or parameter.

Property	Description
Object	<p>If the source type is a single object, this property specifies the file source, for example, Customers.csv.</p> <p>If the source property is a file list, this property specifies the text file that contains the file list, for example, SourceList.txt.</p> <p>If the source type is a command, this property specifies the sample file from which Data Integration imports the source fields.</p>
Command	If the source type is a command, this property specifies the command that generates the source file list, for example, ItemSourcesCmd.bat.
Parameter	If the source type is a parameter, this property specifies the source parameter.
Formatting Options	<p>Flat file format options. Opens the Formatting Options dialog box to define the format of the file. Select Delimited. Fixed-width files are not used in Data Integration.</p> <p>For a delimited flat file type, configure the following file format options:</p> <ul style="list-style-type: none"> - Delimiter. Delimiter character. Can be a comma, tab character, colon, semicolon, nonprintable control character, or a single-byte or multibyte character that you specify. - Text Qualifier. Character to qualify text. - Escape character. Escape character. - Field labels. Determines if the task generates field labels or imports labels from the source file. - First data row. The first row of data. The task starts the read at the row number that you enter. <p>You can use a tab, space, or any printable special character as a delimiter. The delimiter can have a maximum of 10 characters. The delimiter must be different from the escape character and text qualifier.</p>

For more information about file lists and commands, see [“File lists” on page 31](#). For more information about parameters and file formats, see *Mappings*.

The following table lists the advanced properties for file sources:

Property	Description
Tracing Level	Detail level of error and status messages that Data Integration writes in the session log. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Thousand Separator	<p>Thousand separator character. Can be none, comma, or period. Cannot be the same as the decimal separator or the delimiter character.</p> <p>Field type must be Number. You might also need to update the field precision and scale.</p> <p>Default is None.</p>
Decimal Separator	<p>Decimal character. Can be a comma or period. Cannot be the same as the thousand separator or delimiter character.</p> <p>Field type must be Number. You might also need to update the field precision and scale.</p> <p>Default is Period.</p>

Property	Description
Source File Directory	<p>Name of the source directory for a flat file source. By default, the mapping task reads source files from the source connection directory.</p> <p>You can also use an input parameter to specify the file directory.</p> <p>If you use the service process variable directory \$PMSourceFileDir, the task writes target files to the configured path for the system variable. To find the configured path of a system variable, see the pmrtdm.cfg file located at the following directory:</p> <pre><Secure Agent installation directory>\apps\Data_Integration_Server\<Data Integration Server version>\ICS\main\bin\rdtm</pre> <p>You can also find the configured path for the \$PMSourceFileDir variable in the Data Integration Server system configuration details in Administrator.</p>
Source File Name	File name, or file name and path of the source file.

Database sources

Database sources include relational sources such as Oracle, MySQL, and Microsoft SQL Server. When you configure a Source transformation for a database source, you can use a single source table or multiple source tables. If you use multiple tables as a source, you can select related tables or create a relationship between tables.

To configure a Source transformation for a database source, perform the following tasks:

- Configure the source properties.
- If the source includes multiple tables, configure the Source transformation to join the tables. You can join related tables or specify a custom relationship.
- Optionally, configure a custom SQL query to extract source data.
- Optionally, configure the Source transformation to filter or sort source data.
- Ensure that the table and column names do not exceed 74 characters.

Database source properties

Configure properties for database sources such as the database connection, source type, and source objects. You can also specify filter and sort conditions, pre- and post-SQL commands, and whether the output is deterministic or repeatable.

The following table describes the database source properties:

Property	Description
Connection	Name of the source connection.
Source Type	Source type.
Object	Source object for a single source.

Property	Description
Add Source Object	Primary source object for multiple sources.
Add Related Objects	For multiple sources. Displays objects related to the selected source object. Select an object with an existing relationship or click Custom Relationship to create a custom relationship with another object.
Filter	Adds conditions to filter records. Configure a simple or an advanced filter.
Sort	Adds conditions to sort records.
Select Distinct Rows Only	Reads unique rows from the source. Adds SELECT DISTINCT to the SQL query.
Define Query	For a custom query. Displays the Edit Custom Query dialog box. Enter a valid custom query and click OK .
Tracing Level	Detail level of error and status messages that Data Integration writes in the session log. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Pre SQL	SQL command to run against the source before reading data from the source. You can enter a command of up to 5000 characters.
Post SQL	SQL command to run against the source after writing data to the target. You can enter a command of up to 5000 characters.
SQL Query	SQL query to override the default query that Data Integration uses to read data from the source. You can enter an SQL statement supported by the source database.
Output is deterministic	Relational source or transformation output that does not change between mapping runs when the input data is consistent between runs. When you configure this property, the Secure Agent does not stage source data for recovery if transformations in the pipeline always produce repeatable data.
Output is repeatable	Relational source or transformation output that is in the same order between session runs when the order of the input data is consistent. When output is deterministic and output is repeatable, the Secure Agent does not stage source data for recovery.

Related objects

You can configure a Source transformation to join related objects. You can join related objects based on existing relationships or custom relationships. The types of relationships that you can create are based on the connection type.

Use the following relationships to join related objects:

Existing relationships

You can use relationships defined in the source system to join related objects. You can join objects with existing relationships for the following connection types:

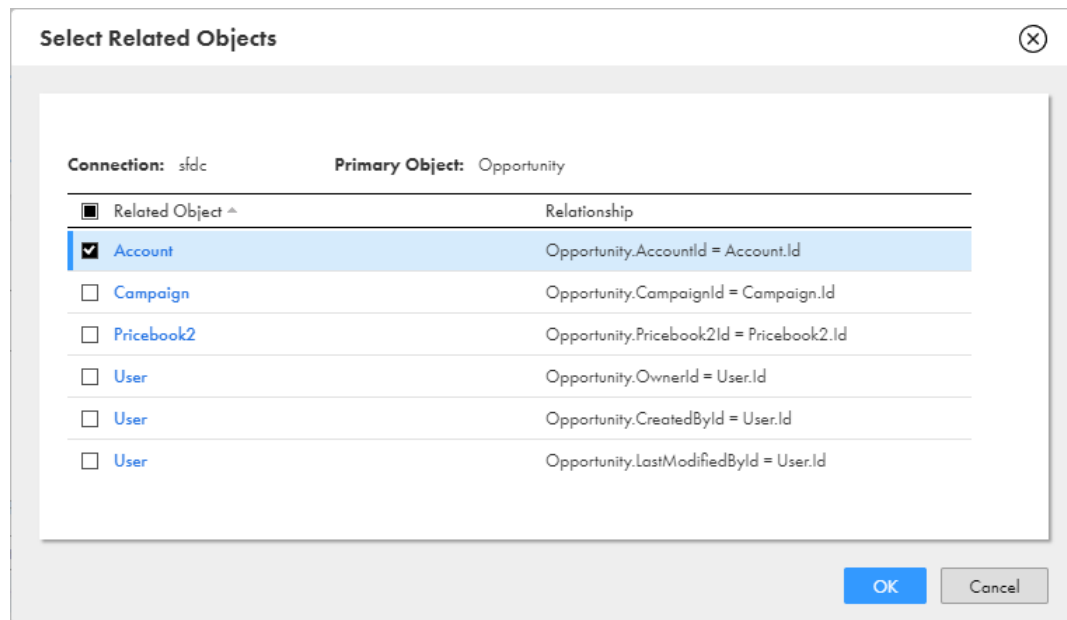
- Database
- Salesforce

- Some Data Integration connectors

To join related objects, you select a primary object. Then you select a related object from a list of related objects.

For example, after you add Opportunity as a primary Salesforce source object, you can add any related objects, such as Account.

The following image shows a list of Salesforce objects with existing relationships with the Opportunity object:



Custom relationships

You can create custom relationships to join objects in the same source system. To create a custom relationship, select a primary object, select another object from the source system, and then select a field from each source to use in the join condition. You must also specify the join type and join operator.

You can select one of the following join types:

Inner

Performs a normal join. Includes rows with matching join conditions. Discards all rows that do not match, based on the condition.

Left

Performs a left outer join. Includes all rows for the source to the left of the join syntax and the rows from both tables that meet the join condition. Discards the unmatched rows from the right source.

Right

Performs a right outer join. Includes all rows for the source to the right of the join syntax and the rows from both tables that meet the join condition. Discards the unmatched rows from the left source.

For example, the following image shows a custom relationship that uses an inner join to join the EMPLOYEE and MANAGER database tables when the EMPLOYEE.E_MANAGERID and MANAGER.M_ID fields match:

Edit Relationship

Select the related objects using: ☐ Existing Relationships ☒ Custom Relationships

Connection: SQL_Server2012 Primary Object: EMPLOYEE ⓘ

Related Object: MANAGER Select...

Configure Relationships

Primary Object Key:
E_MANAGERID ▼

Join Type:
Inner Join ▼

Join Operator:
= (Equals) ▼

Related Object Key:
M_ID ▼

Configured Relationships

Relationship ^	
EMPLOYEE.E_MANAGERID = MAN...	🗑️

Add >

OK Cancel

Joining related objects

You can join related objects in the Source transformation. You can join related objects based on a relationship defined in the source system or a custom relationship defined in the Source transformation.

1. On the **Source** tab, select **Multiple Objects** as the source type.
2. In the **Objects and Relationships** table, select **Add Source Object** from the **Actions** menu.
3. In the **Select Source Object** dialog box, select a source object.
The source object appears in the **Objects and Relationships** table.
4. From the **Actions** menu, select **Add Related Objects**.
5. To join an object based on relationships in the source system, select **Existing Relationships**, select the related object, and then click **OK**.
6. To join an object based on a custom relationship, select **Custom Relationships** and perform the following steps:
 - a. Select the **Related Object** to use in the join.
 - b. Configure the **Primary Object Key** by selecting the primary object field to use in the join.
 - c. Configure the join type.
You can configure an inner, left outer, or right outer join.
 - d. Configure the join operator.
 - e. Configure the **Related Object Key** by selecting the related object field to use in the join.

- f. Click **Add**.
The relationship appears in the **Configured Relationships** table.
 - g. Click **OK**.
7. To join additional sources, select the source to act as the primary source and then repeat steps 4 through 6.

Advanced relationships

You can create an advanced relationship for database sources when the source object in the mapping is configured for multiple sources.

To create an advanced relationship, you add the primary source object in the **Objects and Relationships** table. Then you select fields and write the SQL statement that you want to use. Use an SQL statement that is valid for the source database. You can also add additional objects from the source.

You can also convert a custom relationship to an advanced relationship. To do this, create a custom relationship, and then select **Advanced Relationship** from the menu above the **Objects and Relationships** table. You can edit the relationship that Data Integration creates.

When you create an advanced relationship, the wizard converts any relationships that you defined to an SQL statement that you can edit.

Custom queries

Create a custom query when you want to use a database source that you cannot configure using the single- or multiple-object source options. You might create a custom query to perform a complicated join of multiple tables or to reduce the number of fields that enter the data flow in a very large source.

To use a custom query as a source, select **Query** as the source type, and then click **Define Query**. When you define the query, use SQL that is valid for the source database. You can use database-specific functions in the query.

You can also use a custom query as a lookup source. For information about using a custom query in a Lookup transformation, see [“Custom queries” on page 78](#).

When you create a custom query, enter an SQL SELECT statement to select the source columns you want to use. Data Integration uses the SQL statement to retrieve source column information. You can edit the datatype, precision, or scale of each column before you save the custom query.

For example, you might create a custom query based on a TRANSACTIONS table that includes transactions from 2016 with the following SQL statement:

```
SELECT TRANSACTION_ID, TRANSACTION_TOTAL, TRANSACTION_TIMESTAMP from dbo.TRANSACTIONS WHERE  
TRANSACTION_TIMESTAMP>'0:0:0:0 01/01/2016'
```

Data Integration ensures that custom query column names are unique. If an SQL statement returns a duplicate column name, Data Integration adds a number to the duplicate column name as follows:

```
<column_name><number>
```

When you change a custom query in a saved mapping, at design time Data Integration replaces the field metadata with metadata using the revised query. Typically, this is the desired behavior. However, if the mapping uses a relational source and you want to retain the original metadata, use the **Retain existing field metadata** option. When you use this option, Data Integration doesn't refresh the field metadata during design time. Data Integration maps the existing fields with the fields from the revised query at run time. Fields that can't be mapped will cause run time failure.

Tip: Test the SQL statement you want to use on the source database before you create a custom query. Data Integration does not display specific error messages for invalid SQL statements.

Source filtering and sorting

You can configure the Source transformation to filter or sort data before the data enters the data flow. Use the source query options to filter or sort source data.

Configure the query options on the **Source** tab of the Source transformation. Expand the **Query Options** section, and configure the filter and sort conditions.

You can use the following source query options:

Filter

Filter source data to limit the amount of source data that enters the data flow. You can create the following types of filters:

- **Non-parameterized.** Select the source field and configure the operator and value to use in the filter. When you configure more than one filter, the task applies the filter expressions in the listed order with an AND operator between the filters.
- **Completely parameterized.** Use a parameter for a filter expression and define the filter expression in the task.
- **Advanced.** Create complex expressions that use AND, OR, or nested conditions. The expression that you enter becomes the WHERE clause in the query used to retrieve records from the source. You can use source fields, input and in-out parameters, or system variables in the expression. For example, you can use an input parameter for one of the fields, and select it when the task runs. You can reuse the same parameter in an Expression transformation to create a field expression and also in the Target transformation. Or, you can use an in-out parameter in the expression to retrieve rows that have been updated since the last run.

For more information about system variables, see *Function Reference*. For more information about parameters, see *Mappings*.

You can convert simple non-parameterized data filters to an advanced data filter, but you cannot convert an advanced data filter to simple data filters.

Sort

You can sort source data to provide sorted data to the mapping. For example, you can improve task performance when you provide sorted data to an Aggregator transformation that uses sorted data.

When you sort data, you select one or more source fields. When you select more than one source field, the task sorts the fields in the listed order.

Data in each field is sorted in ascending order. If you want to sort in descending order, you can use the Sorter transformation.

You can use parameters for the sort fields and define the sort fields in the task.

Multibyte hierarchical data

If a mapping includes a transformation that processes hierarchical data and the data uses multibyte characters, you must configure your system to use UTF-8.

On Windows, create the INFA_CODEPAGE=UTF-8 environment variable in Windows System Properties.

On Linux, set the LC_LOCALE variable to UTF-8.

Source fields

You can configure the source fields that you want to use in the data flow. Configure source fields on the **Fields** tab of the **Properties** panel.

Configuration options vary based on the connection type. For most connection types, you can add and remove source fields, configure how the fields are displayed, edit field metadata, and restore original fields from the source object. For some connection types, you can restore original fields from the source object only. For more information about configuring source fields, see the help for the appropriate connector.

You can configure source fields in the following ways:

Add the source file name to each row.

If you are using a file list as the source, and you want to identify the source for each row, add the source file name to the field list. You can pass this information to the target table.

To add the source file name to each row, enable the **Add Currently Processed Filename Field** option. The **Add Currently Processed Filename Field** option is visible for file sources.

When you enable or disable this option, Data Integration adds or removes the `CurrentlyProcessedFileName` field but it doesn't synchronize the fields with the source object. To synchronize with the source object, click the **Refresh** icon. You can synchronize all fields, synchronize new fields only, or skip the synchronization.

Retain existing fields at runtime.

If field metadata changes after a mapping is saved, Data Integration uses the updated field metadata when you run the mapping. Typically, this is the desired behavior. However, if the mapping uses a native flat file connection and you want to retain the metadata used at design time, enable the **Retain existing fields at runtime** option. When you enable this option, Data Integration mapping tasks will use the field metadata that was used when you created the mapping.

Add and remove fields.

You can add fields to a mapping source. Add a field to retrieve a field from the source object that is not displayed in the list. To add a field, click **Add Field**, and then enter the field name, type, precision, and scale.

You can also remove fields that you do not want to use in the mapping. To remove fields, select the fields that you want to remove, and then click **Delete**.

Change the sort order.

You can display source fields in native order, ascending order, or descending order. To change the sort order, click **Sort**, and select the appropriate sort order.

Use technical field names or labels.

You can display field names by label or technical field name.

To change the display option for field names, select **Options > Use Technical Field Names** or **Options > Use Labels**.

Edit field metadata.

You can edit the metadata for a field. You might edit metadata to change information that is incorrectly inferred. When you edit metadata, you can change the name, native type, native precision, and native scale, if applicable for the data type. For some source types, you can also change the transformation data type in the **Type** column.

To edit the name or metadata for one or more fields, click **Options > Edit Metadata**. When you edit metadata, you can also display native names by label or technical field name. To change the display option for native names, select **Options > Show Technical Field Names** or **Options > Show Labels**.

When you change the metadata for a field, avoid making changes that can cause errors when you run the task. For example, you can usually increase the native precision or native scale of a field without causing errors. But if you reduce the precision of a field, you might cause the truncation of data.

Restore original fields from the source object.

To restore the original fields from the source object, enable the **Synchronize** option. When you synchronize fields, Data Integration restores deleted source fields and adds fields that are new to the source. Data Integration removes any added fields that do not have corresponding fields in the source object.

Data Integration updates the metadata for existing source fields based on whether you synchronize all fields or synchronize new fields only. When you synchronize all fields, Data Integration replaces any field metadata that you edited with the field metadata from the source object. When you synchronize new fields only, Data Integration retains the metadata for any existing source field. Data Integration does not revert changes that you made to the **Name** field.

Editing transformation data types

When Data Integration reads source data, it converts the native data types to comparable transformation data types before it transforms the data. When Data Integration writes to a target, it converts the transformation data types to the comparable native data types. When you edit source metadata, you can sometimes change the transformation data type for a field.

You can change the transformation data type for connectors in which the native data type has multiple corresponding transformation data types.

To change the transformation data type, edit the metadata for the source, and select the appropriate transformation data type in the **Type** column.

When you edit the transformation data type for a field, Data Integration updates the data type for the field in the downstream transformations. It also updates the data type for the field in the target if the target is created at runtime. If the mapping contains an existing target, you might need to edit the field metadata in the target to ensure that the data types are compatible.

For more information about editing transformation data types for different source types, see the help for the appropriate connector.

CHAPTER 3

Target transformation

Use the Target transformation to define the target connection and target object for the mapping. You can use one or more Target transformations in a mapping.

Based on the connection type, you can define advanced target options, specify to use a new or existing target object, or configure update columns for the target. The target options that appear also depend on the connection type that you select. For example, when you select a Salesforce connection, you can configure success and error log details.

The following properties are available for the Target transformation:

- General. Defines the transformation name and a description.
- Incoming Fields. Includes the field rules that define the data written to the target. Allows a preview of target fields.
- Target. Defines the target connection, target object, and advanced options. Based on the connection type, you can create a new target, use an existing target, or configure update columns.
- Target Fields. Lists the fields in the target objects. Optionally add or remove fields. You can also edit target field metadata.
- Field Mapping. Defines the field mappings from the upstream transformation to the target. Field mapping is only applicable when using an existing target object.

Target object

Select the target object for the Target transformation on the **Target** tab of the Properties panel.

The properties that you configure for the target object vary based on the connection type and the mapping type.

The following image shows the **Target** tab for an Amazon Redshift target:

The screenshot shows the 'Target' tab in a configuration window. The left sidebar has tabs for 'General', 'Incoming Fields', 'Target' (selected), 'Target Fields', and 'Field Mapping'. The main area is divided into 'Details' and 'Advanced' sections. In the 'Details' section, there are four rows of configuration: 'Connection' with a dropdown menu showing 'Sample Redshift Connection (MockConn...)' and buttons 'View...', 'New Connection...', and 'New Parameter...'; 'Target Type' with a dropdown menu showing 'Single Object'; 'Object' with a text field containing 'CUSTOMERS' and buttons 'Select...' and 'Preview Data...'; and 'Operation' with a dropdown menu showing 'Insert'. The 'Advanced' section has two text input fields for 'Success File Directory' and 'Error File Directory', and a checkbox labeled 'Forward Rejected Rows' which is checked. A small information icon is next to the checkbox.

1. Target details where you configure the target connection, target type, target object, and target operation.

You can select a target object in the following ways:

Select the connection and target object.

In the **Details** area, select the target connection, target type, and target object. You can create a new connection. For flat file and relational targets, you can also create the target object at run time.

If you use an existing target object, select the target from a list of target objects and link the target to the upstream transformation. If the target table changes, you must update the Target transformation to match it. If the physical target and the Target transformation do not match, the mapping fails.

If you use an existing target object for a flat file target, the existing target is overwritten when you run the mapping task.

Use a parameter.

You can use input parameters to define the target connection and target object when you run the mapping task. For more information about parameters, see *Mappings*.

You must also define the target operation. The available target operations vary based on the connection type.

Database targets

Database targets include relational sources such as Oracle, MySQL, and Microsoft SQL Server.

When you configure a Target transformation for a database target, you can write data to a single target table. You can select an existing table or create the table at run time.

Ensure that the table and column names do not exceed 74 characters.

Database target properties

You configure database target properties on the **Target** tab of the Properties panel.

The following table describes the database target properties:

Property	Description
Connection	Name of the target connection. Alternatively, you can define a parameter, and then specify the connection in the mapping task.
Target Type	Target type, either single object or parameter.
Object	Name of the target object. If you select a single object, you can also preview the data.
Operation	Target operation, either insert, update, upsert, delete, or data driven.
Truncate Target	Truncates the target object before inserting new rows. Applies to insert and data driven operations.
Enable Target Bulk Load	Uses the database bulk API to perform an insert operation. Use the bulk API to write large amounts of data to the database with a minimal number of API calls. Loading in bulk mode can improve performance, but it limits the ability to recover because no database logging occurs. Applies to insert operations.
Update Columns	The fields to use as temporary primary key columns when you update, upsert, or delete target data. When you select more than one update column, the mapping task uses the AND operator with the update columns to identify matching rows. Applies to update, upsert, delete and data driven operations.
Data Driven Condition	Enables you to define expressions that flag rows for an insert, update, delete, or reject operation. For example, the following IIF statement flags a row for reject if the ID field is null. Otherwise, it flags the row for update: <pre>IIF (ISNULL(ID), DD_REJECT, DD_UPDATE)</pre> Applies to the data driven operation.
Forward Rejected Rows	Causes the mapping task to forward rejected rows to the reject file. If you do not forward rejected rows, the mapping task drops rejected rows and writes them to the session log. If you enable row error handling, the mapping task writes the rejected rows and the dropped rows to the row error logs. It does not generate a reject file. If you want to write the dropped rows to the session log in addition to the row error logs, you can enable verbose data tracing.
Pre SQL	SQL command to run against the target before reading data from the source. You can enter a command of up to 5000 characters.
Post SQL	SQL command to run against the target after writing data to the target. You can enter a command of up to 5000 characters.

Property	Description
Update Override	<p>Overrides the default UPDATE statement for the target.</p> <p>Enter the update statement. Alternatively, click Configure to generate the default UPDATE statement, and then modify the default statement.</p> <p>The UPDATE statement that you enter overrides the default UPDATE statement that Data Integration uses to update targets based on key columns. You can define an override UPDATE statement to update target tables based on non-key columns.</p>
Reject File Directory	<p>Directory path to write the reject file. By default, the mapping task writes all reject files to the following service process variable directory:</p> <pre>\$PMBadFileDir/<federated task ID></pre> <p>If you specify both the directory and file name in the Reject File Name field, clear this field. The mapping task concatenates this field with the Reject File Name field when it runs the task.</p>
Reject File Name	<p>File name, or file name and path of the reject file. By default, the mapping task names the reject file after the target object name: <target name>.bad.</p> <p>The mapping task concatenates this field with the Reject File Directory field when it runs the task. For example, if you have C:\reject_file\ in the Reject File Directory field, and enter filename.bad in the Reject File Name field, the mapping task writes rejected rows to C:\reject_file\filename.bad.</p>

For more information about database target properties, see the help for the appropriate connector.

Database targets created at run time

If a mapping includes a database target, you can select an existing target table or create the target table at run time. When you create a database target at run time, Data Integration automatically discovers the target object metadata for data type, precision, and scale, based on the data source.

If you need to edit target object metadata, you can edit it in the Source transformation.

You cannot link the target fields to the upstream transformation. If you want to reduce the number of unused fields in the target, configure field rules in the Target transformation or in the upstream transformations.

When you create a database target at run time, the mapping task creates the database table the first time the mapping runs based on the fields from the upstream transformation.

In subsequent runs, the mapping task replaces the data in the target table that was created in the initial run. Consequently, if you change the mapping after the initial run, in subsequent runs the target will not reflect changes to the number of target fields and its metadata. To see the changes, you can either delete the existing target before you run the mapping or change the name of the target.

If you create a relational target at run time, the target operation is always insert. You can choose to truncate the target.

Creating a database target at run time

To create a database target at run time, select **Create New at Runtime** in the **Target Object** dialog box and enter the target table name.

1. On the **Target** tab of the Target transformation, select a database connection.
2. Set the target type to **Single Object**.
3. Click **Select** to select the target object.

4. In the **Target Object** dialog box, select **Create New at Runtime**.
5. Enter the target table name.
6. Click **OK**.

Update columns for relational targets

You can configure one or more fields as update columns in relational targets. Update columns are columns that uniquely identify rows in the target table. The mapping task uses them to update, upsert, or delete data in the target.

Configure update columns when the target table does not contain a primary key and the mapping uses an update, upsert, or delete operation. When you select more than one update column, the mapping uses the AND operator with the update columns to identify matching rows.

When you run the mapping, it uses the field mapping to match rows in the upstream transformations to the target table. If the mapping task matches an incoming row to multiple target rows, it performs the specified task operation on all matched target rows.

When you use a parameter for the target connection or target object, you can configure update columns in the mapping task.

Configuring update columns

You can configure update columns when you use the update or upsert operation to update data in a relational target.

1. In the Properties panel, click the **Target** tab.
2. Select a relational connection.
You can also use a connection parameter for a relational database connection type.
3. Select the target type that you want to use.
4. Select a target object.
5. Select the update or upsert operation.
6. To select update columns, click **Add**.
The **Update Columns** window displays all target columns.
7. Move the fields that you want to use from the **Target Columns** list to the **Update Columns** list.
8. Click **OK**.

Target update override

By default, Data Integration updates target tables based on key values. However, you can override the default UPDATE statement for each target in a mapping. You might want to update the target based on non-key columns.

You can enter a target update override for relational and ODBC connections. For more information, see the help for the appropriate connector.

Override the UPDATE statement in the Target transformation advanced properties. Enter the target UPDATE statement in the **Update Override** field. Alternatively, click **Configure** to generate the default UPDATE statement and then modify the statement.

Because the target fields must match the target column names, the update statement includes the keyword :TU to specify the fields in the target transformation. If you modify the UPDATE portion of the statement, you must use :TU to specify fields.

When you override the default UPDATE statement, you must enter an SQL statement that is valid for the database. Data Integration does not validate the syntax.

Example

A mapping passes the total sales for each salesperson to the T_SALES table.

Data Integration generates the following default UPDATE statement for the target T_SALES:

```
UPDATE
  T_SALES
SET
  EMP_NAME = :TU.EMP_NAME,
  DATE_SHIPPED = :TU.DATE_SHIPPED,
  TOTAL_SALES = :TU.TOTAL_SALES
WHERE
  EMP_ID = :TU.EMP_ID
```

You want to override the WHERE clause to update records for employees named Mike Smith only. To do this, you edit the WHERE clause as follows:

```
UPDATE
  T_SALES
SET
  DATE_SHIPPED = :TU.DATE_SHIPPED,
  TOTAL_SALES = :TU.TOTAL_SALES
WHERE
  :TU.EMP_NAME = EMP_NAME AND EMP_NAME = 'MIKE SMITH'
```

Guidelines for configuring the target update override

Use the following guidelines when you enter target update queries:

- If you use target update override, you must manually put all database reserved words in quotes.
- You cannot override the default UPDATE statement if a target column name contains any of the following characters:

```
' , ( ) < > = + - * / \ t \ n \ 0 <space>
```

- If you update an individual row in the target table more than once, the database only has data from the last update. If the mapping does not define an order for the result data, different runs of the mapping on identical input data may result in different data in the target table.
- A WHERE clause that does not contain any column references updates all rows in the target table, or no rows in the target table, depending on the WHERE clause and the data from the mapping. For example, the following query sets the EMP_NAME to "MIKE SMITH" for all rows in the target table if any row of the transformation has EMP_ID > 100:

```
UPDATE T_SALES SET EMP_NAME = 'MIKE SMITH' WHERE :TU.EMP_ID > 100
```

- If the WHERE clause contains no field references, the mapping updates the same set of rows for each row of the mapping. For example, the following query updates all employees with EMP_ID > 100 to have the EMP_NAME from the last row in the mapping:

```
UPDATE T_SALES SET EMP_NAME = :TU.EMP_NAME WHERE EMP_ID > 100
```

- If you set the target operation to update or upsert, configure the mapping task to treat source rows as update in the advanced session properties.

Entering a target update statement

Enter a target update statement on the **Target** tab of the Target transformation.

1. On the **Target** tab of the Target transformation, open the advanced properties.
2. Click **Configure** next to the **Update Override** field.
3. In the **Update Override SQL Editor**, click **Generate SQL**.

The default UPDATE statement appears.

4. Modify the UPDATE statement.

Tip: Click **Format SQL** to format the UPDATE statement for easier readability.

You can override the WHERE clause to include non-key columns. Enclose all database reserved words in quotes.

5. Click **OK**.

Multibyte hierarchical data

If a mapping includes a transformation that processes hierarchical data and the data uses multibyte characters, you must configure your system to use UTF-8.

On Windows, create the INFA_CODEPAGE=UTF-8 environment variable in Windows System Properties.

On Linux, set the LC_LOCALE variable to UTF-8.

Target fields

You can configure the target fields that you want to use in the data flow. You can add and remove target fields, configure how the fields are displayed, edit field metadata, and restore original fields from the target object.

Configure target fields on the **Target Fields** tab of the **Properties** panel.

You can configure target fields in the following ways:

Add and remove fields.

You can add fields to a mapping target. To add a field, click **Add Field**, and then enter the field name, type, precision, and scale.

You can also remove fields that you do not want to use in the mapping. To remove fields, select the fields that you want to remove, and then click **Delete**.

Change the sort order.

You can display target fields in native order, ascending order, or descending order. To change the sort order, click **Sort**, and select the appropriate sort order.

Use technical field names or labels.

You can display field names by label or technical field name.

To change the display option for field names, select **Options > Use Technical Field Names** or **Options > Use Labels**.

Edit field metadata.

You can edit the metadata for a field. You might edit metadata to change information that is incorrectly inferred. When you edit metadata, you can change the name, native type, native precision, and native scale, if applicable for the data type.

To edit the name or metadata for one or more fields, click **Options > Edit Metadata**. When you edit metadata, you can also display native names by label or technical field name. To change the display option for native names, select **Options > Show Technical Field Names** or **Options > Show Labels**.

When you change the metadata for a field, avoid making changes that can cause errors when you run the task. For example, you can usually increase the native precision or native scale of a field without causing errors. But if you reduce the precision of a field, you might cause the truncation of data.

Restore original fields from the target object.

To restore the original fields from the target object, use the **Synchronize** option. When you synchronize fields, Data Integration restores deleted target fields, reverts data type and precision changes, and adds fields that are new to the target. Data Integration removes any added fields that do not have corresponding fields in the target object.

For existing target fields, Data Integration replaces any metadata that you edited with the field metadata from the target object. Data Integration does not revert changes that you made to the **Name** field.

Target transformation field mappings

Configure field mappings in a Target transformation to define how data moves from the data flow to the target object.

Configure field mappings on the **Field Mappings** tab.

The **Field Mappings** tab includes a list of incoming fields and a list of target fields.

You can configure the following field mapping options:

Field Map Options

Method of mapping incoming fields to target fields. Select one of the following options:

- **Manual.** Manually link incoming fields to target fields. Selecting this option removes links for automatically mapped fields. To map fields manually, drag a field from the incoming fields list and position it next to the appropriate field in the target fields list. Or, you can map selected fields, unmap selected fields, or clear all of the mappings using the **Actions** menu.
- **Automatic.** Automatically link fields with the same name. Use when all of the fields that you want to link share the same name. You cannot manually link fields with this option.
- **Completely Parameterized.** Use a parameter to represent the field mapping. In the task, you can configure all field mappings.
- **Partially Parameterized.** Configure links in the mapping that you want to enforce and use a parameter to allow other fields to be mapped in the mapping task. Or, use a parameter to configure links in the mapping, and allow all fields and links to display in the task for configuration.
To see more information on field mapping parameters, see *Mappings*.

Options

You can configure how the fields display and which fields to display. To do so, click **Options** and select from the following display options:

- Display fields using field labels or technical field names.
- Show mapped, unmapped, or all fields.

Automap

If you want Data Integration to automatically link fields with the same name and you also want to manually map fields, select the **Manual** option and open the **Automap** menu.

You can map fields in the following ways:

- **Exact Field Name.** Data Integration matches fields of the same name.
- **Smart Map.** Data Integration matches fields with similar names. For example, if you have an incoming field `Cust_Name` and a target field `Customer_Name`, Data Integration automatically links the `Cust_Name` field with the `Customer_Name` field.

You can use both Exact Field Name and Smart Map in the same field mapping. For example, use Exact Field Name to match fields with the same name and then use Smart Map to map fields with similar names.

You can undo all automapped field mappings by clicking **Automap > Undo Automap**. To unmap a single field, select the field to unmap and click **Actions > Unmap**.

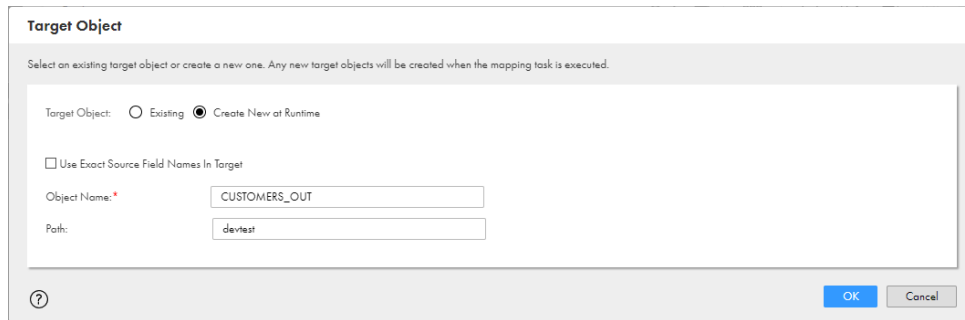
Data Integration highlights newly mapped fields. For example, when you use Exact Field Name, Data Integration highlights the mapped fields. If you then use Smart Map, Data Integration only highlights the fields mapped using Smart Map.

Configuring a Target transformation

You can use an existing target or create a target to hold the results of a mapping. If you choose to create the target, the Secure Agent creates the target when you run the task.

1. Select the Target transformation in the mapping.
2. On the **General** tab, enter a target name and optional description
3. On the **Incoming Fields** tab, configure field rules that specify the fields to include in the target
For more information about field rules, see ["Field rules" on page 13](#).
4. On the **Target** tab, select the target connection and specify the target type, for example, **Single Object** or **Parameter**.
The option to select multiple objects is available for NetSuite connections only.
5. To use an input parameter for the target object, select an existing parameter, or click **New Parameter** and create a new parameter for the target object.
6. To use an existing target object, either enter the object name or click **Select** and select the target object.
7. To create a new target object, perform the following steps:
 - a. Click **Select**.

- b. In the **Target Object** dialog box, select **Create New at Runtime**:

The image shows a 'Target Object' dialog box. At the top, it says 'Select an existing target object or create a new one. Any new target objects will be created when the mapping task is executed.' Below this, there are two radio buttons: 'Existing' and 'Create New at Runtime'. The 'Create New at Runtime' option is selected. Underneath, there is a checkbox labeled 'Use Exact Source Field Names In Target' which is currently unchecked. Below the checkbox, there are two text input fields. The first is labeled 'Object Name:' and contains the text 'CUSTOMERS_OUT'. The second is labeled 'Path:' and contains the text 'devtest'. At the bottom right of the dialog box, there are two buttons: 'OK' and 'Cancel'. A help icon (?) is located at the bottom left.

- c. Optionally, choose to use the exact source field names in the new target object.
- d. Enter the object name.
- e. Enter the path to the new target object, if required.
- For information about the target properties for different connection types, see the help for the appropriate connector.
- f. Click **OK**.
8. Select the target operation and related properties such as the update columns.
9. Specify advanced properties for the target, if required.
- Advanced properties vary based on the connection type. For information about connector properties, see the help for the appropriate connector.
10. Configure the target fields on the **Target Fields** tab.
- You can edit field names and metadata, add fields, and delete unnecessary fields.
- If you create the target at run time, you cannot configure target fields.
11. Map incoming fields to target fields on the **Field Mapping** tab.
- If you create the target at run time, fields are mapped automatically.
- For more information about field mapping, see [“Target transformation field mappings” on page 53](#).

CHAPTER 4

Aggregator transformation

Configure an Aggregator transformation to perform aggregate calculations, such as averages and sums, against groups of data. You can use an Aggregator transformation to remove duplicate rows.

The Aggregator transformation behaves like the Expression transformation except you can configure the Aggregator transformation to perform calculations on a group of data. The Expression transformation returns results on a row-by-row basis.

For example, you can use the Aggregator transformation to calculate the average salary for employees in each department of an organization. In the Aggregator transformation, create a group for the department number and then configure an expression to calculate the average salary for the employees in each group.

Group by fields

Use group by fields to define how to group data for aggregate expressions. Configure group by fields on the **Group By** tab of the **Properties** panel.

To define a group for the aggregate expression, select the appropriate input, input/output, and output fields in the Aggregator transformation. You can select multiple group by fields to create a new group for each unique combination. Data Integration then performs the defined aggregation for each group.

When you group values, Data Integration produces one row for each group. If you do not group values, Data Integration returns one row for all input rows.

When you select multiple group by fields in the Aggregator transformation, Data Integration uses field order to determine the order by which it groups. The group order can affect the results. Order the group by fields to ensure the appropriate grouping. You can change the field order after you select the fields in the group.

For example, you create aggregate fields called TOTAL_QTY and TOTAL_PRICE to store the total quantity and total price for each item by store. You define the following expressions for each field:

- TOTAL_QTY: SUM (QTY)
- TOTAL_PRICE: SUM (QTY*PRICE)

You define STORE_ID and ITEM as the group by fields.

The input rows contain the following data:

STORE_ID	ITEM	QTY	PRICE
101	'battery'	3	2.99
101	'battery'	1	3.19

STORE_ID	ITEM	QTY	PRICE
101	'battery'	2	2.59
101	'AAA'	2	2.45
201	'battery'	1	1.99
201	'battery'	4	1.59
301	'battery'	1	2.45

Data Integration performs the aggregate calculations on the following unique groups:

STORE_ID	ITEM
101	'battery'
101	'AAA'
201	'battery'
301	'battery'

Data Integration returns the store ID, item name, total quantity for each item by store, and total price for each item by store:

STORE_ID	ITEM	TOTAL_QTY	TOTAL_PRICE
101	'AAA'	2	4.90
101	'battery'	6	17.34
201	'battery'	5	8.35
301	'battery'	1	2.45

Sorted data

To improve job performance, you can configure an Aggregator transformation to use sorted data. To configure the Aggregator transformation to process sorted data, on the **Advanced** tab, select **Sorted Input**.

When you configure an Aggregator transformation to use sorted data, you must sort data earlier in the data flow. If the Aggregator transformation processes data from a relational database, you must also ensure that the sort keys in the source are unique. If the data is not presorted correctly or the sort keys are not unique, you can receive unexpected results or errors when you run the mapping task.

When the mapping task performs aggregate calculations on sorted data, the task caches sequential rows of the same group. When the task reads data for different group, it performs aggregate calculations for the cached group, and then continues with the next group.

For example, an Aggregator transformation has the STORE_ID and ITEM group by fields, with the sorted input option selected. When you pass the following data through the Aggregator, the mapping task performs an aggregation for the three rows in the 101/battery group as soon as it finds the new group, 201/battery:

STORE_ID	ITEM	QTY	PRICE
101	'battery'	3	2.99
101	'battery'	1	3.19
101	'battery'	2	2.59
201	'battery'	4	1.59
201	'battery'	1	1.99

When you do not use sorted data, the mapping task performs aggregate calculations after it reads all data.

Aggregate fields

Use an aggregate field to define aggregate calculations.

When you configure an Aggregator transformation, create an aggregate field for the output of each calculation that you want to use in the data flow. You can use aggregate functions in aggregate fields. You can also use conditional clauses and nonaggregate functions.

Configure aggregate fields on the **Aggregate** tab of the **Properties** panel. When you configure an aggregate field, you define the field name, data type, precision, scale, and optional description. The description can contain up to 4000 characters. You also define the calculations that you want to perform.

When you configure aggregate fields, you can use variable fields for calculations that you want to use within the transformation. You can also include macros in aggregate and variable fields.

Aggregate functions

You can use aggregate functions in expressions in aggregate fields.

For more information about aggregate functions, see *Function Reference*.

Nested aggregate functions

A nested aggregate function is an aggregate function within another aggregate function.

For example, the following expression sums sales and returns the highest number:

```
MAX( SUM( SALES ) )
```

You can include multiple single-level or multiple nested functions in different output fields in an Aggregator transformation. You cannot include both single-level and nested functions in an Aggregator transformation.

When an Aggregator transformation contains a single-level function in any output field, you cannot use a nested function in any other field in that transformation. If you need to create both single-level and nested functions, create separate Aggregator transformations.

Conditional clauses

Use conditional clauses in the aggregate expression to reduce the number of rows used in the aggregation. The conditional clause can be any clause that evaluates to TRUE or FALSE.

For example, use the following expression to calculate the total commissions of employees who exceeded their quarterly quota:

```
SUM( COMMISSION, COMMISSION > QUOTA )
```

Advanced properties

You can configure advanced properties for an Aggregator transformation. The advanced properties control settings such as the tracing level for session log messages, whether the transformation uses sorted input, cache settings, and whether the transformation is optional or required.

You can configure the following properties:

Property	Description
Tracing Level	Detail level of error and status messages that Data Integration writes in the session log. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Sorted Input	Indicates that input data is pre-sorted by groups. Select this option only if the mapping passes sorted data to the Aggregator transformation.
Cache Directory	Local directory where Data Integration creates the index and data cache files. By default, Data Integration uses the directory entered in the Secure Agent \$PMCacheDir property for the Data Integration Server. If you enter a new directory, make sure that the directory exists and contains enough disk space for the aggregate caches.
Data Cache Size	Data cache size for the transformation. Select one of the following options: <ul style="list-style-type: none">- Auto. Data Integration sets the cache size automatically. If you select Auto, you can also configure a maximum amount of memory for Data Integration to allocate to the cache.- Value. Enter the cache size in bytes. Default is Auto.
Index Cache Size	Index cache size for the transformation. Select one of the following options: <ul style="list-style-type: none">- Auto. Data Integration sets the cache size automatically. If you select Auto, you can also configure a maximum amount of memory for Data Integration to allocate to the cache.- Value. Enter the cache size in bytes. Default is Auto.

Property	Description
Transformation Scope	<p>Specifies how Data Integration applies the transformation logic to incoming data. Select one of the following options:</p> <ul style="list-style-type: none"> - Transaction. Applies the transformation logic to all rows in a transaction. Choose Transaction when a row of data depends on all rows in the same transaction, but does not depend on rows in other transactions. - All Input. Applies the transformation logic on all incoming data. When you choose All Input, Data Integration drops incoming transaction boundaries. Choose All Input when a row of data depends on all rows in the source.
Optional	<p>Determines whether the transformation is optional. If a transformation is optional and there are no incoming fields, the mapping task can run and the data can go through another branch in the data flow. If a transformation is required and there are no incoming fields, the task fails.</p> <p>For example, you configure a parameter for the source connection. In one branch of the data flow, you add a transformation with a field rule so that only Date/Time data enters the transformation, and you specify that the transformation is optional. When you configure the mapping task, you select a source that does not have Date/Time data. The mapping task ignores the branch with the optional transformation, and the data flow continues through another branch of the mapping.</p>

CHAPTER 5

Expression transformation

The Expression transformation calculates values within a single row. Use the Expression transformation to perform non-aggregate calculations.

For example, you might use an Expression transformation to adjust bonus percentages or to concatenate first and last names.

When you configure an Expression transformation, create an expression field for the output of each calculation that you want to use in the data flow. Create a variable field for calculations that you want to use within the transformation.

Expression fields

An expression field defines the calculations to perform on an incoming field and acts as the output field for the results. You can use as many expression fields as necessary to perform calculations on incoming fields.

When you configure an expression field, you define different fields based on the field type, for example the field name, data type, precision, scale, default value, and optional description. The description can contain up to 4000 characters. You also define the calculations that you want to perform.

You cannot specify an expression decimal field precision value greater than 38.

Effect of the default value

The default value tells the mapping task what to do when the transformation encounters output errors. The default value is not available in mappings in advanced mode.

You can set one of the following values for output fields:

- ERROR('transformation error'). System default. When a transformation error occurs, the mapping task skips the row and writes the error to the session log or row error log.
- A constant or constant expression. The mapping task replaces the error with the constant or constant expression. Nothing is written to the logs.
- ABORT. Transformation aborts and the mapping task writes a message to the session log.

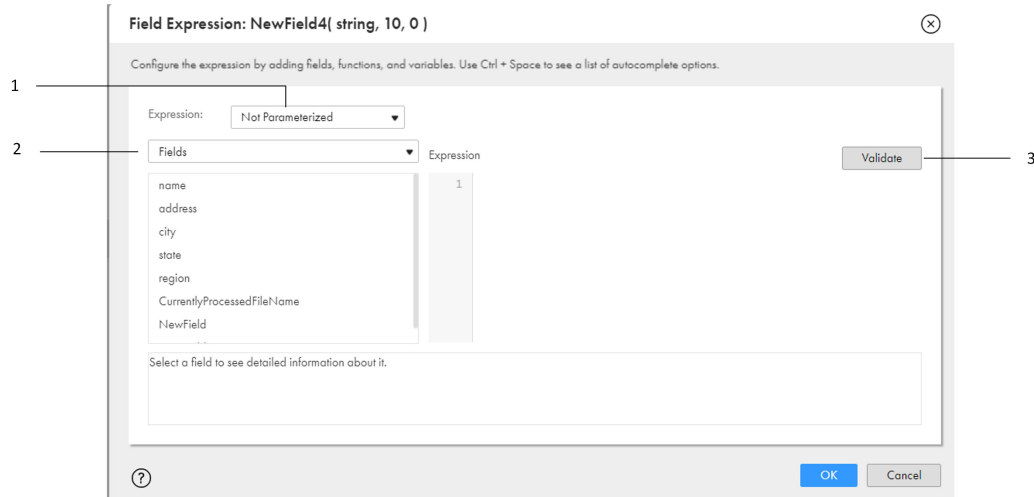
Data Integration validates the output field default value when you save or validate the mapping. If you enter an invalid default value, the Mapping Designer marks the mapping as not valid.

Expression editor

Use the expression editor to configure the expression field. The expression can contain constants, variables, built-in functions, and user-defined functions. You can also create a complex expression by nesting functions within functions.

To configure an expression, enter it in the expression editor.

The following image shows the expression editor and the actions you can perform:



1. Parameterize the expression.
2. Switch between fields, system variables, parameters, built-in functions, and user defined functions.
3. Validate the expression.

You can add source fields, functions, and variables to the expression by clicking **Add** next to the object that you want to use. You can also type in the expression manually.

Alternatively, press the **Ctrl + Space** keys to see a list of recommended arguments and functions in-line. Data Integration provides recommendations based on the type of function arguments and keystrokes. In-line recommendations are not available for hierarchical source data.

To validate the expression, click **Validate**. Data Integration validates the expression.

Transformation language components for expressions

The transformation language includes the following components to create simple or complex expressions:

- Fields. Use the name of a source field to refer to the value of the field.
- Literals. Use numeric or string literals to refer to specific values.
- Functions. Use these SQL-like functions to change data in a task.
- Operators. Use transformation operators to create expressions to perform mathematical computations, combine data, or compare data.
- Constants. Use the predefined constants to reference values that remain constant, such as TRUE.

Expression syntax

You can create a simple expression that only contains a field, such as `ORDERS`, or a numeric literal, such as `10`. You can also write complex expressions that include functions nested within functions, or combine different fields using the transformation language operators.

Note: Although the transformation language is based on standard SQL, there are differences between the two languages.

String and numeric literals

You can include numeric or string literals.

Enclose string literals within single quotation marks. For example:

```
'Alice Davis'
```

String literals are case sensitive and can contain any character except a single quotation mark. For example, the following string is not allowed:

```
'Joan's car'
```

To return a string containing a single quotation mark, use the `CHR` function:

```
'Joan' || CHR(39) || 's car'
```

Do not use single quotation marks with numeric literals. Just enter the number you want to include. For example:

```
.05
```

or

```
$$Sales_Tax
```

Adding comments to expressions

You can use the following comment specifiers to insert comments in expressions:

- Two dashes:

```
-- These are comments
```

- Two forward slashes:

```
// These are comments
```

Data integration tasks ignore all text on a line preceded by comment specifiers. For example, to concatenate two strings, enter the following expression with comments in the middle of the expression:

```
-- This expression concatenates first and last names for customers:  
FIRST_NAME -- First names from the CUST table  
|| // Concat symbol  
LAST_NAME // Last names from the CUST table  
// Joe Smith Aug 18 1998
```

Data integration tasks ignore the comments and evaluates the expression as follows:

```
FIRST_NAME || LAST_NAME
```

You cannot continue a comment to a new line:

```
-- This expression concatenates first and last names for customers:
FIRST_NAME -- First names from the CUST table
|| // Concat symbol
LAST_NAME // Last names from the CUST table
Joe Smith Aug 18 1998
```

In this case, data integration tasks do not validate the expression because the last line is not a valid expression.

Reserved words

Some keywords, such as constants, operators, and system variables, are reserved for specific functions. These include:

- :EXT
- :INFA
- :LKP
- :MCR
- :SD
- :SEQ
- :SP
- :TD
- AND
- DD_DELETE
- DD_INSERT
- DD_REJECT
- DD_UPDATE
- FALSE
- NOT
- NULL
- OR
- PROC_RESULT
- SPOUTPUT
- TRUE
- WORKFLOWSTARTTIME

The following words are reserved for Informatica Intelligent Cloud Services:

- ABORTED
- DISABLED
- FAILED
- NOTSTARTED
- STARTED

- STOPPED
- SUCCEEDED

Note: You cannot use a reserved word to name a field. Reserved words have predefined meanings in expressions.

Advanced properties

You can configure advanced properties for an Expression transformation. The advanced properties control settings such as the tracing level for session log messages and whether the transformation is optional or required.

You can configure the following properties:

Property	Description
Tracing Level	Detail level of error and status messages that Data Integration writes in the session log. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Optional	<p>Determines whether the transformation is optional. If a transformation is optional and there are no incoming fields, the mapping task can run and the data can go through another branch in the data flow. If a transformation is required and there are no incoming fields, the task fails.</p> <p>For example, you configure a parameter for the source connection. In one branch of the data flow, you add a transformation with a field rule so that only Date/Time data enters the transformation, and you specify that the transformation is optional. When you configure the mapping task, you select a source that does not have Date/Time data. The mapping task ignores the branch with the optional transformation, and the data flow continues through another branch of the mapping.</p>

CHAPTER 6

Filter transformation

The Filter transformation filters data out of the data flow based on a specified filter condition. To improve job performance, place the Filter transformation close to mapping sources to remove unnecessary data from the data flow.

A filter condition is an expression that returns TRUE or FALSE. When the filter condition returns TRUE for a row, the Filter transformation passes the row to the rest of the data flow. When the filter condition returns FALSE, the Filter transformation drops the row.

You can filter data based on one or more conditions. For example, to work with data within a data range, you can create conditions to remove data before and after specified dates.

Link a single transformation to the Filter transformation. You cannot merge transformations into the Filter transformation.

Filter conditions

The filter condition is an expression that returns TRUE or FALSE.

You can create one or more simple filter conditions. A simple filter condition includes a field name, operator, and value. For example, `Sales > 0` retains rows where all sales values are greater than zero.

Filter conditions are case sensitive. You can use the following operators in a simple filter:

- = (equals)
- < (less than)
- > (greater than)
- < = (less than or equal to)
- > = (greater than or equal to)
- ! = (not equals)

When you define more than one simple filter condition, the mapping task evaluates the conditions in the order that you specify. The task evaluates the filter conditions using the AND logical operator to join the conditions. The task returns rows that match all of the filter conditions.

You can use an advanced filter condition to define a complex filter condition. When you configure an advanced filter condition, you can incorporate multiple conditions using the AND or OR logical operators. You can use a constant to represent condition results: 0 is the equivalent of FALSE, and any non-zero value is the equivalent of TRUE.

When you change the filter condition type from simple to advanced, the Mapping Designer includes configured simple filter conditions in the advanced filter condition. You can use or delete the simple filter conditions. The conversion does not include parameters.

To filter rows that contain null values, use the ISNULL function to test the value of the field. To filter rows that contain spaces, use IS_SPACES.

For example, if you want to filter out rows that contain a null value in the First_Name field, use the following condition: IIF(ISNULL(First_Name),FALSE,TRUE). The condition states that if the First_Name field is NULL, the return value is FALSE. The mapping task discards the row. Otherwise, the row passes through to the next transformation.

Advanced properties

You can configure advanced properties for a Filter transformation. The advanced properties control settings such as the tracing level for session log messages and whether the transformation is optional or required.

You can configure the following properties:

Property	Description
Tracing Level	Detail level of error and status messages that Data Integration writes in the session log. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Optional	<p>Determines whether the transformation is optional. If a transformation is optional and there are no incoming fields, the mapping task can run and the data can go through another branch in the data flow. If a transformation is required and there are no incoming fields, the task fails.</p> <p>For example, you configure a parameter for the source connection. In one branch of the data flow, you add a transformation with a field rule so that only Date/Time data enters the transformation, and you specify that the transformation is optional. When you configure the mapping task, you select a source that does not have Date/Time data. The mapping task ignores the branch with the optional transformation, and the data flow continues through another branch of the mapping.</p>

CHAPTER 7

Input transformation

The Input transformation is a passive transformation that you use to configure the data that you want to pass into a mapplet. Use an Input transformation when you want a mapplet to receive data from an upstream transformation in a mapping or mapplet.

Add input fields to define the data fields that you want to pass into the mapplet from the upstream transformation.

You can add multiple Input transformations to a mapplet. Each Input transformation in a mapplet becomes an input group in when you use the mapplet in a Mapplet transformation. You must connect at least one input group to an upstream transformation.

You can use an Input transformation in a mapplet, but not in a mapping.

Input fields

Add input fields to an Input transformation to define the fields that you want to pass into the mapplet. You must add at least one input field to each Input transformation.

Add input fields on the **Input Fields** tab. To add a field, click **Add Fields**, and then enter the field name, data type, precision, scale, and optional description. The description can contain up to 4000 characters.

When you use the mapplet in a Mapplet transformation, map at least one input field to the upstream transformation.

CHAPTER 8

Joiner transformation

The Joiner transformation can join data from two related heterogeneous sources. For example, you can use the Joiner transformation to join account information from flat files with data from the Salesforce Account object.

The Joiner transformation joins data based on the join conditions and the join type. A join condition matches fields between the two sources. You can create multiple join conditions. A join type defines the set of data that is included in the results.

When you link a transformation to the Joiner transformation, you connect it to the Master or Detail group. To improve job performance, connect the transformation that represents the smaller data set to the Master group.

To join more than two sources in a mapping, you can use multiple Joiner transformations. You can join the output from the Joiner transformation with another source pipeline. You can add Joiner transformations to the mapping until you join all source pipelines.

Field name conflicts can occur when you join sources with matching field names. You can resolve the conflict in one of the following ways:

- Create a field name conflict resolution.
- Rename matching fields in an upstream transformation.
- Pass data through an Expression transformation to rename fields.

Join condition

The join condition defines when incoming rows are joined. It includes fields from both sources that must match to join source rows.

You define one or more conditions based on equality between the master and detail data. For example, if two sets of employee data contain employee ID numbers, the following condition matches rows with the same employee IDs in both sets of data:

```
EMP_ID1 = EMP_ID2
```

Use one or more join conditions. Additional join conditions increase the time necessary to join the data. When you use multiple join conditions, the mapping task evaluates the conditions in the order that you specify.

Both fields in a condition must have the same data type. If you need to use two fields with non-matching data types, convert the data types so they match.

For example, when you try to join Char and Varchar data, any spaces that pad Char values are included as part of the string. Both fields might include the value "Shoes," but because the Char(40) field includes 35

trailing spaces, the values do not match. To ensure that the values match, change the data type of one field to match the other.

Note: The Joiner transformation does not match null values. To join rows with null values, you can replace null values with default values, and then join on the default values.

Join type

The join type determines the result set that passes to the rest of the mapping.

You can use the following join types:

Normal Join

Includes rows with matching join conditions. Discards rows that do not match the join conditions.

Master Outer

Includes all rows from the detail pipeline and the matching rows from the master pipeline. It discards the unmatched rows from the master pipeline.

Detail Outer

Includes all rows from the master pipeline and the matching rows from the detail pipeline. It discards the unmatched rows from the detail pipeline.

Full Outer

Includes rows with matching join conditions and all incoming data from the master pipeline and detail pipeline.

Advanced properties

You can configure advanced properties for a Joiner transformation. The advanced properties control settings such as the tracing level for session log messages, cache settings, null ordering, and whether the transformation is optional or required.

You can configure the following properties:

Property	Description
Tracing Level	Detail level of error and status messages that Data Integration writes in the session log. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Cache Directory	Specifies the directory used to cache master or detail rows and the index to these rows. By default, Data Integration uses the directory entered in the Secure Agent \$PMCacheDir property for the Data Integration Server. If you enter a new directory, make sure that the directory exists and contains enough disk space for the cache files. The directory can be on a mapped or mounted drive.
Null Ordering in Master	Null ordering in the master pipeline. Select Null is Highest Value or Null is Lowest Value.

Property	Description
Null Ordering in Detail	Null ordering in the detail pipeline. Select Null is Highest Value or Null is Lowest Value.
Data Cache Size	<p>Data cache size for the transformation. Select one of the following options:</p> <ul style="list-style-type: none"> - Auto. Data Integration sets the cache size automatically. If you select Auto, you can also configure a maximum amount of memory for Data Integration to allocate to the cache. - Value. Enter the cache size in bytes. <p>Default is Auto.</p>
Index Cache Size	<p>Index cache size for the transformation. Select one of the following options:</p> <ul style="list-style-type: none"> - Auto. Data Integration sets the cache size automatically. If you select Auto, you can also configure a maximum amount of memory for Data Integration to allocate to the cache. - Value. Enter the cache size in bytes. <p>Default is Auto.</p>
Sorted Input	Specifies that data is sorted. Select this option to join sorted data, which can improve performance.
Master Sort Order	Specifies the sort order of the master source data. Select Ascending if the master source data is in ascending order. If you select Ascending, enable sorted input. Default is Auto.
Transformation Scope	<p>Specifies how Data Integration applies the transformation logic to incoming data:</p> <ul style="list-style-type: none"> - Transaction. Applies the transformation logic to all rows in a transaction. Choose Transaction when a row of data depends on all rows in the same transaction, but does not depend on rows in other transactions. - All Input. Applies the transformation logic on all incoming data. When you choose All Input, Data Integration drops incoming transaction boundaries. Choose All Input when a row of data depends on all rows in the source. - Row. Applies the transformation logic to one row of data at-a-time. Choose Row when a row of data does not depend on any other row.
Optional	<p>Determines whether the transformation is optional. If a transformation is optional and there are no incoming fields, the mapping task can run and the data can go through another branch in the data flow. If a transformation is required and there are no incoming fields, the task fails.</p> <p>For example, you configure a parameter for the source connection. In one branch of the data flow, you add a transformation with a field rule so that only Date/Time data enters the transformation, and you specify that the transformation is optional. When you configure the mapping task, you select a source that does not have Date/Time data. The mapping task ignores the branch with the optional transformation, and the data flow continues through another branch of the mapping.</p>

Creating a Joiner transformation

Use a Joiner transformation to join data from two related heterogenous sources.

Before you create a Joiner transformation, add Source transformations to the mapping to represent source data. Include any other upstream transformations that you want to use.

If the data in the two pipelines include matching field names, rename one set of fields in a transformation upstream from the Joiner transformation.

1. In the **Transformation palette**, drag a Joiner transformation onto the mapping canvas.

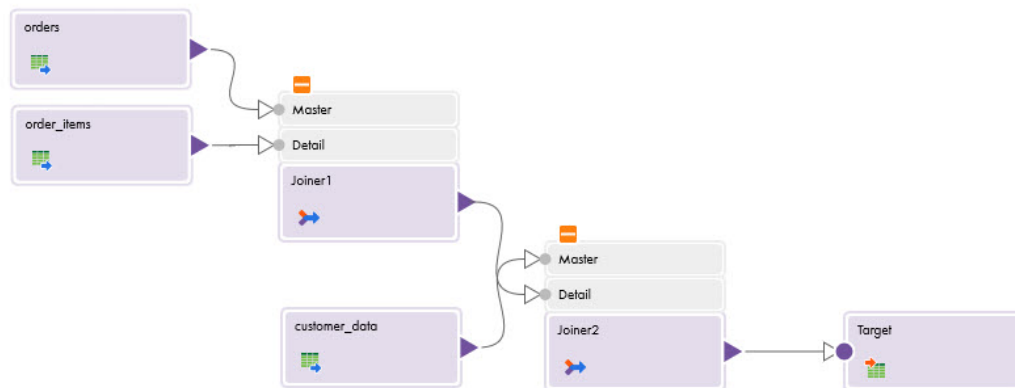
2. Connect an upstream transformation that represents one data set to the Master group of the Joiner transformation.
To improve job performance, use the transformation that represents the smaller data set.
3. Connect an upstream transformation that represents the other data set to the Detail group of the Joiner transformation.
4. On the **General** tab, enter a name and optional description for the transformation.
5. On the **Incoming Fields** tab, configure the field rules that define the data that enters the transformation.
6. On the **Join Condition** tab, select the join type.
7. To configure a join condition, select Simple for the join condition. Click **Add New Join Condition**, and then select the master and detail fields to use and the operator. You can create multiple join conditions.
Alternatively, to use a parameter for the join condition, select Completely Parameterized for the join condition.

You can add downstream transformations to the mapping and configure them. When the mapping is complete, you can validate and save the mapping.

Joiner transformation example

You're a marketing manager for an online retailer, and you want to merge order data with product and customer data from different Amazon S3 sources to understand what customers are purchasing. Use Joiner transformations to join the data from your sources.

You have three source data tables in an Amazon S3 bucket: *orders*, *order_items*, and *customer_data*. The following image shows a mapping that joins the data from these sources:



The mapping contains the following elements:

Source transformation for *orders*

The *orders* data table includes fields for the order number, date, price, and ID of the customer for each online order.

The following table shows a portion of *orders*:

order_id	order_date	customer_id	order_price
1005	2023-01-20	789	78.25
1006	2023-01-24	268	150.09
1007	2023-02-07	268	30.20

Source transformation for *order_items*

The *order_items* data table includes details about the items in each order, including the quantity and price.

The following table shows a portion of *order_items*:

order_id	item_id	qty	price
1005	5063	2	34.99
1006	2389	3	19.99
1006	5063	1	34.99
1007	9871	2	10.99

In the Source transformation, you rename the field *order_id* to *items_order_id* to avoid a field name conflict when you join *order_items* with *orders*.

Source transformation for *customer_data*

The *customer_data* table includes fields for information that the customers provide, including their name, date of birth, and phone number.

The following table shows a portion of *customer_data*:

c_id	c_name	c_dob
789	Kelcy Almeida	1969-07-20
268	Chidi Donalds	1972-12-07

Joiner transformation for *orders* and *order_items*

The first Joiner transformation performs a normal join between *orders* and *order_items*. The *orders* Source transformation is the master group and the *order_items* Source transformation is the detail group so that order information is added to each item ordered.

The Joiner transformation uses the following join condition to match the data by the order ID: *order_id* = *items_order_id*.

Joiner transformation for *customer_data*

The second Joiner transformation performs a detail outer join between *customer_data* and the output from the first Joiner transformation. The transformation uses the *customer_data* Source transformation as the master group since it is the smaller data set.

The second Joiner transformation uses the following join condition to match the data by the customer ID: `customer_id = c_id`.

Target transformation

The Target transformation writes the data to a new file in Amazon S3. You can configure the incoming fields to exclude the duplicate fields that result from the joins.

The following table shows a portion of the output data:

order_id	order_date	order_price	item_id	qty	price	c_id	c_name	c_dob
1005	2023-01-20	78.25	5063	2	34.99	789	Kelcy Almeida	1969-07-20
1006	2023-01-24	150.09	2389	3	19.99	268	Chidi Donalds	1972-12-07
1006	2023-01-24	150.09	5063	1	34.99	268	Chidi Donalds	1972-12-07
1007	2023-02-07	30.20	9871	2	10.99	268	Chidi Donalds	1972-12-07

CHAPTER 9

Lookup transformation

Use a Lookup transformation to retrieve data based on a specified lookup condition. For example, you can use a Lookup transformation to retrieve values from a database table for codes used in source data.

When a mapping task includes a Lookup transformation, the task queries the lookup source based on the lookup fields and a lookup condition. The Lookup transformation returns the result of the lookup to the target or another transformation. You can configure the Lookup transformation to return a single row or multiple rows. When you configure the Lookup transformation to return a single row, the Lookup transformation is a passive transformation. When you configure the Lookup transformation to return multiple rows, the Lookup transformation is an active transformation.

You can use multiple Lookup transformations in a mapping.

Perform the following tasks with a Lookup transformation:

- Get a related value. Retrieve a value from the lookup table based on a value in the source. For example, the source has an employee ID. Retrieve the employee name from the lookup table.
- Get multiple values. Retrieve multiple rows from a lookup table. For example, return all employees in a department.
- Update slowly changing dimension tables. Determine whether rows exist in a target.

You can perform the following types of lookups:

Connected or unconnected lookup

A connected Lookup transformation receives source data, performs a lookup, and returns data.

An unconnected Lookup transformation is not connected to a source or target. A transformation calls the Lookup transformation with a lookup expression. The unconnected Lookup transformation returns one column to the calling transformation.

Cached or uncached lookup

Cache the lookup source to optimize performance. If you cache the lookup source, you can use a static or dynamic cache. You can also use a persistent or non-persistent cache.

By default, the lookup cache remains static and does not change as the mapping task runs. With a dynamic cache, the task inserts or updates rows in the cache as the target table changes. When you cache the target table as the lookup source, you can look up values in the cache to determine if the values exist in the target.

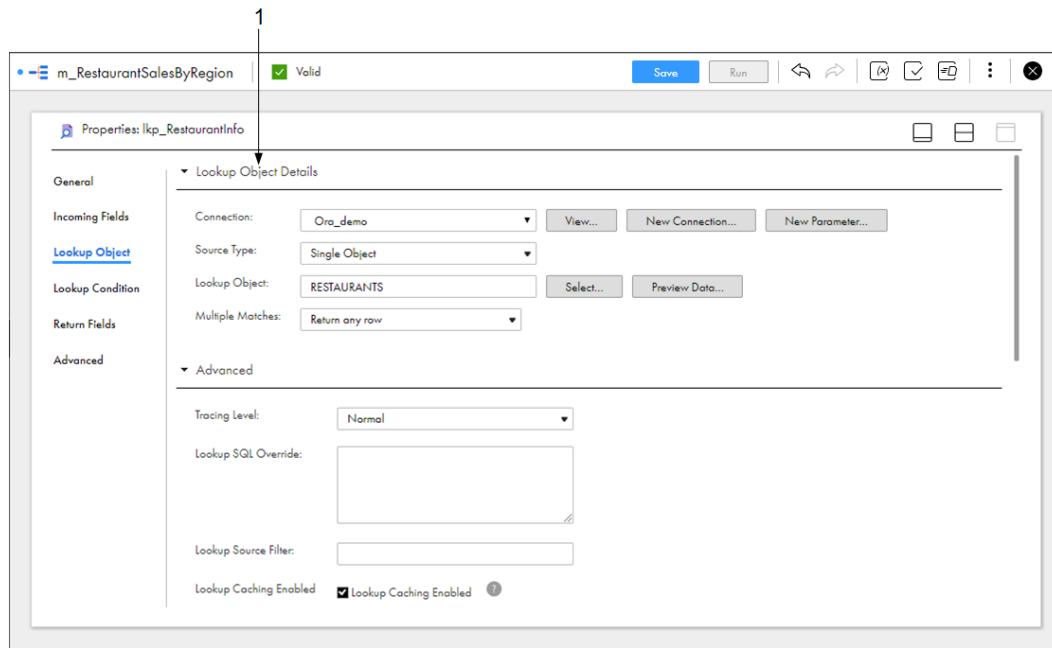
By default, the lookup cache is also non-persistent. Therefore, Data Integration deletes the cache files after the mapping task completes. If the lookup table does not change between mapping runs, you can use a persistent cache to increase performance.

Lookup object

The Lookup object is the source object that Data Integration queries when it performs the lookup. The lookup object is also called the lookup source.

Select the lookup source on the **Lookup Object** tab of the Properties panel. The properties that you configure for the lookup source vary based on the connection type

The following image shows the **Lookup Object** tab for a relational lookup:



1. Lookup object details where you configure the connection, source type, lookup object, and multiple match behavior.

You can select the lookup source in the following ways:

Select the connection and lookup object.

In the **Lookup Object Details** area, select the connection, source type, and lookup object. You can also create a new connection.

Use a parameter.

You can use an input parameter to define the connection or lookup object when you run the mapping task. For more information about parameters, see *Mappings*.

Use a custom query.

You can use a custom query to reduce the number of columns to query. You might want to use a custom query when the source object is large.

You must also specify the transformation behavior when the lookup returns multiple matches.

Lookup object properties

When you configure a lookup, you select the lookup connection and lookup object. You also define the behavior when a lookup condition returns more than one match.

The following table describes the lookup object properties:

Property	Description
Connection	Name of the lookup connection.
Source Type	Source type. For database lookups, the source type can be single object, parameter, or query. For flat file lookups, the source type can be single object, file list, command, or parameter.
Lookup Object	If the source type is a single object, this property specifies the lookup file, table, or object. If the source property is a file list, this property specifies the text file that contains the file list. If the source type is a command, this property specifies the sample file from which Data Integration imports the return fields.
Parameter	If the source type is a parameter, this property specifies the parameter.
Define Query	If the source type is a query, displays the Edit Custom Query dialog box. Enter a valid custom query and click OK .
Multiple Matches	Behavior when the lookup condition returns multiple matches. You can return all rows, any row, the first row, the last row, or an error. If you choose all rows and there are multiple matches, the Lookup transformation is an active transformation. If you choose any row, the first row, or the last row and there are multiple matches, the Lookup transformation is a passive transformation.
Formatting Options	File formatting options which are applicable if the lookup object is a flat file. Opens the Formatting Options dialog box to define the format of the file. Configure the following file format options: <ul style="list-style-type: none">- Delimiter. Delimiter character.- Text Qualifier. Character to qualify text.- Escape character. Escape character.- Field labels. Determines if the task generates field labels or imports labels from the source file.- First data row. The first row of data. The task starts the read at the row number that you enter.
Command	If the source type is a command, this property specifies the command that generates the file list.

For more information about file lists and commands, see [“File lists” on page 31](#). For more information about parameters and file formats, see *Mappings*.

Multiple match policy restrictions

When you configure a lookup, you define the behavior when a lookup condition returns more than one match. Some types of lookups have restrictions on the multiple match policy.

The following types of lookups have multiple match policy restrictions:

Uncached lookups

Some connector types do not support the multiple match policies **Return first row** and **Return last row** in uncached lookups. If you select either of these policies, and the connector does not support the policy in uncached lookups, Data Integration enables the **Lookup Caching Enabled** advanced property, and you cannot edit it.

Lookups that use a dynamic cache

If the Lookup transformation uses a dynamic cache, you must configure the multiple match policy to return an error. Other multiple match policies are not supported.

Salesforce lookups

When you perform a lookup against a Salesforce object, you can return any row or return an error.

For more information about the multiple match policies supported by different connectors, see the help for the appropriate connector.

Custom queries

You can create a custom query for database lookups. You might create a custom query to reduce the number of columns to query.

To use a custom query as a lookup source, select **Query** as the source type, and then define the query. When you define the query, enter an SQL SELECT statement to select the source columns that you want to use. Data Integration uses the SQL statement to retrieve source column information.

When you use a custom query in a lookup transformation, use the following format for the SQL statement:

- For a relational database connection, use an alias for each column in the SQL statement, for example:

```
SELECT COL1 AS COL1, COL2 AS COL2, COL3 AS COL3 from TABLE_NAME
```

- For other types of database connections, use SQL that is valid for the source database. You can use database-specific functions in the query.

To use a custom query as a lookup source, you must enable lookup caching.

Tip: Test the SQL statement you want to use on the source database before you create a custom query. Data Integration does not display specific error messages for invalid SQL statements.

Lookup condition

The lookup condition defines when the lookup returns values from the lookup object. When you configure the lookup condition, you compare the value of one or more fields from the data flow with values in the lookup object.

A lookup condition includes an incoming field from the data flow, a field from the lookup object, and an operator. For flat file and database connections, you can use the following operators in a lookup condition:

- = (Equal to)
- < (Less than)
- > (Greater than)
- <= (Less than or equal to)
- >= (Greater than or equal to)
- != (Not equal to)

For other connections and for Lookup transformations that use a dynamic cache, you can use the = (Equal to) operator in a lookup condition.

Note the following information about lookup conditions:

- When you enter multiple conditions, the mapping task evaluates the lookup conditions using the AND logical operator to join the conditions. It returns rows that match all of the lookup conditions.
- When you include multiple conditions, to optimize performance enter the conditions in the following order:
 1. = (Equal to)
 2. < (Less than), <= (Less than or equal to), > (Greater than), >= (Greater than or equal to)
 3. != (Not equal to)
- The lookup condition matches null values. When an input field is NULL, the mapping task evaluates the NULL equal to null values in the lookup.

Lookup return fields

Select the fields to return from the lookup object on the **Return Fields** tab of the Properties panel.

The **Return Fields** tab displays all fields from the selected lookup object. By default, the mapping includes all fields in the list in the data flow. Remove fields that you do not want to use.

For Lookup transformations that use a dynamic cache, the task returns the NewLookupRow return field. You cannot remove this field. For more information about the NewLookupRow return field, see [“Dynamic cache updates” on page 87](#).

You can edit the name of a field and edit the metadata for a field. When you edit field metadata, you can change the name, native data type, native precision, and native scale.

For some relational connection types, you can specify the default column value for lookup return fields. You can use a string or an expression for the default value. If you use a string, enclose the string in single quotes, for example, 'ABC'. To see if you can set default values for lookup return fields, see the help for the appropriate connector.

Note: When you change field metadata, you cannot automatically revert your changes. Avoid making changes that can cause errors when you run the task.

You can add a field to the field list if the field exists in the lookup object. To add a field, you need exact field details, including the field name, data type, precision, and scale.

To restore the original fields from the lookup object, use the Synchronize icon. Synchronization restores deleted fields, adds new fields, and retains added fields with corresponding fields in the lookup object. Synchronization removes any added fields that do not have corresponding fields in the lookup object. Synchronization does not revert any local changes to the field metadata.

The following table describes the options that you can use on the **Return Fields** tab:

Field option	Description
Add Field icon	Adds a field from the selected lookup object. Use to retrieve a field from the object that does not display in the list. Opens the New Field dialog box. Enter the exact field name, data type, precision, and scale, and click OK .
Delete icon	Deletes the field from the list, removing the field from the data flow.
Sort icon	Sorts fields in native order, ascending order, or descending order.

Field option	Description
Find field	Enter a search string to find the fields with names that contain the string.
Options menu	<p>Contains the following options:</p> <ul style="list-style-type: none"> - Use Technical Field Names. Displays field names by technical field name. - Use Labels. Displays field names by label. - Edit Metadata. Change the name, native type, native precision, or native scale, if applicable for the data type. For some relational connection types, you can set the default value for lookup return fields. <p>When you edit metadata, you can display native names by technical field name or label.</p>
Synchronize icon	<p>Synchronizes the list of fields with the lookup object.</p> <p>Note: If you select this option, you lose any changes you make to the metadata for return fields.</p>
Ignore in Comparison	<p>When the transformation uses a dynamic cache, by default, Data Integration compares the values in all lookup fields with the values in the associated incoming fields to determine whether to update the row in the lookup cache.</p> <p>Enable this property if you want Data Integration to ignore the field when it compares the values before updating a row. You must configure the transformation to compare at least one field.</p> <p>This property is displayed for each field when the Lookup transformation uses a dynamic cache.</p>
Retain existing fields at runtime	<p>If field metadata changes after a mapping is saved, Data Integration uses the updated field metadata when you run the mapping. Typically, this is the desired behavior. However, if the mapping uses a native flat file connection and you want to retain the metadata used at design time, enable the Retain existing fields at runtime option. When you enable this option, Data Integration mapping tasks will use the field metadata that was used when you created the mapping.</p>

Advanced properties

You can configure advanced properties for a Lookup transformation. The connection type and the mapping type determine which advanced properties are available for the Lookup transformation.

You can set the following properties:

Property	Description
Tracing Level	Detail level of error and status messages that Data Integration writes in the session log. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Lookup Source File Directory	<p>Name of the directory for a flat file lookup source. By default, Data Integration reads files from the lookup source connection directory.</p> <p>You can also use an input parameter to specify the source file directory.</p> <p>If you use the service process variable directory \$PMLookupFileDir, the task writes target files to the configured path for the system variable. To find the configured path of a system variable, see the pmdtm.cfg file located at the following directory:</p> <pre><Secure Agent installation directory>\apps\Data_Integration_Server\<Data Integration Server version>\ICS\main\bin\rdtm</pre> <p>You can also find the configured path for the \$PMLookupFileDir variable in the Data Integration Server system configuration details in Administrator.</p>

Property	Description
Lookup Source File Name	File name, or file name and path of the lookup source file.
Lookup SQL Override	Overrides the default SQL statement to query the lookup table. Specifies the SQL statement you want to use for querying lookup values. Use with lookup cache enabled.
Lookup Source Filter	Restricts the lookups based on the value of data in any field in the Lookup transformation. Use with lookup cache enabled.
Lookup Caching Enabled	Determines whether to cache lookup data during the runtime session. When you enable caching, Data Integration queries the lookup source once and caches the values for use during the session, which can improve performance. When you disable caching, a SELECT statement gets the lookup values each time a row passes into the transformation. Caching is enabled and is not editable in the following circumstances: <ul style="list-style-type: none"> - When the lookup source type does not support uncached lookups. - When you select a multiple match policy, but the lookup source type does not support the policy in uncached lookups. For example, you cannot disable caching when you select Return first row or Return last row as the multiple match policy for a lookup against an Amazon Redshift V2 source. Default is enabled. This property is not displayed for flat file lookups because flat file lookups are always cached.
Lookup Cache Directory Name	Specifies the directory to store cached lookup data when you select Lookup Caching Enabled. The directory name can be an environment variable.
Lookup Cache Persistent	Specifies whether to save the lookup cache file to reuse it the next time Data Integration processes a Lookup transformation configured to use the cache.
Cache File Name Prefix	Use with persistent lookup cache. Specifies the file name prefix to use with persistent lookup cache files. Data Integration uses the file name prefix as the file name for the persistent cache files that it saves to disk. If the named persistent cache files exist, Data Integration builds the memory cache from the files. If the named persistent cache files do not exist, Data Integration rebuilds the persistent cache files. Enter the prefix. Do not include a file extension such as .idx or .dat.
Re-cache from Lookup Source	Use with persistent lookup cache. When selected, Data Integration rebuilds the persistent lookup cache from the lookup source when it first calls the Lookup transformation instance.
Data Cache Size	Data cache size for the transformation. Select one of the following options: <ul style="list-style-type: none"> - Auto. Data Integration sets the cache size automatically. If you select Auto, you can also configure a maximum amount of memory for Data Integration to allocate to the cache. - Value. Enter the cache size in bytes. Default is Auto.
Index Cache Size	Index cache size for the transformation. Select one of the following options: <ul style="list-style-type: none"> - Auto. Data Integration sets the cache size automatically. If you select Auto, you can also configure a maximum amount of memory for Data Integration to allocate to the cache. - Value. Enter the cache size in bytes. Default is Auto.

Property	Description
Dynamic Lookup Cache	Determines whether to use a dynamic cache instead of a static cache. When you enable dynamic caching, the task updates the cache as it inserts or updates rows in the target so that the cache and target remain in sync. Use when lookup cache is enabled.
Output Old Value On Update	When you enable this property, when the task updates a row in the cache, it outputs the value that existed in the lookup cache before it updated the row. When it inserts a row, it returns a null value. Use when dynamic lookup cache is enabled.
Synchronize dynamic cache	When you enable this property, the task retrieves the latest values from the lookup source and updates the dynamic cache. This is helpful when multiple tasks that use the same lookup source are running simultaneously. Use when dynamic lookup cache is enabled. Cache synchronization is not available for some connection types. For more information, see the help for the appropriate connector.
Insert Else Update	Applies to rows entering the Lookup transformation with the row type of insert. When enabled, the mapping task inserts rows in the cache and updates existing rows. When disabled, the mapping task does not update existing rows. Use when dynamic lookup cache is enabled.
Lookup Source is Static	When you enable this property, the lookup source does not change when the task runs.
Datetime Format	Sets the datetime format and field width. Milliseconds, microseconds, or nanoseconds formats have a field width of 29. If you do not specify a datetime format here, you can enter any datetime format for fields. Default is YYYY-MM-DD HH24:MI:SS. The format does not change the size of the field. Default is YYYY-MM-DD HH24:MI:SS. The Datetime format does not change the size of the field.
Thousand Separator	Specifies the thousand separator. Enter a comma (,) a period (.) or None. Default is None.
Decimal Separator	Specifies the decimal separator. Enter a comma (,) or a period (.). Default is period.
Case Sensitive String Comparison	Determines whether to enable case-sensitive string comparisons when you perform lookups on string columns in flat files. For relational uncached lookups, the column types that support case-sensitive comparison depend on the database.
Null Ordering	Determines how the null values are ordered. You can choose to sort null values high or low. By default, null values are sorted high. This overrides configuration to treat nulls in comparison operators as high, low, or null. For relational lookups, null ordering depends on the database default value.
Sorted Input	Indicates whether or not the lookup file data is in sorted order. This increases lookup performance for file lookups. If you enable sorted input and the condition columns are not grouped, the session fails. If the condition columns are grouped but not sorted, the lookup is processed as if you did not configure sorted input.
Pre-build Lookup Cache	Specifies to build the lookup cache before the Lookup transformation receives data. Multiple lookup cache files can be built at the same time to improve performance.

Property	Description
Subsecond Precision	<p>Sets the subsecond precision for datetime fields. For relational lookups, you can change the precision for databases that have an editable scale for datetime data. You can change the subsecond precision for Oracle Timestamp, Informix Datetime, and Teradata Timestamp data types.</p> <p>Enter a positive integer value from 0 to 9. Default is 6 microseconds.</p> <p>If you enable pushdown optimization in a task, the database returns the complete datetime value, regardless of the subsecond precision setting.</p>
Optional	<p>Determines whether the transformation is optional. If a transformation is optional and there are no incoming fields, the mapping task can run and the data can go through another branch in the data flow. If a transformation is required and there are no incoming fields, the task fails.</p> <p>For example, you configure a parameter for the source connection. In one branch of the data flow, you add a transformation with a field rule so that only Date/Time data enters the transformation, and you specify that the transformation is optional. When you configure the mapping task, you select a source that does not have Date/Time data. The mapping task ignores the branch with the optional transformation, and the data flow continues through another branch of the mapping.</p>

Lookup SQL overrides

When a mapping includes a Lookup transformation, the mapping task queries the lookup object based on the fields and properties that you configure in the Lookup transformation. The mapping task runs a default lookup query when the first row of data enters the Lookup transformation. If the Lookup transformation performs a relational lookup, you can override the default query.

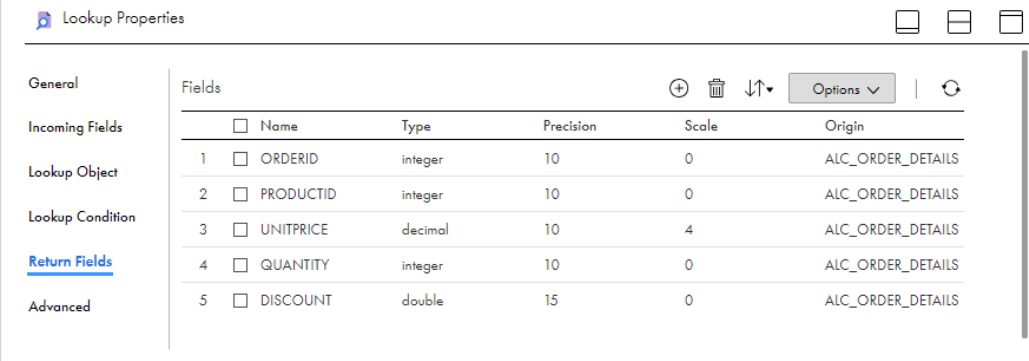
The default query contains a SELECT statement that includes all lookup fields in the mapping. The SELECT statement also contains an ORDER BY clause that orders all columns in the same order in which they appear in the Lookup transformation. To view the default query, run the mapping task. The default query appears in the log file.

If you want to change the ORDER BY clause, add a WHERE clause, or transform the lookup data before it is cached, you can override the default query. For example, you might use database functions to adjust the data types or formats in the lookup table to match the data types and formats of fields that are used in the mapping. Or, you might override the default query to query multiple tables.

Override the default query on the **Advanced** tab of the Lookup transformation. Enter the entire SELECT statement in the **Lookup SQL Override** field. Use an alias for each column in the query. If you want to change the ORDER BY clause, you must also append "--" to the end of the query to suppress the ORDER BY clause that the mapping task generates.

Example

A Lookup transformation returns the following fields from Microsoft SQL Server table ALC_ORDER_DETAILS:



	Name	Type	Precision	Scale	Origin
1	<input type="checkbox"/> ORDERID	integer	10	0	ALC_ORDER_DETAILS
2	<input type="checkbox"/> PRODUCTID	integer	10	0	ALC_ORDER_DETAILS
3	<input type="checkbox"/> UNITPRICE	decimal	10	4	ALC_ORDER_DETAILS
4	<input type="checkbox"/> QUANTITY	integer	10	0	ALC_ORDER_DETAILS
5	<input type="checkbox"/> DISCOUNT	double	15	0	ALC_ORDER_DETAILS

The transformation uses the following lookup condition:

```
ORDERID=in_ORDERID
```

When you run the mapping task, the following query appears in the log file:

```
LKPDP_1> DBG_21097 [2018-11-07 14:11:33.509] Lookup Transformation [lkp_ALC_ORDER_DETAILS]:  
Default sql to create lookup cache: SELECT PRODUCTID,UNITPRICE,QUANTITY,DISCOUNT,ORDERID FROM  
"icsauto"."ALC_ORDER_DETAILS" ORDER BY ORDERID,PRODUCTID,UNITPRICE,QUANTITY,DISCOUNT
```

To override the ORDER BY clause and sort by PRODUCTID, enter the following query in the **Lookup SQL Override** field on the **Advanced** tab:

```
SELECT PRODUCTID AS PRODUCTID, UNITPRICE AS UNITPRICE, QUANTITY AS QUANTITY, DISCOUNT AS  
DISCOUNT, ORDERID AS ORDERID FROM "icsauto"."ALC_ORDER_DETAILS" ORDER BY PRODUCTID --
```

When you run the mapping task again, the following query appears in the log file:

```
LKPDP_1> DBG_21312 [2018-11-07 14:14:36.734] Lookup Transformation [lkp_ALC_ORDER_DETAILS]:  
Lookup override sql to create cache: SELECT PRODUCTID AS PRODUCTID, UNITPRICE AS UNITPRICE,  
QUANTITY AS QUANTITY, DISCOUNT AS DISCOUNT, ORDERID AS ORDERID FROM  
"icsauto"."ALC_ORDER_DETAILS" ORDER BY PRODUCTID -- ORDER BY  
ORDERID,PRODUCTID,UNITPRICE,QUANTITY,DISCOUNT
```

Guidelines for overriding the lookup query

Certain rules and guidelines apply when you override a lookup query.

Use the following guidelines:

- You can override the lookup SQL query for relational lookups.
- If you override the lookup query, you must also enable lookup caching for the transformation.
- Enter the entire SELECT statement using the syntax that is required by the database.
- Enclose all database reserved words in quotes.
- Include all lookup and return fields in the SELECT statement.
If you add or subtract fields in the SELECT statement, the mapping task fails.
- Use an alias for each column in the query.
If you do not use column aliases, the mapping task fails with the following error:

```
Failed to initialize transformation [<Lookup Transformation Name>]
```

- To override the ORDER BY clause, append "--" at the end of the query.
The mapping task generates an ORDER BY clause, even when you enter one in the override. Therefore, you must enter two dashes (--) at the end of the query to suppress the generated ORDER BY clause.
- If the ORDER BY clause contains multiple columns, enter the columns in the same order as the fields in the lookup condition.
- If the mapping task uses pushdown optimization, you cannot override the ORDER BY clause or suppress the generated ORDER BY clause with comment notation.
- If multiple Lookup transformations share a lookup cache, use the same lookup SQL override for each Lookup transformation.
- When you configure a Lookup transformation that returns all rows, the mapping task builds the lookup cache with sorted keys. When the transformation retrieves all rows in a lookup, the mapping task builds the data cache with the keys in sorted order. The mapping task cannot retrieve all the rows from the cache if the rows are not sorted. If the data is not sorted on the keys, you might get unexpected results.
- You cannot include parameters in the lookup SQL override.
- If you configure a lookup SQL override and a lookup source filter in the same transformation, the mapping task ignores the filter.

Lookup source filter

You can configure a lookup source filter for a relational Lookup transformation that has caching enabled. Add the lookup source filter to limit the number of lookups that the mapping task performs on a lookup source table. When you configure a lookup source filter, the mapping task performs lookups based on the results of the filter statement.

To configure a lookup source filter, open the **Advanced** tab of the Lookup transformation, and enter the filter in the **Lookup Source Filter** field. Do not include the WHERE keyword in the filter condition.

For example, you might need to retrieve the last name of every employee whose ID is greater than 510.

You configure the following lookup source filter on the EmployeeID field in the Lookup transformation:

```
EmployeeID >= 510
```

When the mapping task reads the source row, it performs a lookup on the cache when the value of EmployeeID is greater than 510. When EmployeeID is less than or equal to 510, the Lookup transformation does not retrieve the last name.

When you add a lookup source filter to the Lookup query for a mapping task that uses pushdown optimization, the mapping task creates a view to represent the SQL override. The mapping task runs an SQL query against this view to push the transformation logic to the database.

Note: If you configure a lookup source filter and a lookup SQL override in the same transformation, the mapping task ignores the filter.

Dynamic lookup cache

Use a dynamic lookup cache to keep the lookup cache synchronized with the target.

When you enable lookup caching, a mapping task builds the lookup cache when it processes the first lookup request. The cache can be static or dynamic. If the cache is static, the data in the lookup cache doesn't change as the mapping task runs. If the task uses the cache multiple times, the task uses the same data. If the cache is dynamic, the task updates the cache based on the actions in the task, so if the task uses the lookup multiple times, downstream transformations can use updated data.

You can use a dynamic cache with most types of lookup sources. You cannot use a dynamic cache with flat file or Salesforce lookups. For more information about using a dynamic cache with a specific type of lookup source, see the help for the appropriate connector.

Based on the results of the lookup query, the row type, and the Lookup transformation properties, the mapping task performs one of the following actions on the dynamic lookup cache when it reads a row from the source:

Inserts the row into the cache

The mapping task inserts the row when the row is not in the cache. The mapping task flags the row as insert.

Updates the row in the cache

The mapping task updates the row when the row exists in the cache. The mapping task updates the row in the cache based on the input fields. The mapping task flags the row as an update row.

Makes no change to the cache

The mapping task makes no change when the row is in the cache and nothing changes. The mapping task flags the row as unchanged.

The dynamic Lookup transformation includes the return field, `NewLookupRow`, which describes the changes the task makes to each row in the cache. Based on the value of the `NewLookupRow`, you can also configure a Router or Filter transformation with the dynamic Lookup transformation to route insert or update rows to the target table. You can route unchanged rows to another target table or flat file, or you can drop them.

You cannot use a parameterized source, target, or lookup with a Lookup transformation that uses a dynamic cache.

Static and dynamic lookup comparison

You might want to use dynamic cache instead of a static cache if the source might contain duplicate private keys. Or, you might want to use a dynamic cache when the source contains a large table of data to optimize performance.

Data Integration processes lookup conditions differently based on whether you configure the Lookup transformation to use a static or dynamic cache.

The following table compares a Lookup transformation that uses a static cache to a Lookup transformation that uses a dynamic cache:

Static Lookup Cache	Dynamic Lookup Cache
The cache does not change during the task run.	The task inserts or updates rows in the cache as it passes rows to the target.
You can use a flat file, relational database, and other connection types such as Salesforce for lookup.	You cannot use a flat file or Salesforce connection type.
When the lookup condition is true, the task returns a value from the lookup table or cache. When the condition is not true, the task returns the default value.	When the lookup condition is true, the task either updates the row in the cache and target or leaves the cache unchanged. This indicates that the row is in the cache and target table. When the lookup condition is not true, the task either inserts the row in the cache and target or leaves the cache unchanged based on the row type. This indicates that the row is not in the cache or target table.

Dynamic cache updates

When the mapping task reads a row, it changes the lookup cache depending on the results of the lookup query and the Lookup transformation properties that you define. The mapping task assigns a value to the NewLookupRow return field that indicates the action it takes.

The following table lists the possible NewLookupRow values:

NewLookupRow Value	Description
0	Mapping task does not update or insert the row in the cache.
1	Mapping task inserts the row into the cache.
2	Mapping task updates the row in the cache.

You can use the NewLookupRow values in downstream transformations.

Inserts and updates for insert rows

You can configure how the mapping task handles inserts and updates to the cache for insert rows. To update existing rows in the dynamic lookup cache when the row type is insert, enable the **Insert Else Update** advanced property for the transformation.

Note: This property only applies to rows entering the Lookup transformation where the row type is insert. When a row of any other row type, such as update, enters the Lookup transformation, the **Insert Else Update** property has no effect on how the mapping task handles the row.

When you enable **Insert Else Update** and the row type entering the Lookup transformation is insert, the mapping task inserts the row into the cache if it is new. If the row exists in the index cache but the data cache is different than the current row, the mapping task updates the row in the data cache.

If you do not enable **Insert Else Update** and the row type entering the Lookup transformation is insert, the mapping task inserts the row into the cache if it is new, and makes no change to the cache if the row exists.

The following table describes how the mapping task changes the lookup cache when the row type of the rows entering the Lookup transformation is insert:

Insert Else Update Option	Row found in cache?	Data cache is different?	Lookup Cache Result	NewLookupRow Value
Disabled - insert only	Yes	-	No change	0
Disabled - insert only	No	-	Insert	1
Enabled	Yes	Yes	Update	2
Enabled	Yes	No	No change	0
Enabled	No	-	Insert	1

Dynamic cache and lookup source synchronization

The Lookup transformation maintains a dynamic lookup cache to track the rows that it passes to the target. When multiple tasks update the same target, you can configure the Lookup transformation in each task to synchronize the dynamic lookup cache to the same lookup source instead of a target.

To synchronize the cache with the lookup source, enable the **Synchronize Dynamic Cache** property for the Lookup transformation.

When you configure a Lookup transformation to synchronize the cache with the Lookup source, the Lookup transformation performs a lookup on the lookup source. If the data does not exist in the Lookup source, the Lookup transformation inserts the row into the lookup source before it updates the dynamic lookup cache.

The data might exist in the lookup source if another task inserted the row. To synchronize the lookup cache to the lookup source, the task retrieves the latest values from the lookup source. The Lookup transformation inserts the values from the Lookup source in the dynamic lookup cache.

For example, you have multiple tasks running simultaneously. Each task generates product numbers for new product names. When a task generates a product number, the other tasks must use the same product number to identify the product. The product number is generated once and inserted in the lookup source. If another task processes a row containing the product, it must use the product number that is in the lookup source. Each task performs a lookup on the lookup source to determine which product numbers have already been generated.

When you configure the Lookup transformation to synchronize the cache with the lookup source, the task performs a lookup on the dynamic lookup cache for insert rows. If data does not exist in the dynamic lookup cache, the task performs a lookup on the lookup source. It then completes one of the following tasks:

- If data exists in the lookup source, the task inserts a row in the dynamic lookup cache with the columns from the lookup source. It does not update the cache with the source row.
- If data does not exist in the lookup source, the task inserts the data into the lookup source and inserts the row into the cache.

The lookup source contains the same fields as the lookup cache. The task does not insert a row in the lookup cache unless the column is projected from the Lookup transformation or the field is part of a lookup condition.

Dynamic cache and target synchronization

Configure downstream transformations to ensure that the dynamic lookup cache and target are synchronized.

When you use a dynamic lookup cache, the mapping task writes to the lookup cache before it writes to the target table. The lookup cache and target table can become unsynchronized if the task does not write the data to the target. For example, the target database might reject the data.

Consider the following guidelines to keep the lookup cache synchronized with the lookup table:

- Use the Router transformation to pass rows to the cached target when the NewLookupRow value equals one or two.
- Use the Router transformation to drop rows when the NewLookupRow value equals zero. Or, output the rows to a different target.

Field mapping

When you use a dynamic lookup cache, map incoming fields with lookup cache fields on the **Field Mapping** tab. The **Field Mapping** tab is only available when you configure the Lookup transformation to use a dynamic cache.

You must map all of the incoming fields when you use a dynamic cache so that the cache can update as the task runs. Optionally, you can map the Sequence-ID field instead of an incoming field if you want to create a generated key for a field in the target object.

Generated key fields

When you configure a dynamic lookup cache, you can create a generated key for a field in the target object.

To create a generated key for a field in the target object, map the Sequence-ID field to a lookup cache field on the **Field Mapping** tab. You can map the Sequence-ID field instead of an incoming field to lookup cache fields with the integer or Bigint data type. For integer lookup fields, the generated key maximum value is 2,147,483,647. For Bigint lookup fields, the generated key maximum value is 9,223,372,036,854,775,807.

When you map the Sequence-ID field, Data Integration generates a key when it inserts a row into the lookup cache.

Data Integration uses the following process to generate sequence IDs:

1. When Data Integration creates the dynamic lookup cache, it tracks the range of values for each field that has a sequence ID in the dynamic lookup cache.
2. When Data Integration inserts a row of data into the cache, it generates a key for a field by incrementing the greatest sequence ID value by one.
3. When Data Integration reaches the maximum number for a generated sequence ID, it starts over at one. Data Integration increments each sequence ID by one until it reaches the smallest existing value minus one. If Data Integration runs out of unique sequence ID numbers, the mapping task fails.

Data Integration generates a sequence ID for each row it inserts into the cache.

Ignore fields in comparison

When you use a dynamic lookup cache, you can configure fields to be ignored when Data Integration compares the values in the lookup fields with the values in the associated incoming fields. Ignoring some fields in the comparison can improve mapping performance.

When you run a mapping that uses a dynamic lookup cache, by default, Data Integration compares the values in all lookup fields with the values in the associated incoming fields. Data Integration compares the values to determine whether to update the row in the lookup cache. When a value in an incoming field differs from the value in the lookup field, Data Integration updates the row in the cache.

If you do not want to compare all fields, you can choose the fields that you want Data Integration to ignore when it compares fields. For example, the source data includes a column that indicates whether the row contains data that you need to update. Enable the **Ignore in Comparison** property for all lookup fields except the field that indicates whether to update the row in the cache and target table.

Configure the fields to be ignored on the **Return Fields** tab of the Lookup transformation. To ignore a field, enable the **Ignore in Comparison** property for the field.

You must configure the transformation to compare at least one field.

Dynamic lookup query overrides

When you add a WHERE clause in a Lookup transformation that uses a dynamic cache, connect a Filter transformation before the Lookup transformation to filter rows that you do not want to insert into the cache or target table. If you do not include the Filter transformation, you might get inconsistent results between the cache and the target table.

For example, you configure a Lookup transformation to perform a dynamic lookup on the employee table, EMP, matching rows by EMP_ID. You define the following lookup SQL override:

```
SELECT EMP_ID, EMP_STATUS FROM EMP ORDER BY EMP_ID, EMP_STATUS WHERE EMP_STATUS = 4
```

When you first run the mapping, the mapping task builds the lookup cache from the target table based on the lookup SQL override. All rows in the cache match the condition in the WHERE clause, EMP_STATUS = 4.

The mapping task reads a source row that meets the lookup condition you specify, but the value of EMP_STATUS is 2. Although the target might have the row where EMP_STATUS is 2, the mapping task does not find the row in the cache because of the SQL override. The mapping task inserts the row into the cache and passes the row to the target table. When the mapping task inserts this row in the target table, you might get inconsistent results when the row already exists. In addition, not all rows in the cache match the condition in the WHERE clause in the SQL override.

To verify that you only insert rows into the cache that match the WHERE clause, you add a Filter transformation before the Lookup transformation and define the filter condition as the condition in the WHERE clause in the lookup SQL override.

You enter the following filter condition in the Filter transformation and the WHERE clause in the SQL override:

```
EMP_STATUS = 4
```

Persistent lookup cache

You can configure a Lookup transformation to use a persistent cache. When you use a persistent cache, Data Integration saves and reuses the cache files from mapping run to mapping run.

By default, Data Integration uses a non-persistent cache when you enable caching in a Lookup transformation. When you use a non-persistent cache, Data Integration deletes the cache files at the end of the mapping run. The next time you run the mapping, Data Integration builds the memory cache from the database.

If the lookup table does not change between mapping runs, you can use a persistent cache. A persistent cache can improve mapping performance because it eliminates the time required to read the lookup table. The first time that Data Integration runs a mapping using a persistent lookup cache, it saves the cache files to disk. The next time that Data Integration runs the mapping, it builds the memory cache from the cache files.

Configure the Lookup transformation to use a persistent lookup cache in the transformation advanced properties. To use a persistent cache, enable the **Lookup Cache Persistent** property.

You can configure the following options when you use a persistent cache:

Specify a name for the cache files.

When you use a persistent lookup cache, you can specify a name for the cache files.

To specify a name, enter the file name prefix in the **Cache File Name Prefix** field on the **Advanced** tab of the Lookup transformation. Do not enter a suffix such as .idx or .dat.

Rebuild the lookup cache.

If the lookup table changes occasionally, you can configure the Lookup transformation to rebuild the lookup cache. When you do this, Data Integration rebuilds the lookup cache from the lookup source when it first calls the Lookup transformation instance.

To configure the transformation to rebuild the cache, enable the **Re-cache from Lookup Source** property on the **Advanced** tab of the Lookup transformation.

Rebuilding the lookup cache

You can rebuild the lookup cache if you think the lookup source changed since the last time Data Integration built the persistent cache.

When you rebuild a cache, Data Integration creates new cache files, overwriting existing persistent cache files. Data Integration writes a message to the session log when it rebuilds the cache.

If Data Integration cannot reuse the cache, it rebuilds the cache or fails the mapping task. The behavior can differ based on whether the cache is named or unnamed.

The following table summarizes how Data Integration handles named and unnamed persistent caches when the mapping changes between runs:

Mapping changes between runs	Named cache	Unnamed cache
Data Integration cannot locate cache files. For example, the file no longer exists.	Rebuilds cache	Rebuilds cache
Enable or disable the Enable High Precision option in the mapping task advanced session properties.	Fails mapping task	Rebuilds cache

Mapping changes between runs	Named cache	Unnamed cache
Edit the transformation in the Mapping Designer or Maplet Designer, excluding editing the transformation description.	Fails mapping task	Rebuilds cache
Edit the mapping, excluding the Lookup transformation.	Reuses cache	Rebuilds cache
Change database connection or the file location used to access the lookup table.	Fails mapping task	Rebuilds cache

Unconnected lookups

An unconnected Lookup transformation is a Lookup transformation that is not connected to other transformations in a mapping. A transformation in the mapping pipeline calls the Lookup transformation with a :LKP expression. The unconnected Lookup transformation returns one column to the calling transformation.

You can use an unconnected Lookup transformation to perform a lookup against the following types of data objects:

- Flat file
- Relational database
- Amazon Redshift V2
- Amazon S3 V2
- Google BigQuery V2
- Microsoft Azure Synapse SQL
- Snowflake Data Cloud

The following table lists the differences between connected and unconnected Lookup transformations:

Functionality	Connected lookup	Unconnected lookup
Input values	Receives input values directly from the mapping pipeline.	Receives input values from the result of a :LKP expression in another transformation.
Cache	Cache includes all lookup columns used in the mapping. This includes columns in the lookup condition and columns linked as output fields to other transformations. Can use static or dynamic cache.	Cache includes all lookup/output fields in the lookup condition and the lookup/return field. Cannot use dynamic cache.
Return values	Returns multiple values from the same row.	Returns the specified field for each row.

Functionality	Connected lookup	Unconnected lookup
Lookup conditions	<p>If there is no match for a lookup condition, Data Integration returns the default value for all output fields.</p> <p>If there is a match, Data Integration returns the results of the lookup condition for all lookup/output fields.</p>	<p>If there is no match for the lookup condition, Data Integration returns NULL.</p> <p>If there is a match, Data Integration returns the result of the lookup condition to the return field.</p>
Output values	Passes multiple output values to another transformation. Links lookup/output fields to another transformation.	Passes one output value to another transformation. The lookup/output/return field passes the value to the transformation that contains the :LKP expression.

Configuring an unconnected Lookup transformation

To configure an unconnected Lookup transformation, select the **Unconnected Lookup** option, add incoming fields, configure the lookup condition, and designate a return value. Then configure a lookup expression in a different transformation.

1. On the **General** tab of the Lookup transformation, enable the **Unconnected Lookup** option.
2. Create the incoming fields.
 On the **Incoming Fields** tab of the Lookup transformation, create an incoming field for each argument in the :LKP expression. For each lookup condition you plan to create, you need to add an incoming field to the Lookup transformation. You can create a different field for each condition, or use the same incoming field in more than one condition.
3. Designate a return value.
 You can pass multiple input values into a Lookup transformation and return one column of data. Data Integration can return one value from the lookup query. Use the return field to specify the return value.
4. Configure a lookup expression in another transformation.
 Supply input values for an unconnected Lookup transformation from a :LKP expression in a transformation that uses expressions such as an Expression, Aggregator, Filter, or Router transformation. The arguments are local input fields that match the Lookup transformation input fields used in the lookup condition.

Calling an unconnected lookup from another transformation

Supply input values for an unconnected Lookup transformation from a :LKP expression in another transformation such as an Expression transformation or Aggregator transformation. You can call the same lookup multiple times in one mapping. You cannot call an unconnected lookup from a Joiner or Java transformation.

Use the following syntax for a :LKP expression:

```
:LKP.<Lookup transformation name> (<argument>, <argument>, ...)
```

The arguments are local input fields that match the Lookup transformation input fields used in the lookup condition.

For example, the following expression passes the ITEM_ID and PRICE fields to an unconnected Lookup transformation named lkp_ItemPrices:

```
:LKP.lkp_ItemPrices (ITEM_ID, PRICE)
```

Use the following guidelines to write an expression that calls an unconnected Lookup transformation:

- The order in which you list each argument must match the order of the lookup conditions in the Lookup transformation.
- The datatypes for the fields in the expression must match the datatypes for the input fields in the Lookup transformation.
- The argument fields in the expression must be in the same order as the input fields in the lookup condition.
- If you call a connected Lookup transformation in a :LKP expression, Data Integration marks the mapping invalid.

Connected Lookup example

In the following example, the Lookup transformation performs a lookup on an Orders table and returns all the orders for a specific customer. You could also define the Lookup to return only the first order or last order for the customer.

First, you configure a Lookup Condition, which is an expression that identifies what rows to return from the lookup table. For example, create a Simple Lookup Condition to find all the records where the CUSTOMER_ID Lookup Field is equal to the Incoming Field, CUSTOMER_IN.

Based on this condition, the Lookup finds all the rows where the customer ID is equal to the customer number that is passed to the Lookup transformation.

You can also add multiple conditions. For example, if you add this condition, the Lookup returns only the orders that are greater than \$100.00.

O_ORDER_AMT > 100.00

The Lookup returns data only when all conditions are true.

Dynamic Lookup example

To configure a Lookup transformation to be dynamic, use a dynamic lookup cache.

A dynamic cache is helpful when the source table contains a large amount of data or it contains duplicate primary keys.

The following example illustrates the advantage of using a dynamic cache rather than a static cache when the source table includes duplicate primary keys.

You want to update your payroll table with data from your Human Resources department. The payroll table includes the following data, where ID is the primary key:

ID	Name	Location
1	Abhi	USA
2	Alice	UK

You create a mapping with a Lookup transformation and use the payroll table for the target and the lookup cache. You configure the cache to be dynamic because you expect the Human Resources department's table to contain duplicate keys.

In the mapping, you specify the Human Resources department's table to be the source. The source table includes the following data:

ID	Name	Location
1	Abhi	India
2	Alice	UK
3	James	Japan
3	James	USA

You create a mapping task to update the payroll table. When the mapping task begins, it creates the cache file that contains the rows in the target table. As the task processes the rows, it flags the first row as an update and it updates the cache. It flags the third row as an insert and inserts the row in the cache. It flags the fourth row as an update because the row exists in the cache.

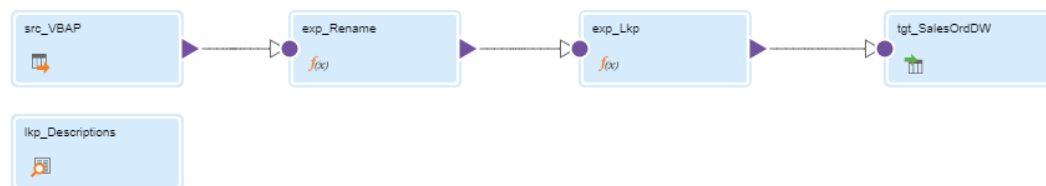
If you follow the same scenario using a static cache, the task flags the fourth row as an insert. The cache does not contain the row for James because it does not update as the task processes the rows. The target database produces an error because it cannot handle two rows with the same primary key.

Unconnected Lookup example

You can use an unconnected Lookup transformation to replace cryptic or numeric ID values in a table with meaningful names from a lookup table.

For example, you need to load some sales order data from transactional tables to a relational table in your data warehouse. The tables contain numeric IDs for values such as the sales group and sales office. In the data warehouse table, you want to replace the numeric IDs with the corresponding names in your local language. The name that is associated with each ID is stored in a reference table. Use an unconnected Lookup transformation to retrieve the names from the reference table.

The following image shows the mapping:



Configure the transformations in the following ways:

Source transformation

Use a Source transformation to specify the tables from which to extract data.

On the **Source** tab, configure the source connection and select the tables from which you want to extract data.

First Expression transformation (optional)

Optionally, use an Expression transformation to rename fields and replace null values.

On the **Incoming Fields** tab, use the **Named Fields** field selection criteria to select the fields that you want to load to the target table. If required, rename the selected fields to give them more meaningful names.

On the **Expression** tab, create output fields to replace the null values. For example, to replace null values for the sales group code and sales office code with spaces, you might create the following output fields:

Output Field	Expression
in_sales_group	IIF(ISNULL(sales_group_code), ' ', sales_group_code)
in_sales_office	IIF(ISNULL(sales_office_code), ' ', sales_office_code)

Unconnected Lookup transformation

Use an unconnected Lookup transformation to retrieve the descriptions from the reference table.

On the **General** tab, enable the **Unconnected Lookup** option.

On the **Incoming Fields** tab, create an incoming field for each value that you need to pass to the Lookup transformation to retrieve the data that you want. For example, to pass the domain name, language, and code value to the Lookup transformation, create the in_domain_name, in_language, and in_lookup_code fields.

On the **Lookup Object** tab, configure the lookup connection and select the reference table that you want to use as the lookup table.

On the **Lookup Condition** tab, specify the lookup condition for each incoming field. For example:

Lookup Field	Operator	Incoming Field
domain_name	=	in_domain_name
language_code	=	in_language
lookup_code	=	in_lookup_code

On the **Return Fields** tab, select the field in the reference table that you want to return. For example, to return a description, you might select lookup_description as the return field.

Second Expression transformation

Use an Expression transformation to call the unconnected Lookup transformation and retrieve the name that is associated with each ID value.

On the **Incoming Fields** tab, include all fields from the upstream transformation.

On the **Expression** tab, create an output field to retrieve each description from the Lookup transformation. Call the Lookup transformation with a :LKP expression. For example, to retrieve the

sales group and sales office names from the appropriate domain in English, you might create the following output fields:

Output Field	Expression
sales_group	:LKP.lkp_Descriptions('sales_group','en',in_sales_group)
sales_office	:LKP.lkp_Descriptions('sales_office','en',in_sales_office)

Target transformation

On the **Target** tab, configure the target connection and select the relational table to which you want to load data.

On the **Field Mapping** tab, map the output fields from the upstream transformation to the appropriate target fields. For example, to map the sales_group and sales_office output fields from the second Expression transformation to the SALES_GROUP and SALES_OFFICE target fields, configure the following field mapping:

Target Field	Mapped Field
SALES_GROUP	sales_group
SALES_OFFICE	sales_office

CHAPTER 10

Mapplet transformation

The Mapplet transformation inserts a mapplet that you created in Data Integration into a mapping. Each Mapplet transformation can contain one mapplet. You can add multiple Mapplet transformations to a mapping or mapplet.

The Mapplet transformation can be active or passive based on the transformation logic within the mapplet. An active mapplet includes at least one active transformation. An active mapplet can return a number of rows that is different from the number of source rows passed to the mapplet. A passive mapplet includes only passive transformations. A passive mapplet returns the same number of rows that are passed from the source.

Use a Mapplet transformation to accomplish the following goals:

Reuse transformation logic in different mappings.

For example, you have different fact tables that require a series of dimension keys. You create a mapplet that contains a series of Lookup transformations to find each dimension key. You include the mapplet in different fact table mappings instead of re-creating the lookup logic in each mapping.

Hide complex transformation logic.

The Mapplet transformation shows the mapplet incoming and outgoing fields. It does not display the transformations that the mapplet contains.

For more information about mapplets, see *Components*.

Mapplet transformation configuration

When you add a Mapplet transformation to a mapping, you must first select the mapplet that contains the transformation logic that you want to include in the mapping. If the mapplet includes one or more input groups, configure the incoming fields and field mappings. If the mapplet includes one or more output groups, verify the output fields.

To configure a Mapplet transformation, complete the following tasks:

1. Select the mapplet that you want to include in the transformation.
2. If the mapplet includes one or more input groups, configure the incoming fields.

By default, the transformation inherits all incoming fields from the upstream transformation. You can define a field rule to limit or rename the incoming fields. If the mapplet contains multiple input groups, configure incoming fields for each input group.

For information about configuring incoming fields for a transformation, see [“Incoming fields” on page 12](#).

3. If the mapplet includes one or more input groups, configure field mappings to define how data moves from the upstream transformation to the Mapplet transformation.
If the mapplet contains multiple input groups, configure the field mappings for each input group.
4. If the mapplet contains one or more output groups, verify the mapplet output fields on the **Output Fields** tab. Connect at least one output group to a downstream transformation.

Selecting a mapplet

Select the mapplet that you want to use in the Mapplet transformation on the **Mapplet** tab of the **Properties** panel.

1. In the **Properties** panel for the Mapplet transformation, click the **Mapplet** tab.
2. Click **Select**.
3. Open the project and folder that contains the mapplet and click **Select**.

The selected mapplet appears in the **Properties** panel.

If the mapplet that you select does not include a source, configure the incoming fields and field mappings after you select the mapplet.

If the mapplet that you select does not contain a target, configure the output fields and field mappings after you select the mapplet.

Mapplet transformation field mappings

Configure field mappings in a Mapplet transformation to define how data moves from the upstream transformation to the Mapplet transformation. Configure field mappings on the **Field Mapping** tab of the **Properties** panel.

Note: Mapped field names can have a maximum of 72 characters.

You can configure the following field mapping options:

Mapplet Input Group

The input group for which you want to configure field mappings. This option appears when the Mapplet transformation has multiple input groups.

Field Map Options

Method of mapping fields to the Mapplet transformation. Select one of the following options:

- **Manual.** Manually link incoming fields to Mapplet transformation input fields. Removes links for automatically mapped fields.
- **Automatic.** Automatically link fields with the same name. Use when all of the fields that you want to link share the same name. You cannot manually link fields with this option.

Options

Controls how fields are displayed in the **Incoming Fields** and **Mapplet Input Fields** lists. Configure the following options:

- The fields that appear. You can show all fields, unmapped fields, or mapped fields.
- Field names. You can use technical field names or labels.

Automap

Links fields with matching names. Allows you to link matching fields, and then manually configure other field mappings.

You can map fields in the following ways:

- **Exact Field Name.** Data Integration matches fields of the same name.
- **Smart Map.** Data Integration matches fields with similar names. For example, if you have an incoming field `Cust_Name` and a target field `Customer_Name`, Data Integration automatically links the `Cust_Name` field with the `Customer_Name` field.

You can use both Exact Field Name and Smart Map in the same field mapping. For example, use Exact Field Name to match fields with the same name and then use Smart Map to map fields with similar names.

You can undo all automapped field mappings by clicking **Automap > Undo Automap**. To unmap a single field, select the field to unmap and click **Actions > Unmap**.

Data Integration highlights newly mapped fields. For example, when you use Exact Field Name, Data Integration highlights the mapped fields. If you then use Smart Map, Data Integration only highlights the fields mapped using Smart Map.

Action menu

Additional field mapping options. Provides the following options:

- **Map Selected.** Links the selected incoming field with the selected mapplet input field.
- **Unmap Selected.** Clears the link for the selected field.
- **Clear Mapping.** Clears all field mappings.

Mapplet parameters

When you select a mapplet that contains parameters, the parameters are renamed in the Mapplet transformation. You can view the corresponding parameter names on the **Parameter** tab of the **Properties** panel.

In the Mapplet transformation, mapplet parameter names are prefixed with the Mapplet transformation name.

The following image shows the Parameters tab:

General	Parameters defined in the mapplet get renamed when used. The table below shows the parameter name defined in the mapplet, and its new name.		
Incoming Fields	Input Parameters		
Mapplet			
Field Mapping			
Parameters			
Output Fields	In-Out Parameters		

You can edit the properties of the maplet parameters, but you cannot change the parameter type or delete the parameters. To delete a parameter, open the maplet and remove the parameter.

To edit the parameter properties, click the new parameter name on the **Parameters** tab or on the **Parameters** panel. When you change the parameter properties in a Mapplet transformation, the changes do not affect the mapplet.

You can reuse the parameters in other transformations in the mapping.

Mapplet transformation output fields

When you select a mapplet that contains an Output transformation to use in a Mapplet transformation, the mapplet output fields appear on the **Output Fields** tab of the **Properties** panel.

The Mapping Designer displays the name, type, precision, scale, and origin for each output field in each output group. You cannot edit the transformation output fields. If you want to exclude output fields from the data flow or rename output fields before you pass them to a downstream transformation, configure the field rules in the downstream transformation.

Mapplet transformation names

Data Integration renames the transformations in a mapplet when you use the mapplet in a Mapplet transformation. If you reference a transformation in the mapplet in a downstream transformation, be sure to use the updated name.

The mapplet that you use in the Mapplet transformation might contain transformations with names that conflict with the names of transformations in the mapping. To avoid name conflicts with transformations in the mapping, Data Integration prefixes the names of transformations in the mapplet with the Mapplet transformation name at run time.

For example, a mapplet contains an Expression transformation named Expression_1. You create a mapping and use the mapplet in the Mapplet transformation Mapplet_Tx_1. When you run the mapping, the Expression transformation is renamed to Mapplet_Tx_1_Expression_1.

If the mapplet contains another Mapplet transformation, Data Integration also prefixes the transformation names with the second Mapplet transformation name. For example, the mapplet in the previous example also contains a Mapplet transformation named MappletTx, which contains FilterTx_1. In the mapping, FilterTx_1 is renamed to Mapplet_Tx_1_MappletTx_FilterTx_1.

In most cases, Data Integration truncates transformation names that contain more than 80 characters. Data Integration doesn't truncate the names of Mapplet transformations that contain Hierarchy Builder, Hierarchy Parser, or Structure Parser transformations. When you use a Hierarchy Builder, Hierarchy Parser, or Structure Parser transformation in a mapplet, to prevent runtime errors, be sure that the combined Mapplet transformation name doesn't exceed 80 characters.

Synchronizing a mapplet

If the interface of a mapplet changes after it has been added to a Mapplet transformation, you must synchronize the mapplet to get the changes. Synchronize a mapplet on the **Mapplet** tab.

Mappings and mapplets that use the mapplet are invalid until the mapplet is synchronized. If you run a mapping task that includes a changed mapplet, the task fails.

When you synchronize a mapplet, the updates might cause validation errors in other transformations in the mapping or mapplet.

To synchronize a mapplet, perform the following steps:

1. Open the mapping or mapplet that uses the mapplet.
2. Select the mapplet transformation.
3. On the **Mapplet** tab, click **Synchronize**.
4. Correct any resulting errors in the transformation logic.

CHAPTER 11

Normalizer transformation

The Normalizer transformation is an active transformation that transforms one incoming row into multiple output rows. When the Normalizer transformation receives a row that contains multiple-occurring data, it returns a row for each instance of the multiple-occurring data.

For example, a relational source includes four fields with quarterly sales data. You can configure a Normalizer transformation to generate a separate output row for each quarter.

When the Normalizer transformation returns multiple rows from an incoming row, it returns duplicate data for single-occurring incoming columns.

When you configure a Normalizer transformation, you define Normalizer properties on the following tabs of the **Properties** panel:

- **Normalized Fields** tab. Define the multiple-occurring fields and specify additional fields that you want to use in the mapping.
- **Field Mapping** tab. Connect the incoming fields to the normalized fields.

Normalized fields

Define the fields to be normalized on the **Normalized Fields** tab. You can also include other incoming fields that you want to use in the mapping.

When you define normalized fields, you can create fields manually or select fields from a list of incoming fields. When you create a normalized field, you can set the data type to String or Number, and then define the precision and scale.

When incoming fields include multiple-occurring fields without a corresponding category field, you can create the category field to define the occurs for the data. For example, to represent three fields with different types of income, you can create an Income category field and set the occurs value to 3.

Occurs configuration

Configure the occurs value for a normalized field to define the number of instances the field occurs in incoming data.

To define a multiple occurring field, set the occurs value for the field to an integer greater than one. When you set an occurs value to greater than one, the Normalizer transformation creates a generated column ID field for the field. The Normalizer transformation also creates a generated key field for all normalized data.

The Normalizer transformation also uses the occurs value to create a corresponding set of output fields. The output fields display on the **Field Mapping** tab of the Normalizer transformation. The naming convention for the output fields is <occurs field name>_<occurs number>.

To define a single-occurring field, set the occurs value for the field to one. Define a single-occurring field to include incoming fields that do not need to be normalized in the normalized fields list.

Unmatched groups of multiple-occurring fields

You can normalize more than one group of multiple-occurring fields in a Normalizer transformation. When you include more than one group and the occurs values do not match, configure the mapping to avoid validation errors.

Use one of the following methods to process groups of multiple-occurring fields with different occurs values.

Write the normalized data to different targets

You can use multiple-occurring fields with different occurs values when you write the normalized data to different targets.

For example, the source data includes an Expenses field with four occurs and an Income field with three occurs. You can configure the mapping to write the normalized expense data to one target and to write the normalized income data to a different target.

Use the same occurs value for multiple occurring fields

You can configure the multiple-occurring fields to use the same number of occurs, and then use the generated fields that you need. When you use the same number of occurs for multiple-occurring fields, you can write the normalized data to the same target.

For example, when the source data includes an Expenses field with four occurs and an Income field with three occurs, you can configure both fields to have four occurs.

When you configure the Normalizer field mappings, you can connect the four expense fields and the three income fields, leaving the unnecessary income output field unused. Then, you can configure the mapping to write all normalized data to the same target.

Generated keys

The Normalizer transformation generates key values for normalized data.

Generated keys fields appear on the **Normalized Fields** tab when you configure the field to have more than one occurrence.

The mapping task generates the following fields for normalized data.

Generated Key

A key value that the task generates each time it processes an incoming row. When a task runs, it starts the generated key with one and increments by one for each processed row.

The Normalizer transformation uses one generated key field for all data to be normalized.

The naming convention for the Normalizer generated key is GK_<redefined_field_name>.

Generated Column ID

A column ID value that represents the instance of the multiple-occurring data. For example, if an Expenses field that includes four occurs, the task uses values 1 through 4 to represent each type of occurring data.

The Normalizer transformation uses a generated column ID for each field configured to occur more than one time.

The naming convention for the Normalizer generated key is GCID_<redefined_field_name>.

Normalizer field mapping

Map incoming fields to normalized fields on the **Field Mapping** tab of the Normalizer transformation.

When you configure the Normalizer field mappings, complete the following steps:

1. Map the multiple-occurring incoming fields that you want to normalize to the corresponding output fields that the Normalizer transformation created.

Note: Map at least one output field for each set of normalized fields.

2. Map incoming fields to all normalized fields with a single occurrence.

Normalizer field mapping options

You can use the following field mapping options on the Normalizer **Field Mapping** tab.

Show Fields

Controls the fields that appear in the **Incoming Fields** list. Show all fields, unmapped fields, or mapped fields.

Automap

Links fields with matching names. Allows you to link matching fields, and then manually configure other field mappings.

You can map fields in the following ways:

- **Exact Field Name.** Data Integration matches fields of the same name.
- **Smart Map.** Data Integration matches fields with similar names. For example, if you have an incoming field `Cust_Name` and a target field `Customer_Name`, Data Integration automatically links the `Cust_Name` field with the `Customer_Name` field.

You can use both Exact Field Name and Smart Map in the same field mapping. For example, use Exact Field Name to match fields with the same name and then use Smart Map to map fields with similar names.

You can undo all automapped field mappings by clicking **Automap > Undo Automap**. To unmap a single field, select the field to unmap and click **Actions > Unmap**.

Data Integration highlights newly mapped fields. For example, when you use Exact Field Name, Data Integration highlights the mapped fields. If you then use Smart Map, Data Integration only highlights the fields mapped using Smart Map.

Action menu

Additional field link options. Provides the following options:

- **Map Selected.** Links the selected incoming field with the selected target field.
- **Unmap Selected.** Clears the link for the selected field.
- **Clear Mapping.** Clears all field mappings.

Advanced properties

You can configure advanced properties for a Normalizer transformation. The advanced properties control settings such as the tracing level for session log messages and whether the transformation is optional or required.

You can configure the following properties:

Property	Description
Tracing Level	Detail level of error and status messages that Data Integration writes in the session log. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Optional	<p>Determines whether the transformation is optional. If a transformation is optional and there are no incoming fields, the mapping task can run and the data can go through another branch in the data flow. If a transformation is required and there are no incoming fields, the task fails.</p> <p>For example, you configure a parameter for the source connection. In one branch of the data flow, you add a transformation with a field rule so that only Date/Time data enters the transformation, and you specify that the transformation is optional. When you configure the mapping task, you select a source that does not have Date/Time data. The mapping task ignores the branch with the optional transformation, and the data flow continues through another branch of the mapping.</p>

Target configuration for Normalizer transformations

When you use a Normalizer transformation in a mapping, consider the following configuration guidelines for targets and Target transformations:

- To write normalized data to the target, map the multiple occurring field to a target field in the Target transformation field mapping.
- To include generated keys or generated column IDs in target data, create additional target fields as required, and then map the fields in the Target transformation field mapping.

Normalizer field rule for parameterized sources

When you use a parameter as the source object to provide data for a Normalizer transformation, use a **Named Fields** field rule in the Normalizer to define incoming fields.

When you configure a Normalizer transformation, you define the field mappings between the incoming fields and the normalized fields. When you use a parameter for a source object, field names do not appear in the list of incoming fields until you use a Named Field rule to add fields.

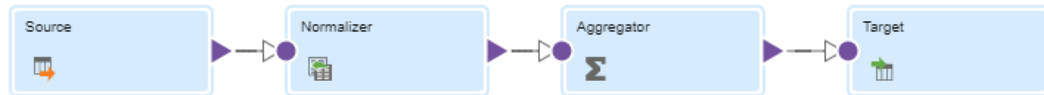
To add fields, on the **Incoming Fields** tab of the Normalizer transformation, create a **Named Fields** rule. In the **Include Named Fields** dialog box, click **Add** and enter the name of an incoming field.

Create fields to represent all of the source fields that you want to use in the Normalizer transformation.

Mapping example with a Normalizer and Aggregator

A relational table includes the quarterly unit sales for each store. To calculate the annual unit sales for each store, use a Normalizer transformation to create a row for each quarter. Then, use an Aggregator transformation to aggregate the quarterly sales for each store.

The following image shows the mapping to create:



The source data includes the following rows:

StoreNo	Q1	Q2	Q3	Q4	Year
1001	30	50	48	80	2014
1022	100	120	125	140	2014
1190	80	108	123	134	2014

Use the Normalizer transformation to pivot source data before the data passes to the Aggregator transformation, as follows:

StoreNo	QuarterlySales
1001	30
1001	50
1001	48
1001	80
1022	100
1022	120
1022	125
1022	140
1190	80
1190	108
1190	123
1190	134

Define the normalized fields

On the **Normalized Fields** tab, create a normalized field called "QuarterlySales." To indicate that this field represents four fields, set the occurs value to four.

To include the store number data in the mapping, from the menu, select **Generate From Incoming Fields** and select **StoreNo**. Use the default occurs value of one because this field does not include multiple-occurring data.

The following image shows the **Normalized Field** tab after adding both fields:

Notice that when you set the QuarterlySales occurs value to four, the Normalizer creates the generated column ID field and the generated key field.

Configure the Normalizer field mappings

On the **Field Mapping** tab of the Normalizer transformation, connect the incoming fields to the normalized fields.

In the **Normalized Fields** list, the Normalizer replaces the multiple-occurring QuarterlySales field with corresponding fields to hold the normalized data: QuarterlySales_1, QuarterlySales_2, QuarterlySales_3, and QuarterlySales_4. The list also includes the StoreNo field.

Connect the incoming fields to the StoreNo and QuarterlySales normalized fields as follows:

The screenshot shows the 'nrm_NormalizeQuarterlySales Properties' dialog with the 'Field Mapping' tab selected. The 'Incoming Fields' list contains StoreNo, Q1, Q2, Q3, Q4, and Year. The 'Normalized Fields' list contains QuarterlySales_1, QuarterlySales_2, QuarterlySales_3, QuarterlySales_4, and StoreNo. The 'Mapped Field' column shows the mapping: Q1 to QuarterlySales_1, Q2 to QuarterlySales_2, Q3 to QuarterlySales_3, Q4 to QuarterlySales_4, and StoreNo to StoreNo.

Incoming Fields: (5 of 6 mapped)	Normalized Fields: (5 of 5 mapped)
Field Name	Field Name
StoreNo	QuarterlySales_1
Q1	QuarterlySales_2
Q2	QuarterlySales_3
Q3	QuarterlySales_4
Q4	StoreNo
Year	

Configure the Aggregator transformation

To calculate the annual sales by store, add an Aggregator transformation to the mapping and connect the Normalizer to the Aggregator.

In the Aggregator transformation, use the default All Fields rule to pass all fields from the Normalizer to the Aggregator.

To group data by store number, add a group by field on the **Group By** tab, and then select the **StoreNo** field.

The following image shows the **Group By** tab with the StoreNo group by field:

The screenshot shows the 'agg_SalesByStore Properties' dialog with the 'Group By' tab selected. The 'Group by' dropdown is set to 'Not Parameterized'. The 'Group by Fields' list contains StoreNo.

Field Name	Actions
StoreNo	

On the **Aggregate** tab, create a Decimal output field named AnnualSales_byStore. To configure the output field, use the QuarterlySales field in the following aggregate expression: `SUM(QuarterlySales)`. The QuarterlySales field represents all of the normalized quarterly data.

The following image shows the **Aggregate** tab with the AnnualSales_byStore output field:

agg_SalesByStore Properties

General

Create simple aggregate expressions. You can also use expression macros to create complex aggregate expressions.

☐ Allow additional fields and expressions during task creation

Incoming Fields

Group By

Aggregate

Advanced

Field Name	Expression	Actions
AnnualSales_byStore	SUM(QuarterlySales)	

Configure the Target

Add a Target transformation and connect the Aggregator transformation to the Target transformation.

Use the default All Fields rule to pass all fields from the Aggregator to the Target transformation.

On the **Target** tab, select the target connection and the target object.

On the **Field Mapping** tab, the incoming fields list includes the AnnualSales_byStore field created in the Aggregator transformation, and the StoreNo field that passed through the mapping from the source.

The incoming fields list also includes the QuarterlySales and generated key columns created by the Normalizer. These fields do not need to be written to the target.

Connect the StoreNo and AnnualSales_byStore fields to corresponding target fields.

The following image shows the configured **Field Mapping** tab:

tgt_SalesByStore Properties

General

Field map options: Manual

Options

Incoming Fields

Target

Target Fields

Field Mapping

Automatch

Field Name	Mapped Field
AnnualSales_byStore	StoreNo
QuarterlySales	AnnualSales_byStore

Task results

When you run the task, the mapping task normalizes the source data, creating one row for each quarter. The task groups the normalized data by store, and then aggregates the quarterly unit sales for each store.

The task writes the following data to the target:

StoreNo	SalesbyStore
1001	208
1022	485
1190	445

CHAPTER 12

Output transformation

The Output transformation is a passive transformation that you use to pass data from a mapplet to a downstream transformation.

Add output fields to the Output transformation to define the data fields you want to pass from the mapplet. You must add at least one output field to each output transformation. You can add multiple output transformations to a mapplet. Each Output transformation becomes an output group when you use the mapplet in a Mapplet transformation. You must connect at least one output group to a downstream transformation. You can connect an output group to multiple downstream transformations.

You can use an Output transformation in a mapplet but not in a mapping.

Output fields

Add output fields to an Output transformation to define the data fields you want to pass from the Mapplet to the downstream transformation. You must add at least one output field to each Output transformation.

Add output fields on the **Output Fields** tab of the properties panel. To add a field, click **Add Field**, and then enter the field name, data type, precision, and scale.

When you use the mapplet in a Mapplet transformation, map at least one output field to the downstream transformation.

Field mapping

Map fields to configure how data moves from the upstream transformation to the Output transformation.

Configure field mappings on the **Field Mapping** tab.

The **Field Mapping** tab includes a list of incoming fields and a list of target fields.

Note: Mapped field names can have a maximum of 72 characters.

You can configure the following field mapping options:

Field Map Options

Method of mapping fields to the Mapplet transformation. Select one of the following options:

- **Manual.** Manually link incoming fields to Mapplet transformation input fields. Removes links for automatically mapped fields.

- **Automatic.** Automatically link fields with the same name. Use when all of the fields that you want to link share the same name. You cannot manually link fields with this option.

Options

Controls how fields are displayed in the **Incoming Fields** and **Output Fields** lists. Configure the following options:

- **The fields that appear.** You can show all fields, unmapped fields, or mapped fields.
- **Field names.** You can use technical field names or labels.

Automap

Links fields with matching names. Allows you to link matching fields, and then manually configure other field mappings.

You can map fields in the following ways:

- **Exact Field Name.** Data Integration matches fields of the same name.
- **Smart Map.** Data Integration matches fields with similar names. For example, if you have an incoming field `Cust_Name` and a target field `Customer_Name`, Data Integration automatically links the `Cust_Name` field with the `Customer_Name` field.

You can use both Exact Field Name and Smart Map in the same field mapping. For example, use Exact Field Name to match fields with the same name and then use Smart Map to map fields with similar names.

You can undo all automapped field mappings by clicking **Automap > Undo Automap**. To unmap a single field, select the field to unmap and click **Actions > Unmap**.

Data Integration highlights newly mapped fields. For example, when you use Exact Field Name, Data Integration highlights the mapped fields. If you then use Smart Map, Data Integration only highlights the fields mapped using Smart Map.

Action menu

Additional field mapping options. Provides the following options:

- **Map Selected.** Links the selected incoming field with the selected mapplet input field.
- **Unmap Selected.** Clears the link for the selected field.
- **Clear Mapping.** Clears all field mappings.

CHAPTER 13

Rank transformation

The Rank transformation selects the top or bottom range of data. Use the Rank transformation to return the largest or smallest numeric values in a group. You can also use the Rank transformation to return strings at the top or bottom of the mapping sort order.

For example, you can use a Rank transformation to select the top 10 customers by region. Or, you might identify the three departments with the lowest expenses in salaries and overhead.

The Rank transformation differs from the transformation functions MAX and MIN because the Rank transformation returns a group of values, not just one value. While the SQL language provides many functions designed to handle groups of data, identifying top or bottom strata within a set of rows is not possible using standard SQL functions.

The Rank transformation is an active transformation because it can change the number of rows that pass through it. For example, you configure the transformation to select the top 10 rows from a source that contains 100 rows. In this case, 100 rows pass into the transformation but only 10 rows pass from the Rank transformation to the downstream transformation or target.

When you run a mapping that contains a Rank transformation, Data Integration caches input data until it can perform the rank calculations.

Ranking string values

You can configure the Rank transformation to return the top or bottom values of a string field. To sort string fields, Data Integration uses the session sort order configured for the mapping task.

You set the **Session Sort Order** property in the advanced session properties for the mapping task. You can select binary or a specific language such as Danish or Spanish. If you select binary, Data Integration calculates the binary value of each string and sorts the strings using the binary values. If you select a language, Data Integration sorts the strings alphabetically using the sort order for the language.

The following image shows the **Session Sort Order** property in the advanced session properties for a mapping task:

Advanced Options

Preprocessing Commands:

Post-processing Commands:

Parameter File Name:

Maximum Number of Log Files: 10

Execution Mode

Mode: ☒ Standard ☐ Verbose

Advanced Session Properties

Remove	Session Property Name*	Session Property Value*
<input type="checkbox"/>	Session Sort Order	BINARY

☒ Enable cross-schema pushdown optimization

Save < Back Next > Finish Cancel

Rank caches

When you run a mapping that contains a Rank transformation, Data Integration creates data and index cache files to run the transformation. By default, Data Integration stores the cache files in the directory entered in the Secure Agent \$PMCacheDir property for the Data Integration Server.

You can change the cache directory and cache sizes on the **Advanced** tab of the Rank transformation.

Data Integration creates the following caches for the Rank transformation:

- Data cache that stores row data based on the group by fields.
- Index cache that stores group values as configured in the group by fields.

When you run a mapping that contains a Rank transformation, Data Integration compares an input row with rows in the data cache. If the input row out-ranks a cached row, Data Integration replaces the cached row with the input row. If you configure the Rank transformation to rank across multiple groups, Data Integration ranks incrementally for each group that it finds.

Defining a Rank transformation

To define a Rank transformation, you drag the transformation into the Mapping Designer, configure fields, and configure the transformation properties.

1. In the Mapping Designer, drag a Rank transformation from the transformation palette onto the canvas and connect it to the upstream and downstream transformations.

2. Configure the transformation fields.

By default, the transformation inherits all incoming fields from the upstream transformation. If you do not need to use all of the incoming fields, you can configure field rules to include or exclude certain fields.

3. Configure the rank properties.

Select the field that you want to rank by, the rank order, and the number of rows to rank.

4. Optionally, configure rank groups.

You can configure the Rank transformation to create groups for ranked rows.

5. Optionally, configure the transformation advanced properties.

You can update the cache properties, tracing level for log messages, transformation scope, case-sensitivity for string comparisons, and whether the transformation is optional.

Rank transformation fields

A Rank transformation inherits incoming fields from the upstream transformation. When you create a Rank transformation, Data Integration also creates a RANKINDEX output field.

The Rank transformation uses the following fields:

Incoming fields

Incoming fields appear on the **Incoming Fields** tab. By default, the Rank transformation inherits all incoming fields from the upstream transformation. If you do not need to use all of the incoming fields, you can define field rules to include or exclude certain fields. For more information about field rules, see [“Field rules” on page 13](#).

RANKINDEX

After the Rank transformation identifies all rows that belong to a top or bottom rank, it assigns rank index values. Data Integration creates the RANKINDEX field to store the rank index value for each row in a group.

For example, you create a Rank transformation to identify the five retail stores in the company with the highest monthly gross sales. The store with the highest sales receives a rank index of 1. The store with the next highest sales receives a rank index of 2, and so on. If two stores have the same gross sales, they receive the same rank index, and the transformation skips the next rank index.

For example, in the following data set, the Long Beach and Anaheim stores have the same gross sales, so they are assigned the same rank index:

RANKINDEX	STORE	GROSS_SALES
1	Long Beach	100000

RANKINDEX	STORE	GROSS_SALES
1	Anaheim	100000
3	Riverside	90000
4	Chula Vista	80050

When measuring a bottom rank, such as the 10 lowest priced products in the inventory, the Rank transformation assigns a rank index from lowest to highest. Therefore, the least expensive item receives a rank index of 1.

The RANKINDEX is an output field. It appears on the **Incoming Fields** tab of the downstream transformation.

Defining rank properties

When you define the rank properties for a Rank transformation, you select the field to rank by, specify the rank order, and specify number of rows to rank by. Define rank properties on the **Rank** tab.

The following image shows the **Rank** tab:

The screenshot shows the 'mk_EmpBySalary Properties' dialog box with the 'Rank' tab selected. The 'Rank By' field is set to 'EMP_SALARY', 'Rank Order' is 'Top', and 'Number of Rows' is '10'.

Configure the following properties:

Rank By

Specify the field that you want to use for ranking in the **Rank By** field.

For example, you create a Rank transformation to rank the top 10 employees in each department based on salary. The EMP_SALARY field contains the salary for each employee. Select EMP_SALARY as the **Rank By** field.

Rank Order

Specify the rank order in the **Rank Order** field. Select **Top** or **Bottom**.

Number of Rows

Specify the number of rows to include in each rank group in the **Number of Rows** field. For example, to rank the top 10 employees in each department based on salary, enter 10 in the **Number of Rows** field.

You can use a parameter to specify the number of rows. To use a parameter, select **Parameterized** in the **Parameterize Number of Rows** field, and enter the parameter in the **Parameter** field.

Defining rank groups

You can configure the Rank transformation to define groups for ranked rows. For example, to select the 10 most expensive items by manufacturer, define a rank group for the manufacturer. Define rank groups on the **Group By** tab.

To define rank groups, select one or more incoming fields as **Group By Fields**. For each unique value in a rank group, the transformation creates a group of rows that fall within the rank definition (top or bottom, and number in each rank).

For example, you create a Rank transformation that ranks the top five salespersons grouped by quarter. The rank index numbers the salespeople from 1 to 5 for each quarter as follows:

RANKINDEX	SALES_PERSON	SALES	QUARTER
1	Alexandra B.	10000	1
2	Boris M.	9000	1
3	Chanchal R.	8000	1
4	Dong T.	7000	1
5	Elias M.	6000	1
1	Elias M.	11000	2
2	Boris M.	10000	2
3	Alexandra B.	9050	2
4	Dong T.	7500	2
5	Frances Z.	6900	2

If you define multiple rank groups, the Rank transformation groups the ranked rows in the order in which the fields are selected in the **Group By Fields** list.

Advanced properties

Configure advanced properties to define how the Rank transformation processes data. Configure advanced properties on the **Advanced** tab.

Configure the following properties:

Property	Description
Cache Directory	Directory where Data Integration creates the data cache and index cache files. By default, Data Integration stores the cache files in the directory entered in the Secure Agent \$PMCacheDir property for the Data Integration Server. If you change the cache directory, verify that the directory exists and contains enough disk space for the cache files. Default is \$PMCacheDir.
Rank Data Cache Size	Data cache size for the transformation. Select one of the following options: <ul style="list-style-type: none">- Auto. Data Integration sets the cache size automatically. If you select Auto, you can also configure a maximum amount of memory for Data Integration to allocate to the cache.- Value. Enter the cache size in bytes. Default is Auto.
Rank Index Cache Size	Index cache size for the transformation. Select one of the following options: <ul style="list-style-type: none">- Auto. Data Integration sets the cache size automatically. If you select Auto, you can also configure a maximum amount of memory for Data Integration to allocate to the cache.- Value. Enter the cache size in bytes. Default is Auto.
Tracing Level	Detail level of error and status messages that Data Integration writes in the session log. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Transformation Scope	The method in which Data Integration applies the transformation logic to incoming data. Select one of the following values: <ul style="list-style-type: none">- Transaction. Applies the transformation logic to all rows in a transaction. Choose Transaction when the results of the transformation depend on all rows in the same transaction, but not on rows in other transactions.- All Input. Applies the transformation logic to all incoming data. When you choose All Input, Data Integration drops transaction boundaries. Select All Input when the results of the transformation depend on all rows of data in the source. Default is All Input.
Case Sensitive String Comparison	Specifies whether Data Integration uses case-sensitive string comparisons when it ranks strings. To ignore case in strings, disable this option. Default is enabled.
Optional	Determines whether the transformation is optional. If a transformation is optional and there are no incoming fields, the mapping task can run and the data can go through another branch in the data flow. If a transformation is required and there are no incoming fields, the task fails. For example, you configure a parameter for the source connection. In one branch of the data flow, you add a transformation with a field rule so that only Date/Time data enters the transformation, and you specify that the transformation is optional. When you configure the mapping task, you select a source that does not have Date/Time data. The mapping task ignores the branch with the optional transformation, and the data flow continues through another branch of the mapping. Default is enabled.

Rank transformation example

You store customer data in a relational database table. You want send a promotion to the top three customers in each tier. Use a Rank transformation to rank the customers in each tier by order amount.

The source, mapping, and target are configured as follows:

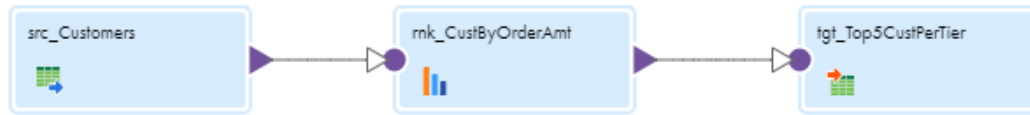
Source Data

The following table shows the source data:

CUST_ID	CUST_TIER	CUST_NAME	ORDER_AMT
10110102	Gold	Brosseau, Derrick	63508.12
10110109	Platinum	Acheson, Jeff	139824.15
10110143	Silver	Cudell, Bob	49614.00
10110211	Silver	Amico, Paul	47677.30
10110215	Platinum	Bergeron, Kim	148871.25
10110224	Silver	Madison, Shelley	40497.10
10110235	Gold	Anderson, Rick	50429.27
10110236	Silver	Tucker, Paul	42585.00
10110237	Silver	Smith, Robert	38563.98
10110393	Gold	Washington, Rochelle	73767.96
10110425	Gold	Nguyen, Trang	65522.25
10110434	Silver	Keane, Thomas	38055.40
10110436	Platinum	Catherwood, Jennifer	117107.44
10110442	Platinum	Charest, Walter	126618.60
10110458	Gold	Coutts, Sylvain	70646.32
10110497	Platinum	Zheng, Wei	191422.00
10110506	Gold	Gonzales, Roberto	79342.90
10110526	Gold	Vanelo, Susan	81978.06
10110528	Platinum	Abedini, John	136506.32
10110530	Silver	Sousa, Maria	10155.42

Mapping Configuration

Configure the mapping as shown in the following image:



Configure the Rank transformation as follows:

Rank tab

Configure the following properties:

Field	Value
Rank By	ORDER_AMT
Rank Order	Top
Parameterize Number of Rows	Not Parameterized
Number of Rows	3

Group By tab

Select CUST_TIER as the group by field.

Target Data

The following table shows the data that is written to the target when you run the mapping:

RANKINDEX	CUST_ID	CUST_TIER	CUST_NAME	ORDER_AMT
1	10110526	Gold	Vanelo, Susan	81978.06
2	10110506	Gold	Gonzales, Roberto	79342.90
3	10110393	Gold	Washington, Rochelle	73767.96
1	10110497	Platinum	Zheng, Wei	191422.00
2	10110215	Platinum	Bergeron, Kim	148871.25
3	10110109	Platinum	Acheson, Jeff	139824.15
1	10110143	Silver	Cudell, Bob	49614.00
2	10110211	Silver	Amico, Paul	47677.30
3	10110236	Silver	Tucker, Paul	42585.00

CHAPTER 14

Router transformation

The Router transformation is an active transformation that you can use to apply a condition to incoming data.

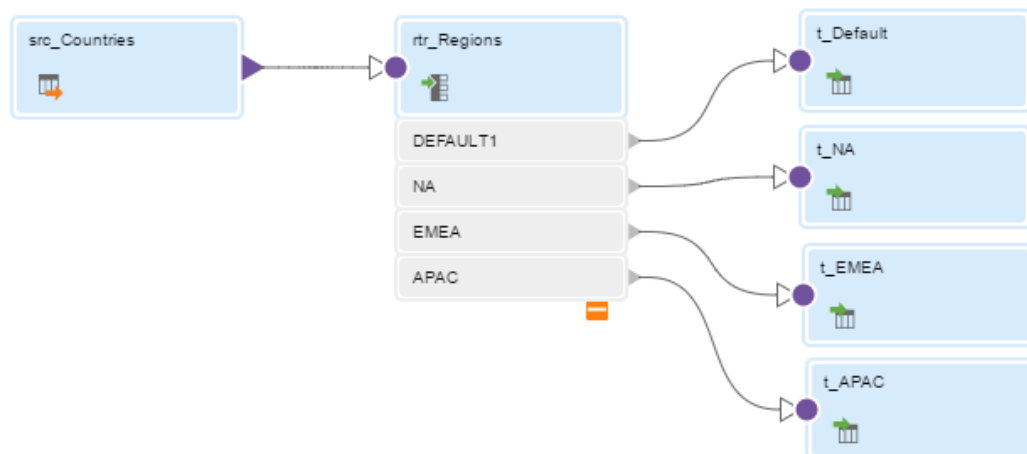
In a Router transformation, Data Integration uses a filter condition to evaluate each row of incoming data. It tests the conditions of each user-defined group before processing the default group. If a row meets more than one group filter condition, Data Integration passes the row multiple times. You can either drop rows that do not meet any of the conditions or route those rows to a default output group.

If you need to test the same input data based on multiple conditions, use a Router transformation in a mapping instead of creating multiple Filter transformations to perform the same task.

The following table compares the Router transformation to the Filter transformation:

Options	Router	Filter
Conditions	Test for multiple conditions in a single Router transformation	Test for one condition per Filter transformation
Handle rows that do not meet the condition	Route rows to the default output group or drop rows that do not meet the condition	Drop rows that do not meet the condition
Incoming data	Process once with a single Router transformation	Process in each Filter transformation

The following figure shows a mapping with a Router transformation that filters data based on region and routes it to a different target, either NA, EMEA, or APAC. The transformation routes data for other regions to the default target:



Working with groups

You use groups in a Router transformation to filter the incoming data.

Data Integration uses the filter conditions to evaluate each row of incoming data. It tests the conditions of each user-defined group before processing the default group. If a row meets more than one group filter condition, Data Integration passes the row to multiple groups.

A Router transformation has the following types of groups:

Input

Data Integration copies properties from the input group fields to create the fields for each output group.

Output

There are two types of output groups:

- User-defined groups. Create a user-defined group to test a condition based on incoming data. A user-defined group consists of output ports and a group filter condition. Create one user-defined group for each condition that you want to specify. Data Integration processes user-defined groups that are connected to a transformation or a target.
- Default group. The default group captures rows that do not satisfy any group condition. You cannot edit, delete, or define a group filter condition for the default group. If all of the conditions evaluate to FALSE, Data Integration passes the row to the default group. If you want to drop rows that do not satisfy any group condition, do not connect the default group to a transformation or a target.

You can modify a user-defined output group name. Click the row to open the **Edit Output Group** dialog box.

Guidelines for connecting output groups

Note the following guidelines when you connect Router transformation output groups to downstream transformations:

- You can connect each output group to one or more transformations or targets.
- You cannot connect more than one group to one target or a single input group transformation.
- You can connect more than one output group to a downstream transformation if you connect each output group to a different input group.
- If you want Data Integration to drop all rows in the default group, do not connect it to a transformation or a target in a mapping.

Group filter conditions

You can test data based on one or more group filter conditions. You can enter any expression that returns a single value. You can also specify a constant for the condition.

A group filter condition returns TRUE or FALSE for each row that passes through the transformation, depending on whether a row satisfies the specified condition. Zero (0) is the equivalent of FALSE, and any non-zero value is the equivalent of TRUE.

When the task runs, Data Integration handles the data in the following ways:

- Passes the rows of data that evaluate to TRUE to each transformation or target that is associated with each user-defined group.

The Router transformation can pass data through multiple output groups. For example, if the data meets three output group conditions, the Router transformation passes the data through three output groups.

- Passes the row to the default group if all of the conditions evaluate to FALSE.

You cannot configure a group filter condition for the default group. However, you can add an Expression transformation to perform a calculation and handle the rows in the default group.

Configuring a group filter condition

Configure a group filter condition for each user-defined output group.

1. Connect the source to the Router transformation.
2. Click the **Output Groups** tab.
3. Click the + sign and add the group you want to configure.
4. Click **Configure**.
5. In the **Edit Filter Condition** dialog box, select one of the following filter condition types:
 - **Simple**. Enter a simple condition based on available field names and operators.
 - **Advanced**. Select this option to open the Expression Editor and validate the syntax for a complex condition.
 - **Completely Parameterized**. Select this option to base the filter condition on an input parameter.

The following image shows the **Edit Filter Condition** dialog box:

Edit Filter Condition for NA

Filter Condition: Simple

Filter Conditions

Field Name	Operator	Value
Region	=	NA

OK Cancel

6. Click the + sign to add a row for each condition that you want to apply to this group.
7. Choose a Field Name, Operator, and Value for each condition.
8. Click **OK** to save the conditions.

Advanced properties

You can configure advanced properties for a Router transformation. The advanced properties control settings such as the tracing level for session log messages and whether the transformation is optional or required.

You can configure the following properties:

Property	Description
Tracing Level	Detail level of error and status messages that Data Integration writes in the session log. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Optional	<p>Determines whether the transformation is optional. If a transformation is optional and there are no incoming fields, the mapping task can run and the data can go through another branch in the data flow. If a transformation is required and there are no incoming fields, the task fails.</p> <p>For example, you configure a parameter for the source connection. In one branch of the data flow, you add a transformation with a field rule so that only Date/Time data enters the transformation, and you specify that the transformation is optional. When you configure the mapping task, you select a source that does not have Date/Time data. The mapping task ignores the branch with the optional transformation, and the data flow continues through another branch of the mapping.</p>

Router transformation examples

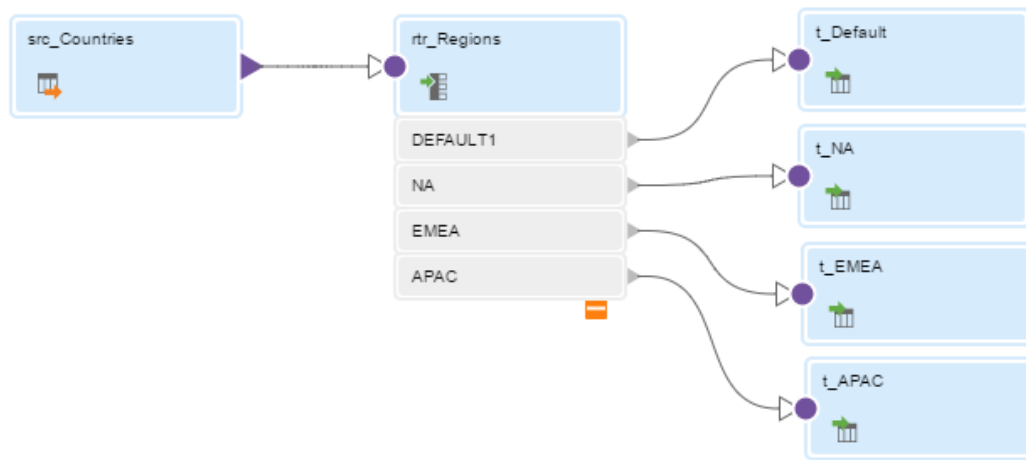
You can use a Router transformation to complete the following tasks:

- Group data by different country attributes and route each group to different target tables based on conditions that test for the region.
- Group inventory items by different price categories and route each group to different target tables based on conditions that test for low, medium, and high prices.

Example 1: Route data to different targets by region.

Your source includes data for customers in different regions. You want to configure a marketing campaign with variants for customers in the North America region, the Europe, Middle East, and Africa region, and the Asia Pacific region. All other customers see the default ad campaign. Use a Router transformation to route the data to four different Target transformations.

The following figure shows a mapping with a Router transformation that filters data based on these conditions:



Create three output groups and specify the group filter conditions on the **Output Groups** tab as shown in the following table:

Group Name	Condition
NA	region = 'NA'
EMEA	region = 'EMEA'
APAC	region = 'APAC'

The default group includes data for all customers that are not in the NA, EMEA, or APAC region.

Example 2: Route some rows to multiple output groups.

The Router transformation passes data through all output groups that meet the filter condition. In the following example, the conditions test for a price threshold, but the filter conditions for the two output groups overlap:

Group Name	Condition
PriceGroup1	item_price > 100
PriceGroup2	item_price > 500

When the Router transformation processes an input row with item_price=510, it routes the row to both output groups.

If you want to pass the data through a single output group, define the filter conditions so that they do not overlap. For example, you might change the filter condition for PriceGroup1 to item_price <= 500.

CHAPTER 15

Sequence Generator transformation

The Sequence Generator transformation is a passive and connected transformation that generates numeric values. Use the Sequence Generator to create unique primary key values, replace missing primary keys, or cycle through a sequential range of numbers.

The Sequence Generator transformation contains pass-through fields and two output fields, NEXTVAL and CURRVAL.

The mapping task generates a numeric sequence of values each time the mapped fields enter a connected transformation. You set the range of numbers in the Mapping Designer. You can change the initial number in the sequence when you run the task.

After the task completes, you can see the current value and the initial value for a Sequence Generator transformation in the mapping task details.

Sequence Generator transformation uses

Use a Sequence Generator transformation to generate a sequence of numbers in the following ways:

Generate a sequence of unique numbers.

The sequence begins with the Initial Value that you specify.

Generate a cyclic sequence of numbers.

You can establish a range of values for the Sequence Generator transformation. If you use the cycle option, the Sequence Generator transformation repeats the range when it reaches the end value. For example, if you set the sequence range to start at 10 and end at 50, and you set an increment value of 10, the Sequence Generator transformation generates the following values: 10, 20, 30, 40, 50. The sequence starts over again at 10.

Continue an existing sequence of numbers.

Each time you run the mapping task, the task updates the value to reflect the last-generated value plus the Increment By value. If you want the numbering to start over each time you run the task, you can enable the Reset configuration property.

Sequence Generator output fields

The Sequence Generator transformation has two output fields, NEXTVAL and CURRVAL. You cannot edit or delete these fields.

You can connect a Sequence Generator transformation to any transformation. If the mapping contains both fields, you do not need to map both of the output fields. If you do not map one of the output fields, the mapping task ignores the unmapped field.

NEXTVAL field

Use the NEXTVAL field to generate a sequence of numbers based on the Initial Value and Increment By properties.

Map the NEXTVAL field to an input field in a Target transformation or other downstream transformation to generate a sequence of numbers. If you do not configure the Sequence Generator to cycle through the sequence, the NEXTVAL field generates sequence numbers up to the configured End Value.

If you map the NEXTVAL field to multiple transformations, the mapping task generates the same sequence or a unique sequence of numbers for each downstream transformation based on the mapping type and whether incoming fields are disabled.

The following table lists the situations where the Sequence Generator transformation generates the same sequence or a unique sequence of numbers:

Mapping type	Incoming fields are...	Same sequence or unique sequence?
Mapping	Not disabled	Same sequence
Mapping	Disabled	Unique sequence*
* To generate the same sequence of numbers when incoming fields are disabled, you can place an Expression transformation between the Sequence Generator and the transformations to stage the sequence of numbers.		

CURRVAL field

The CURRVAL field value is the NEXTVAL value plus the Increment By value. For example, if the Initial Value is 1 and Increment By is 1, the mapping task generates the following values for NEXTVAL and CURRVAL:

NEXTVAL	CURRVAL
1	2
2	3
3	4
4	5
5	6

Typically, you map the CURRVAL field when the NEXTVAL field is already mapped to a downstream transformation in the map. If you map the CURRVAL field without mapping the NEXTVAL field, the mapping task generates the same number for each row.

Sequence Generator properties

Configure Sequence Generator properties to define how the Sequence Generator generates numeric values.

Configure the following Sequence Generator properties on the **Sequence** tab:

Property	Description
Use Shared Sequence	Enable to generate sequence values using a shared sequence. When enabled, the sequence starts with the Current Value of the shared sequence. For information about shared sequences, see <i>Components</i> . Default is disabled.
Increment By	The difference between two consecutive values in a generated sequence. For example, if Increment By is 2 and the existing value is 4, then the next value generated in the sequence will be 6. Default is 1. Maximum value is 2,147,483,647.
End Value	Maximum value that the mapping task generates. If the sequence reaches this value during the task run and the sequence is not configured to cycle, the run fails. Maximum value is 9,223,372,036,854,775,807. If you connect the NEXTVAL field to a downstream integer field, set the End Value to a value no larger than the integer maximum value. If the NEXTVAL exceeds the data type maximum value for the downstream field, the mapping run fails.
Initial Value	The value you want the mapping task to use as the first value in the sequence. If you want to cycle through a series of values, the value must be greater than or equal to the Start Value and less than the End Value. Default is 1.
Cycle	If enabled, the mapping task cycles through the sequence range. If disabled, the task stops the sequence at the configured End Value. The session fails if the task reaches the End Value and still has rows to process. Default is disabled.
Cycle Start Value	Start value of the generated sequence that you want the mapping task to use if you use the Cycle option. When the sequence values reach the End Value, they cycle back to this value. Default is 0. Maximum value is 9,223,372,036,854,775,806.
Number of Cached Values	Number of sequential values the mapping task caches for each run. Each subsequent run uses a new batch of values. The task discards unused sequences for the batch. The mapping task updates the repository as it caches each value. When set to 0, the task does not cache values. Default is 0. This option is not available when the Cycle property is enabled.
Reset	If enabled, the mapping task generates values based on the original Initial Value for each run. Default is disabled.

Configure advanced Sequence Generator properties on the **Advanced** tab:

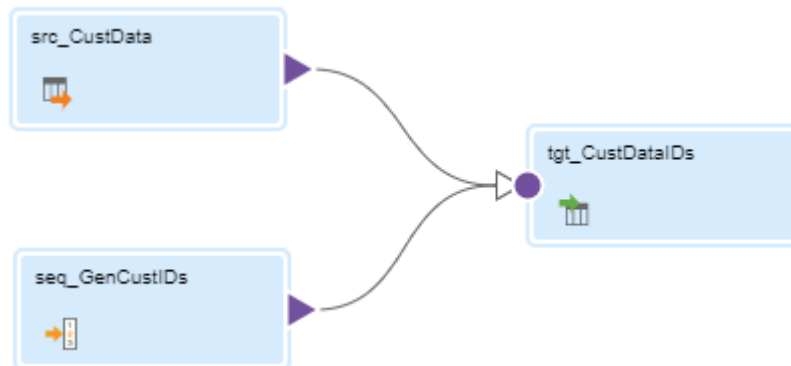
Property	Description
Tracing Level	Detail level of error and status messages that Data Integration writes in the session log. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Optional	<p>Determines whether the transformation is optional. If a transformation is optional and there are no incoming fields, the mapping task can run and the data can go through another branch in the data flow. If a transformation is required and there are no incoming fields, the task fails.</p> <p>For example, you configure a parameter for the source connection. In one branch of the data flow, you add a transformation with a field rule so that only Date/Time data enters the transformation, and you specify that the transformation is optional. When you configure the mapping task, you select a source that does not have Date/Time data. The mapping task ignores the branch with the optional transformation, and the data flow continues through another branch of the mapping.</p>
Disable incoming fields	Disable incoming fields to connect only the generated sequence to a downstream transformation. If you disable incoming fields, you must connect at least one field from another transformation to the downstream transformation.

Disabling incoming fields

You can disable incoming fields to connect only the generated sequence in the output fields, NEXTVAL and CURRVAL, to one or more downstream transformations. When you disable incoming fields, you cannot connect the Sequence Generator transformation to upstream transformations.

When you disable incoming fields, at least one field from another transformation must be connected to the downstream transformation along with the Sequence Generator fields. For example, if the mapping contains a Sequence Generator transformation and a Source transformation and the Sequence Generator transformation is connected to a Target transformation, you must connect at least one field from the Source transformation to the Target transformation.

The following image shows a mapping where incoming fields are disabled in the Sequence Generator transformation:



Sequence Generator transformation rules and guidelines

Consider the following guidelines when you create a Sequence Generator transformation:

- When you map the NEXTVAL or CURRVAL output fields, ensure that the data type of the mapped field is appropriate.
- When you run the mapping in the Mapping Designer, the current value is not saved so each time you run the mapping, it begins with the initial value.
- When you run the task in the mapping task wizard, you can edit the current value to start the sequence with a specified value.
- You cannot use a Sequence Generator transformation in a maplet.

CHAPTER 16

Sorter transformation

Use a Sorter transformation to sort data in ascending or descending order, according to a specified sort condition. You can configure the Sorter transformation for case-sensitive sorting and for distinct output. The Sorter transformation is a passive transformation.

You can use the Sorter transformation to increase performance with other transformations. For example, you can sort data that passes through a Lookup or an Aggregator transformation configured to use sorted incoming fields.

When you create a Sorter transformation, specify fields as sort conditions and configure each sort field to sort in ascending or descending order. You can use a parameter for the sort condition and define the value of the parameter when you configure the mapping task.

Sort conditions

Configure the sort condition to specify the sort fields and the sort order. The mapping task uses the sort condition to sort the data.

The sort fields are one or more fields that you want to use as the sort criteria. Configure the sort order to sort data in ascending or descending order.

When you specify multiple sort conditions, the mapping task sorts each condition sequentially. The mapping task treats each successive sort condition as a secondary sort of the previous sort condition. You can configure the order of sort conditions.

If you use a parameter for the sort condition, define the sort fields and the sort order when you run the mapping or when you configure the mapping task.

Sorter caches

The mapping task passes all incoming data into the Sorter transformation before it performs the sort operation. The mapping task uses cache memory to process Sorter transformations. If the mapping task cannot allocate enough memory, the mapping fails.

By default, the mapping task determines the cache size at run time. Before starting the sort operation, the mapping task allocates the amount of memory configured for the Sorter cache size.

Configure the Sorter cache size with a value less than the amount of available physical RAM on the machine that hosts the Secure Agent. Allocate at least 16 MB (16,777,216 bytes) of physical memory to sort data with

a Sorter transformation. When you allocate memory to the Sorter cache, consider other transformations in the mapping and the volume of data in the mapping task.

If the amount of incoming data is greater than the Sorter cache size, the mapping task temporarily stores data in the work directory. When storing data in the Sorter transformation work directory, the mapping task requires disk space of at least twice the amount of incoming data.

When you configure the tracing level to Normal, the mapping task writes the memory amount that the Sorter transformation uses to the session log.

Advanced properties

You can specify additional sort criteria in the Sorter transformation advanced properties. The mapping task applies the properties to all sort fields. The Sorter transformation properties also determine the system resources that the mapping task allocates when it sorts data.

You can configure the following advanced properties for a Sorter transformation:

Property	Description
Tracing Level	Detail level of error and status messages that Data Integration writes in the session log. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Sorter Cache Size	The maximum amount of memory required to perform the sort operation. The mapping task passes all incoming data into the Sorter transformation before it performs the sort operation. If the mapping task cannot allocate enough memory, the mapping fails. You can configure a numeric value for the sorter cache. Allocate at least 16 MB of physical memory. Default is Auto.
Case Sensitive	Determines whether the mapping task considers case when sorting data. When you enable a case-sensitive sort, the mapping task sorts uppercase characters higher than lowercase characters. Default is Case Sensitive.
Work Directory	The mapping task uses the work directory to create temporary files while it sorts data. After the mapping task sorts data, it deletes the temporary files. You can specify any directory on the Secure Agent machine to use as a work directory. Allocate at least 16 MB (16,777,216 bytes) of physical memory for the work directory. You can configure a system parameter or a user-defined parameter in this field. Default is the TempDir system parameter.
Distinct	Treats output rows as distinct. If you configure the Sorter transformation for distinct output rows, the mapping task configures all fields as part of the sort condition. The mapping task discards duplicate rows compared during the sort operation.
Null Treated Low	Treats a null value as lower than any other value. For example, if you configure a descending sort condition, rows with a null value in the sort field appear after all other rows.

Property	Description
Transformation Scope	<p>The transaction is determined by the commit or rollback point. The transformation scope specifies how the mapping task applies the transformation logic to incoming data:</p> <ul style="list-style-type: none"> - Transaction. Applies the transformation logic to all rows in a transaction. Choose Transaction when the results of the transformation depend on all rows in the same transaction, but not on rows in other transactions. - All Input. Applies the transformation logic to all incoming data. When you choose All Input, the mapping task drops incoming transaction boundaries. Choose All Input when the results of the transformation depend on all rows of data in the source.
Optional	<p>Determines whether the transformation is optional. If a transformation is optional and there are no incoming fields, the mapping task can run and the data can go through another branch in the data flow. If a transformation is required and there are no incoming fields, the task fails.</p> <p>For example, you configure a parameter for the source connection. In one branch of the data flow, you add a transformation with a field rule so that only Date/Time data enters the transformation, and you specify that the transformation is optional. When you configure the mapping task, you select a source that does not have Date/Time data. The mapping task ignores the branch with the optional transformation, and the data flow continues through another branch of the mapping.</p>

Sorter transformation example

You need to create an invoice for customer sales from a customer database. Use a Sorter transformation on the customer sales object to sort the data in ascending order according to the order number. Use the result of the Sorter transformation as an input to the Aggregator transformation. You can increase Aggregator transformation performance with the sorted incoming fields option.

The Product_Orders table contains information about all the orders placed by customers.

OrderID	ItemID	Item	Quantity	Price
43	123456	ItemA	3	3.04
41	456789	ItemB	2	12.02
43	000246	ItemC	6	34.55
45	000468	ItemD	5	0.56
43	NULL	ItemE	1	0.75
41	123456	ItemA	4	3.04
45	123456	ItemA	5	3.04
45	456789	ItemB	3	12.02

In the Mapping Designer, add Product_Orders as a source object.

Add a Sorter transformation to the mapping canvas, and connect it to the data flow. Sort the product orders by order ID and item ID.

The following image shows a sort condition with the order ID and item ID fields configured to sort in descending order:

srt_OrderID Properties

General | Incoming Fields | **Sort** | Advanced

Sort: Not Parameterized ▼

Sort Conditions (+)

Field	Sort Order	Actions
OrderID	Descending	▼
ItemID	Descending	▼

Enable a null treated low sort so that the mapping task considers null values to be lower than other values.

The following image shows the advanced properties for the Sorter transformation, with the Null Treated Low option selected:

srt_OrderID Properties

General | Incoming Fields | Sort | **Advanced**

▼ **Advanced properties**

Tracing Level: Normal ▼

Sorter Cache Size: Auto ☒ Auto ☐ Value

Case Sensitive: ☒

Work Directory: \$PMTempDir

Distinct: ☐

Null Treated Low: ☒

Transformation Scope: All Input ▼

Optional: ☒

After the mapping task sorts the data, it passes the following rows out of the Sorter transformation:

OrderID	ItemID	Item	Quantity	Price
45	456789	ItemB	3	12.02
45	123456	ItemA	5	3.04
45	000468	ItemD	5	0.56
43	123456	ItemA	3	3.04
43	000246	ItemC	6	34.55
43	NULL	ItemE	1	0.75
41	456789	ItemB	2	12.02

OrderID	ItemID	Item	Quantity	Price
41	123456	ItemA	4	3.04

You need to find out the total amount and the item quantity for each order. You can use the result of the Sorter transformation as an input to an Aggregator transformation to increase performance. Add the Aggregator transformation to the mapping, and connect the transformation to the data flow. Group the fields in the Aggregator transformation by the Order ID, and add an expression to sum the orders by price.

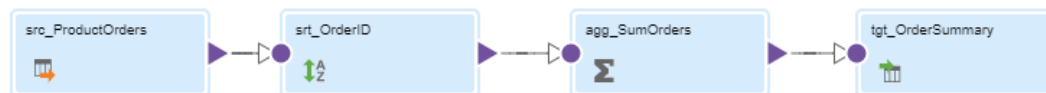
When you pass the data from the Sorter transformation, the Aggregator transformation groups the order ID to calculate the total amount for each order.

OrderID	Sum
45	54.06
43	217.17
41	36.2

Write the order summaries to an object in the data warehouse that stores all order totals.

The mapping task reads the orders data from the source, sorts the orders, totals the prices, and then writes the data to the target.

The following image shows the mapping of the data flow:



CHAPTER 17

SQL transformation

Use the SQL transformation to call a stored procedure or function in a relational database or to process SQL queries midstream in a pipeline. The transformation can call a stored procedure or function or process a query that you create in the transformation SQL editor.

The SQL transformation can process the following types of SQL statements:

Stored procedure or stored function

A stored procedure is a precompiled collection of database procedural statements and optional flow control statements, similar to an executable script. Stored procedures reside in the database and run within the database. A stored function is similar to a stored procedure, except that a function returns a single value.

When the SQL transformation processes a stored procedure or function, it passes input parameters to the stored procedure or function. The stored procedure or function passes the return value or values to the output fields of the transformation.

User-entered query

You can enter a query in the SQL editor. The SQL transformation processes the query and returns rows and database errors.

You can pass strings or parameters to the query to define dynamic queries or change the selection parameters. You can output multiple rows when the query has a SELECT statement.

Stored procedure or function processing

You can use the SQL transformation to call a stored procedure or function in a Microsoft SQL Server, MySQL, ODBC, or Oracle database. The stored procedure or stored function must exist in the database before you create the SQL transformation.

You can call a stored procedure or function with the following types of SQL transformations:

Connected SQL transformation

The transformation is connected to the mapping pipeline. The stored procedure or function runs on a row by row basis and can return a single output parameter or multiple output parameters.

You can map the incoming fields of the SQL transformation to the input fields of a stored procedure. The output fields in the SQL transformation consist of the stored procedure output parameters or return values.

A return value is a code or text string that you define in the stored procedure. For example, a stored procedure can return a value that indicates the date the stored procedure was run. When a stored procedure has a return value, the SQL transformation has a return value field.

Unconnected SQL transformation

The SQL transformation is not connected to the mapping pipeline. An Expression transformation calls the SQL transformation with a stored procedure expression, or the stored procedure runs before or after the mapping.

You can configure the expression to return the stored procedure output to expression output fields and variables. You can call the stored procedure from multiple expressions and nest stored procedures.

You cannot process a stored function with an unconnected SQL transformation.

You might use a stored procedure to perform the following tasks:

- Check the status of a target database before loading data into it.
- Determine if enough space exists in a database.
- Perform a specialized calculation.
- Retrieve data by a value.
- Drop and re-create indexes.
- Remove temporary tables.
- Verify that a table exists in a database

You can use a stored procedure to perform a calculation that you would otherwise make part of a mapping. For example, if you have a stored procedure to calculate sales tax, perform that calculation in an SQL transformation instead of re-creating the calculation in an Expression transformation.

When you run a mapping, the SQL transformation passes input parameters to the stored procedure. The stored procedure passes the return value or values to the output fields of the transformation.

Connected SQL transformation example

Your mapping includes user IDs in the data flow. You want to include user names in addition to user IDs.

You have a stored procedure that matches user IDs with user names in the database. You add an SQL transformation to your mapping, select the stored procedure, and map the `userId` incoming field with the `userId` input field in the stored procedure. You check the **Output Fields** tab for the SQL transformation to confirm that it includes the `username` field. When you run the mapping, the `username` value is returned with the user ID.

Unconnected SQL transformation example

Your mapping includes employee salary data and you want to update each employee's salary with a raise.

You have a stored procedure that calculates employee salary increases. The stored procedure returns the new salary and the percentage of increase. You add an unconnected SQL transformation and select the stored procedure.

You then add an Expression transformation to the mapping pipeline. In the Expression transformation, you add a variable field to capture the new salary. You add an output field and use the stored procedure function to configure the expression. You configure the arguments so that the output field returns the increase percentage and you create a second output field to return the new salary. You then map the new output fields to the downstream transformation.

Connected or unconnected SQL transformation for stored procedure processing

When you call a stored procedure in an SQL transformation, you can use a connected or an unconnected SQL transformation.

You process a stored procedure with a connected SQL transformation when you need data from an input field sent as an input parameter to the stored procedure, or you need the results of a stored procedure sent as an output parameter to another transformation.

You process a stored procedure with an unconnected SQL transformation when you need the stored procedure to run before or after a mapping, run nested stored procedures, or call the stored procedure multiple times.

The following table describes when you would use a connected or unconnected SQL transformation to process a stored procedure:

Scenario	SQL transformation type
Run a stored procedure before or after a mapping.	Unconnected
Run a stored procedure once during a mapping.	Unconnected
Run a stored procedure every time a row passes through the SQL transformation.	Connected or unconnected
Run a stored procedure based on data that passes through the mapping such as when a specific field does not contain a null value.	Unconnected
Pass parameters to the stored procedure and receive a single output parameter.	Connected or unconnected
Pass parameters to the stored procedure and receive multiple output parameters. Note: To get multiple output parameters from an unconnected SQL transformation, you must create variables for each output parameter.	Connected or unconnected
Run nested stored procedures.	Unconnected
Call a stored procedure multiple times within a mapping.	Unconnected

Unconnected SQL transformations

An unconnected SQL transformation is an SQL transformation that is not connected to the mapping pipeline. Use an unconnected SQL transformation to call a stored procedure.

You use an Expression transformation to call the unconnected SQL transformation with an :SP expression. Or, you configure the SQL transformation to invoke a stored procedure before or after a mapping run. For example, you might use an unconnected SQL transformation to remove temporary source tables after the mapping receives data from the source.

You might also use an unconnected SQL transformation when you want to call a stored procedure multiple times in a mapping.

Calling an unconnected SQL transformation from an expression

Call an unconnected SQL transformation from an Expression transformation with an :SP expression.

When you call a stored procedure from an expression, you configure the expression to return the stored procedure output values to fields in the expression. Use one of the following methods to return the output values:

- Assign the output value to a local variable field.
- Assign the output value to the system variable PROC_RESULT.

When you use the PROC_RESULT variable, Data Integration assigns the value of the return parameter directly to the output field, which you can write to a target. You can also assign one output parameter to PROC_RESULT and the other parameter to a variable.

Use expression variables to access OUT or INOUT parameters in the stored procedure. If the stored procedure returns multiple output parameters, you must create variables for each output parameter.

Use the following syntax to call a stored procedure in an expression:

```
:SP.<SQL transformation name> (arg1, arg2, PROC_RESULT)
```

If the stored procedure returns a single output parameter or return value, use the reserved variable PROC_RESULT as the output variable.

For example, the following expression calls a stored procedure called GET_NAME_FROM_ID:

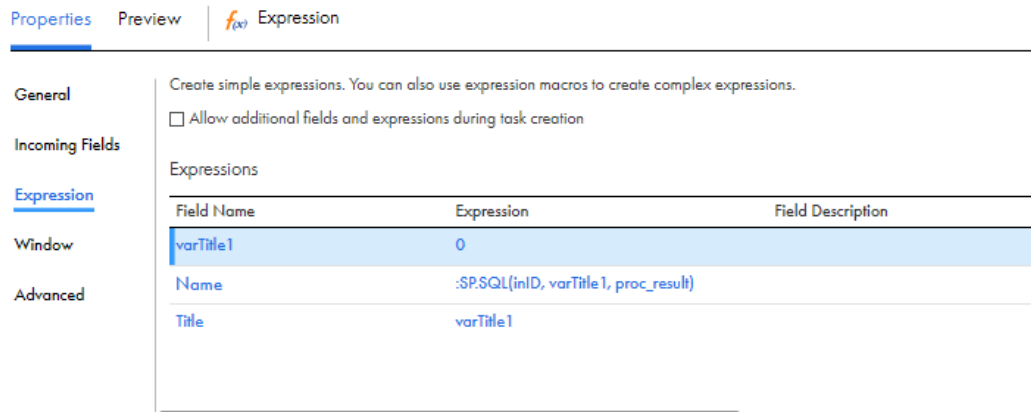
```
:SP.GET_NAME_FROM_ID(inID, PROC_RESULT)
```

inID can be either an input field in the stored procedure or a variable in the Expression transformation. When you run the mapping, Data Integration applies the value of PROC_RESULT to the output field for the expression.

If the stored procedure returns multiple output parameters, you must create expression variables for each output parameter. For example, if the stored procedure also returns a title, create a variable field called varTitle1 in the Expression transformation and use the field as the expression for an output field called Title. You write the following expression:

```
:SP.GET_NAME_FROM_ID(inID, varTitle1, PROC_RESULT)
```

The following image shows how you configure the Expression transformation:



Data Integration returns output parameters in the order they are declared in the stored procedure. In this example, Data Integration applies the value of the first output field in the stored procedure to varTitle1 and passes it to the Title field in the Expression transformation. It applies the value of the second stored procedure output field to the output field for the expression.

The data types for the expression fields and variables must match the data types for the stored procedure input/output variables and return value.

Invoking a stored procedure before or after a mapping run

You can configure an unconnected SQL transformation to process a stored procedure before or after a mapping run. Data Integration invokes the stored procedure at the specified time. You do not need to call the stored procedure with an :SP expression. You can configure the stored procedure to run before or after the mapping receives data from the source, or before or after the mapping loads data to the target.

You can configure the following stored procedure types:

- Source Pre-load. The stored procedure runs before the mapping retrieves data from the source.
- Source Post-load. The stored procedure runs after the mapping retrieves data from the source.
- Target Pre-load. The stored procedure runs before the mapping sends data to the target.
- Target Post-load. The stored procedure runs after the mapping sends data to the target.

On the **Advanced** tab, configure the stored procedure type and enter the call text for the stored procedure. The call text is the name of the stored procedure followed by any applicable input parameters in parentheses. If there are no input parameters, you must include an empty pair of parentheses. Do not include the SQL statement EXEC or use the :SP keyword.

For example, to call the stored procedure Drop_Table, enter the following call text:

```
Drop_Table()
```

To pass a string input parameter, enter it without quotes. If the string has spaces in it, enclose the parameter in double quotes. For example, if the stored procedure Drop_Table requires a table name as an input parameter, enter the following call text:

```
Drop_Table(Customer_list)
```

Query processing

You can configure the SQL transformation to process a user-entered query that runs against a Microsoft SQL Server or Oracle database. When you configure the SQL transformation to process a query, you create an active transformation. The transformation can return multiple rows for each input row.

When you enter a query, you can format the SQL and validate the syntax. Alternatively, you can create a string parameter to define the query in the mapping task.

You can create the following types of SQL queries:

Static SQL query

The query statement does not change, but you can use query parameters to change the data. Data Integration prepares the SQL query once and runs the query for all input rows.

Dynamic SQL query

You can change the query statements and the data. Data Integration prepares an SQL query for each input row.

You can optimize performance by creating static queries.

Static SQL queries

Create a static SQL query when you need to run the same query statements for each input row, but you want to change the data in the query for each input row. When you create a static SQL query, you use parameter binding in the SQL editor to define parameters for query data.

To change the data in the query, you configure query parameters and bind them to input fields in the transformation. When you bind a parameter to an input field, you identify the field by name in the query. Enclose the field name in question marks (?). The query data changes based on the value of the data in the input field.

For example, the following static queries use parameter binding:

```
DELETE FROM Employee WHERE Dept = ?Dept?
INSERT INTO Employee(Employee_ID, Dept) VALUES (?Employee_ID?, ?Dept?)
UPDATE Employee SET Dept = ?Dept? WHERE Employee_ID > 100
```

Example

The following static SQL query uses query parameters that bind to the Employee_ID and Dept input fields of an SQL transformation:

```
SELECT Name, Address FROM Employees WHERE Employee_Num = ?Employee_ID? and Dept = ?Dept?
```

The source has the following rows:

Employee_ID	Dept
100	Products
123	HR
130	Accounting

Data Integration generates the following query statements from the rows:

```
SELECT Name, Address FROM Employees WHERE Employee_ID = '100' and DEPT = 'Products'
SELECT Name, Address FROM Employees WHERE Employee_ID = '123' and DEPT = 'HR'
SELECT Name, Address FROM Employees WHERE Employee_ID = '130' and DEPT = 'Accounting'
```

Selecting multiple database rows

When the SQL query contains a SELECT statement, the transformation returns one row for each database row it retrieves. You must configure an output field for each column in the SELECT statement. The output fields must be in the same order as the columns in the SELECT statement.

When you configure output fields for database columns, you must configure the data type of each database column that you select. Select a native data type from the list. When you select the native data type, Data Integration configures the transformation data type for you.

The native data type in the transformation must match the database column data type. Data Integration matches the column data type in the database with the native database type in the transformation at run time. If the data types do not match, Data Integration generates a row error.

Dynamic SQL queries

A dynamic SQL query can execute different query statements for each input row. When you create a dynamic SQL query, you use string substitution to define string variables in the query and link them to input fields in the transformation.

To change a query statement, configure a string variable in the query for the portion of the query that you want to change. To configure the string variable, identify an input field by name in the query and enclose the name in tilde characters (~). The query changes based on the value of the data in the field.

The transformation input field that contains the query variable must be a string data type. You can use string substitution to change the query statement and the query data.

When you create a dynamic SQL query, Data Integration prepares a query for each input row. You can pass the following types of dynamic queries in an input field:

Full query

You can substitute the entire SQL query with query statements from source data.

Partial query

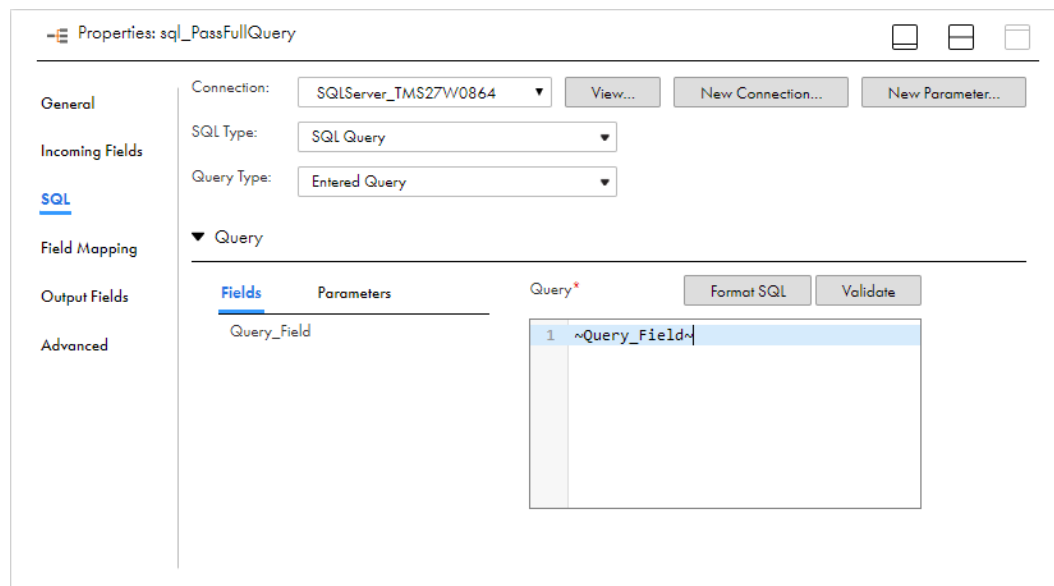
You can substitute a portion of the query statement, such as the table name.

Passing the full query

You can pass the full SQL query through an input field in the transformation. To pass the full query, create a query in the SQL editor that consists of one string variable to represent the full query: ~Query_Field~

To pass the full query, configure the source to pass the full query in an output field. Then, configure the SQL transformation to receive the query in the Query_Field input field.

The following image shows the transformation configuration:



Data Integration replaces the ~Query_Field~ variable in the dynamic query with the SQL statements from the source. It prepares the query and sends it to the database to process. The database executes the query. The SQL transformation returns database errors to the SQL_Error output field.

When you pass the full query, you can pass more than one query statement for each input row. For example, the source might contain the following rows:

```
DELETE FROM Person WHERE LastName = 'Jones'; INSERT INTO Person (LastName, Address)
VALUES ('Smith', '38 Summit Drive')
DELETE FROM Person WHERE LastName = 'Jones'; INSERT INTO Person (LastName, Address)
VALUES ('Smith', '38 Summit Drive')
DELETE FROM Person WHERE LastName = 'Russell';
```

You can pass any type of query in the source data. When you configure SELECT statements in the query, you must configure output fields for the database columns that you retrieve from the database. When you mix SELECT statements and other types of queries, the output fields that represent database columns contain null values when no database columns are retrieved.

Substituting the table name in a string

You can use a partial query to substitute the table name. To substitute the table name, configure an input field to receive the table name from each input row. Identify the input field by name in the query and enclose the name with tilde characters (~).

For example, the following dynamic query contains a string variable, ~Table_Field~:

```
SELECT Emp_ID, Address from ~Table_Field~ where Dept = 'HR'
```

The source might pass the following values to the Table_Field column:

Table_Field

Employees_USA

Employees_England

Employees_Australia

Data Integration replaces the ~Table_Field~ variable with the table name in the input field:

```
SELECT Emp_ID, Address from Employees_USA where Dept = 'HR'
SELECT Emp_ID, Address from Employees_England where Dept = 'HR'
SELECT Emp_ID, Address from Employees_Australia where Dept = 'HR'
```

Passive mode configuration

When you create an SQL transformation, you can configure it to run in passive mode instead of active mode. A passive transformation does not change the number of rows that pass through it. It maintains transaction boundaries and row types.

If you configure the SQL transformation to process a query, you can configure passive mode when you create the transformation. Configure passive mode in the transformation advanced properties.

When you configure the transformation as a passive transformation and a SELECT query returns more than one row, Data Integration returns the first row and an error to the SQLError field. The error states that the SQL transformation generated multiple rows.

If the SQL query has multiple SQL statements, Data Integration executes all statements but returns data for the first SQL statement only. The SQL transformation returns one row. The SQLError field contains the errors from all SQL statements. When multiple errors occur, they are separated by semicolons (;) in the SQLError field.

SQL statements that you can use in queries

You can use certain data definition, data manipulation, data language control, and transaction control statements with the SQL transformation.

The following table lists the statements that you can use in an SQL query in the SQL transformation:

Statement Type	Statement	Description
Data Definition	ALTER	Modifies the structure of the database.
Data Definition	COMMENT	Adds comments to the data dictionary.
Data Definition	CREATE	Creates a database, table, or index.
Data Definition	DROP	Deletes an index, table, or database.
Data Definition	RENAME	Renames a database object.
Data Definition	TRUNCATE	Removes all rows from a table.
Data Manipulation	CALL	Calls a PL/SQL or Java subprogram.
Data Manipulation	DELETE	Deletes rows from a table.
Data Manipulation	EXPLAIN PLAN	Writes the access plan for a statement into the database Explain tables.
Data Manipulation	INSERT	Inserts rows into a table.
Data Manipulation	LOCK TABLE	Prevents concurrent application processes from using or changing a table.
Data Manipulation	MERGE	Updates a table with source data.
Data Manipulation	SELECT	Retrieves data from the database.
Data Manipulation	UPDATE	Updates the values of rows of a table.
Data Control Language	GRANT	Grants privileges to a database user.
Data Control Language	REVOKE	Removes access privileges for a database user.
Transaction Control	COMMIT	Saves a unit of work and performs the database changes for that unit of work.
Transaction Control	ROLLBACK	Reverses changes to the database since the last COMMIT.

Rules and guidelines for query processing

Use the following rules and guidelines when you configure the SQL transformation to process a query:

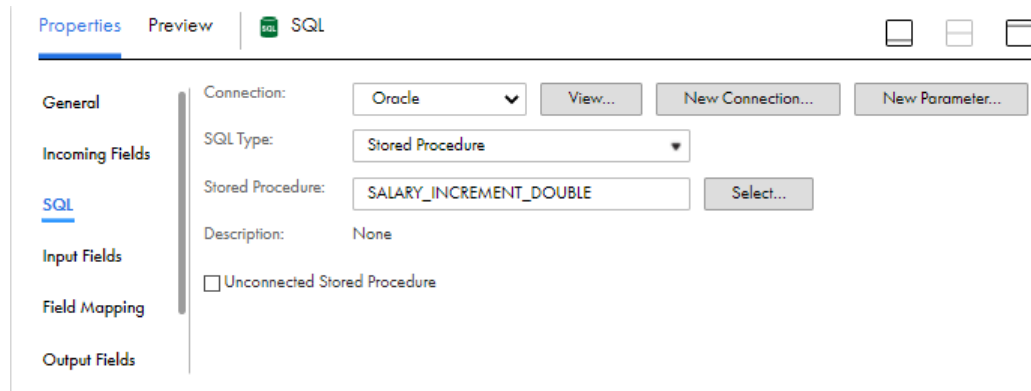
- The number and the order of the output fields must match the number and order of the fields in the query SELECT clause.
- The native data type of an output field in the transformation must match the data type of the corresponding column in the database. Data Integration generates a row error when the data types do not match.

- When the SQL query contains an INSERT, UPDATE, or DELETE clause, the transformation returns data to the SQLError field, the pass-through fields, and the NumRowsAffected field when it is enabled. If you add output fields, the fields receive NULL data values.
- When the SQL query contains a SELECT statement and the transformation has a pass-through field, the transformation returns data to the pass-through field whether or not the query returns database data. The SQL transformation returns a row with NULL data in the output fields.
- When the number of output fields is more than the number of columns in the SELECT clause, the extra fields receive a NULL value.
- When the number of output fields is less than the number of columns in the SELECT clause, Data Integration generates a row error.
- You can use string substitution instead of parameter binding in a query. However, the input fields must be string data types.

SQL transformation configuration

Use the **Properties** panel to configure the SQL transformation. When you configure the transformation, you specify the transformation name and description, configure fields and field mapping, and specify the type of SQL that the transformation processes. You also set the advanced properties for the transformation.

The following image shows the **Properties** panel of an SQL transformation that is configured to process a stored procedure:



Configure the transformation using the following tabs on the **Properties** panel:

General

Configure the SQL transformation name and optional description.

Incoming Fields

Define field rules that determine the data to include in the transformation.

Not applicable to unconnected SQL transformations.

SQL

Define the database connection and the type of SQL that the transformation processes: either a stored procedure, stored function, or query.

If you configure the transformation to process a stored procedure or stored function, you select the stored procedure or stored function on this tab.

If you configure the transformation to process a stored procedure, you can choose to run the transformation as an unconnected transformation.

If you configure the transformation to process a user-entered query, the SQL editor appears on this tab. Enter the query in the SQL editor.

Input Fields

For transformations that process stored procedures, displays the stored procedure input fields.

Field Mapping

For stored procedure and stored functions, specify how to map incoming fields to the input fields of the selected stored procedure or function.

You do not configure the field mapping for queries or unconnected SQL transformations.

Output Fields

For stored procedures and stored functions, displays a preview of the SQL transformation output fields. For user-entered queries, configure output fields for the columns retrieved from the database.

For queries, the output fields also include the `SQL_Error` field, the optional `NumRowsAffected` field, and optional pass-through fields.

Advanced

Define advanced properties for the transformation. Advanced properties differ based on the type of SQL that the transformation processes.

Note: Field name conflicts must be resolved in an upstream transformation. You cannot use field name conflict resolution rules in an SQL transformation.

Configuring the SQL type

You can configure the SQL transformation to process a stored procedure, stored function, or user-entered query on the **SQL** tab of the SQL transformation.

The steps for configuring the transformation vary based on the type of SQL that the transformation processes.

Selecting a stored procedure or function

You can configure the SQL transformation to process a stored procedure or stored function on the **SQL** tab of the SQL transformation.

1. In the **Properties** panel of the SQL transformation, click the **SQL** tab.
2. Select the connection to the database.

You can select the connection or use a parameter.

Note: If you want to parameterize the connection, create the parameter after you select the stored procedure or function.

3. Set the SQL type to **Stored Procedure** or **Stored Function**.
4. Click **Select** to select the stored procedure or function from the database, or enter the exact name of the stored procedure or function to call.

The stored procedure or function name is case-sensitive.

Note: If you add a new stored procedure to the database while you have the mapping open, the new stored procedure does not appear in the list of available stored procedures. To refresh the list, close and reopen the mapping.

5. If the transformation processes a stored procedure and you want to run the transformation in unconnected mode, select **Unconnected Stored Procedure**.
6. If you want to parameterize the connection, click **New Parameter** and enter the details for the connection parameter.

Entering a query

You can configure the SQL transformation to process a user-entered query on the **SQL** tab of the SQL transformation. Optionally, you can parameterize the query. When you parameterize the query, you enter the full query in the mapping task.

1. In the **Properties** panel of the SQL transformation, click the **SQL** tab.
2. Select the connection to the database or use a parameter.
3. Set the SQL type to **SQL Query**.
4. Set the query type to **Entered Query**.
5. If you do not want to parameterize the query, enter the query in the query editor.

Incoming fields are listed on the **Fields** tab. To add a field to the query, select the field and click **Add**.

You can format the SQL and validate the syntax.

Note: The syntax validation performs a general SQL syntax check but does not verify the SQL against the database. The validation can return a syntax error even though the SQL is valid for the database. In this case, you can still save and run the mapping.

If you update the incoming fields after you configure the query, open the **SQL** tab to refresh the changes.

6. If you want to parameterize the query, perform the following steps:
 - a. Open the **Parameters** tab and create a new string parameter.
 - b. Select the parameter, and then click **Add** to add the parameter to the query editor.

When you add the parameter to the query editor, Data Integration encloses it in dollar sign characters (\$).

Do not format the SQL or validate the query.

SQL transformation field mapping

Configure field mapping in an SQL transformation to define how to use data from the upstream transformation in a stored procedure or function. You do not configure field mapping when the transformation processes a query.

Configure field mapping on the **Field Mapping** tab of the **Properties** panel.

You can configure the following field mapping options:

Field Map Options

Method of mapping fields to the SQL transformation. Select one of the following options:

- **Manual.** Manually link incoming fields to the store procedure or function's input fields. Removes links for automatically mapped fields.
- **Automatic.** Automatically link fields with the same name. Use when all of the fields that you want to link share the same name. You cannot manually link fields with this option.

- **Completely Parameterized.** Use a parameter to represent the field mapping. In the task, you can configure all field mappings.
- **Partially Parameterized.** Configure links in the mapping that you want to enforce and use a parameter to allow other fields to be mapped in the mapping task. Or, use a parameter to configure links in the mapping, and allow all fields and links to display in the task for configuration.
To see more information on field mapping parameters, see *Mappings*.

Show Fields

Controls the fields that appear in the **Incoming Fields** list. Show all fields, unmapped fields, or mapped fields.

Automap

Links fields with matching names. Allows you to link matching fields, and then manually configure other field mappings.

You can map fields in the following ways:

- **Exact Field Name.** Data Integration matches fields of the same name.
- **Smart Map.** Data Integration matches fields with similar names. For example, if you have an incoming field `Cust_Name` and a target field `Customer_Name`, Data Integration automatically links the `Cust_Name` field with the `Customer_Name` field.

You can use both Exact Field Name and Smart Map in the same field mapping. For example, use Exact Field Name to match fields with the same name and then use Smart Map to map fields with similar names.

You can undo all automapped field mappings by clicking **Automap > Undo Automap**. To unmap a single field, select the field to unmap and click **Actions > Unmap**.

Data Integration highlights newly mapped fields. For example, when you use Exact Field Name, Data Integration highlights the mapped fields. If you then use Smart Map, Data Integration only highlights the fields mapped using Smart Map.

Action menu

Additional field link options. Provides the following options:

- **Map selected.** Links the selected incoming field with the selected stored procedure or function input field.
- **Unmap selected.** Clears the link for the selected field.
- **Clear all.** Clears all field mappings.

Show

Determines how field names appear in the **Stored Procedure Input Fields** list. Use technical field names or labels.

SQL transformation output fields

You can view output fields for the SQL transformation on the **Output Fields** tab of the **Properties** panel. The Mapping Designer displays the name, type, precision, scale, and origin for each output field.

Information on the **Output Fields** tab varies based on the SQL type.

Output fields for stored procedures and functions

When the SQL transformation processes a stored procedure or function, the output fields include output parameters from the database. You cannot edit the transformation output fields. If you want to exclude

output fields from the data flow or rename output fields before you pass them to a downstream transformation, configure the field rules for the downstream transformation.

Output fields for queries

When the SQL transformation processes a query, the output fields include the following fields:

Retrieved column fields

When the SQL query contains a SELECT statement, the transformation returns one row for each database row that it retrieves.

For user-entered queries, you must configure an output field for each column in the SELECT statement. The output fields must be in the same order as the columns in the SELECT statement.

SQLException field

Data Integration returns row errors to the SQLException field when it encounters a connection or syntax error. It returns NULL to the SQLException field when no SQL errors occur.

For example, the following SQL query generates a row error from an Oracle database when the Employees table does not contain Product_ID:

```
SELECT Product_ID FROM Employees
```

Data Integration generates one row. The SQLException field contains the following error text in one line:

```
ORA-0094: "Product_ID": invalid identifier Database driver error... Function Name:  
Execute SQL Stmt: SELECT Product_ID from Employees Oracle Fatal Error
```

When a query contains multiple statements, and you configure the SQL transformation to continue on SQL error, the SQL transformation might return rows from the database for one query statement, but return database errors for another query statement. The SQL transformation returns any database error in a separate row.

NumRowsAffected field

You can enable the NumRowsAffected output field to return the number of rows affected by the INSERT, UPDATE, or DELETE query statements in each input row. Data Integration returns the NumRowsAffected for each statement in the query. NumRowsAffected is disabled by default.

When you enable NumRowsAffected and the SQL query does not contain an INSERT, UPDATE, or DELETE statement, NumRowsAffected is zero in each output row.

The following table lists the output rows that the SQL transformation generates when you enable NumRowsAffected:

Query Statement	Output Rows
UPDATE, INSERT, DELETE only	One row for each statement with the NumRowsAffected for the statement.
One or more SELECT statements	Total number of database rows retrieved. NumRowsAffected is zero in each row.
DDL queries such as CREATE, DROP, TRUNCATE	One row with zero NumRowsAffected.

When a query contains multiple statements, Data Integration returns the NumRowsAffected for each statement. NumRowsAffected contains the sum of the rows affected by each INSERT, UPDATE, and DELETE statement in an input row.

For example, a query contains the following statements:

```
DELETE from Employees WHERE Employee_ID = '101';
SELECT Employee_ID, LastName from Employees WHERE Employee_ID = '103';
INSERT into Employees (Employee_ID, LastName, Address)VALUES ('102', 'Gein', '38
Beach Rd')
```

The DELETE statement affects one row. The SELECT statement does not affect any row. The INSERT statement affects one row.

Data Integration returns one row from the DELETE statement. NumRowsAffected is equal to one. It returns one row from the SELECT statement, NumRowsAffected is zero. It returns one row from the INSERT statement with NumRowsAffected equal to one.

Pass-through fields

Define incoming fields as pass-through fields to pass data through the SQL transformation. The SQL transformation returns data from pass-through fields whether or not the SQL query returns rows.

When the source row contains a SELECT statement, the SQL transformation returns the data in the pass-through field in each row it returns from the database. If the query result contains multiple rows, the SQL transformation repeats the pass through field data in each row.

When a query returns no rows, the SQL transformation returns the pass-through column data and null values in the output fields. For example, queries that contain INSERT, UPDATE, and DELETE statements return no rows. If the query has errors, the SQL transformation returns the pass-through column data, the SQL error message, and null values in the output fields.

To define a pass-through field, click **Add** in the Pass-Through Fields area, and then select the field you want to pass through the SQL transformation. When you configure an incoming field as a pass-through field, Data Integration adds the field with the suffix "_output" in the Pass-Through Fields area.

If you configure a field as a pass-through field and then change the field name in the source, Data Integration does not update the pass-through field name and no data is passed through the field. In the SQL transformation, delete the old pass-through field and configure the updated incoming field as a pass-through field.

Advanced properties

Configure advanced properties for the SQL transformation on the **Advanced** tab. The advanced properties vary based on whether the transformation processes a stored procedure or function or a query.

Advanced properties for stored procedures or functions

The following table describes the advanced properties when the transformation processes a stored procedure or function:

Property	Description
Tracing Level	Detail level of error and status messages that Data Integration writes in the session log. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Subsecond Precision	Subsecond precision for datetime fields. You can change the precision for databases that have an editable scale for datetime data. If you enable pushdown optimization, the database returns the complete datetime value, regardless of the subsecond precision setting. Enter a positive integer value from 0 to 9. Default is 6 microseconds.

Property	Description
Stored Procedure Type	<p>For unconnected transformations, determines when the stored procedure runs.</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> - Target Pre Load. Runs before the target is loaded. - Target Post Load. Runs after the target is loaded. - Normal. Runs on a row-by-row basis. - Source Pre Load. Runs before the mapping receives data from the source. - Source Post Load. Runs after the mapping receives data from the source.
Call Text	<p>For unconnected transformations with stored procedure type Target Pre/Post Load or Source Pre/Post Load, enter the call text for the stored procedure.</p> <p>The call text is the stored procedure name followed by the input parameters in parentheses. If there are no input parameters, you must include an empty pair of parentheses. Do not include the SQL statement EXEC or use the :SP keyword.</p> <p>Does not apply to Normal stored procedure types.</p>

Advanced properties for queries

The following table describes the advanced properties when the transformation processes a saved or user-entered query:

Property	Description
Behavior	<p>Transformation behavior, either active or passive.</p> <p>If active, the transformation can generate more than one output row for each input row. If passive, the transformation generates one output row for each input row.</p> <p>Default is Active.</p>
Continue on SQL Error within Row	<p>Continues processing the remaining SQL statements in a query after an SQL error occurs.</p> <p>Enable this option to ignore SQL errors in a statement. Data Integration continues to run the rest of the SQL statements for the row. The SQL transformation does not generate a row error, but the SQLError field contains the failed SQL statement and error messages.</p> <p>Tip: Disable this option to debug database errors. Otherwise, you might not be able to associate errors with the query statements that caused them.</p> <p>Default is disabled.</p>
Auto Commit	<p>Enables auto-commit for each database connection.</p> <p>Each SQL statement in a query defines a transaction. A commit occurs when the SQL statement completes or the next statement is executed, whichever comes first.</p> <p>Default is disabled.</p>
Max Output Row Count	<p>The maximum number of rows that the SQL transformation can output from a SELECT query.</p> <p>To configure unlimited rows, set this property to zero. Default is 600.</p>

Property	Description
Tracing Level	Detail level of error and status messages that Data Integration writes in the session log. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Transformation Scope	<p>The method in which Data Integration applies the transformation logic to incoming data.</p> <p>Select one of the following options:</p> <ul style="list-style-type: none"> - Row. Applies the transformation logic to one row of data at a time. Choose Row when the results of the transformation depend on a single row of data. - Transaction. Applies the transformation logic to all rows in a transaction. Choose Transaction when a row of data depends on all rows in the same transaction, but does not depend on rows in other transactions. - All Input. Applies the transformation logic on all incoming data. When you choose All Input, Data Integration drops incoming transaction boundaries. Choose All Input when a row of data depends on all rows in the source. <p>Default is Row.</p>

CHAPTER 18

Union transformation

The Union transformation is an active transformation that you use to merge data from multiple pipelines into a single pipeline.

For data integration patterns, it is common to combine two or more data sources into a single stream that includes the union of all rows. The data sources often do not have the same structure, so you cannot freely join the data streams. The Union transformation enables you to make the metadata of the streams alike so that you can combine the data sources in a single target.

The Union transformation merges data from multiple sources similar to the UNION ALL SQL statement. For example, you might use the Union transformation to merge employee information from ADP with data from a Workday employee object.

You can add, change, or remove specific fields when you merge data sources with a Union transformation.

At run time, the mapping task processes input groups in parallel. It concurrently reads the sources connected to the Union transformation and pushes blocks of data into the input groups of the transformation. As the mapping runs, it merges data into a single output group based on the field mappings.

Comparison to Joiner transformation

A Union transformation can merge data from multiple sources but does not combine data based on a join condition or remove duplicate rows, like a Joiner transformation.

The following table identifies some key differences between the Union transformation and Joiner transformation, which also merges data from multiple sources. Factor these differences into your mapping design:

Requirement	Union transformation	Joiner transformation
Remove duplicate rows	No. You can use a Router or Filter transformation downstream from the Union transformation to remove duplicates.	Yes
Combine records based on a join condition	No. The Union Transformation is equivalent to a UNION ALL statement in SQL, which combines data vertically from multiple sources.	Yes. The Joiner transformation supports Normal, Right Outer, Left Outer, and Full Outer JOINS.
Include multiple input groups	Yes. You can define multiple input groups and one output group.	Yes. You can define two input groups, Master and Detail.

Requirement	Union transformation	Joiner transformation
Include heterogeneous sources	Yes	No
Merge different data types	All of the source columns must have similar data types. The number of columns in each source must be the same.	At least one column in the sources to be joined must have the same data type.
Generate transactions	No	Yes

Planning to use a Union transformation

When you use a Union transformation in a mapping, consider the following guidelines:

- Before you add a Union transformation to a mapping, add all Source transformations and include the other upstream transformations that you want to use.
- You can use a Sequence Generator transformation upstream from a Union transformation if you connect both the Sequence Generator and a Source transformation to one input group of the Union transformation.

Input groups

By default, a Union transformation has two input groups. If you want to merge data from more than two sources, add an input group for each additional source. Each group can have different field rules for each upstream transformation.

The input groups have the following characteristics:

- The Union transformation initializes its output fields based on fields in the first source that you connect to an input group.
- Each input group can use a different field mapping mode or parameter.
- You can parameterize the field mappings or define the field mapping for each input group.

To add an input group, in the Mapping Designer, connect an upstream transformation to the "New Group" group of the Union transformation. You can also add input groups on the **Incoming Fields** tab of the Union transformation.

You can rename input groups. You can also delete input groups as long as there are at least two remaining input groups. Rename and delete input groups on the **Incoming Fields** tab.

Output fields

When you connect upstream transformations to a Union transformation, you initialize the output fields. The initial output fields are an exact copy of the input fields in group Input1.

When you define output fields, note the following:

- After you initialize the output fields, you cannot change the output fields by connecting or disconnecting the input group.
- You can manually add output fields if you add them before you connect one of the Union transformation input groups.
- When you add an output field, you define the field name, data type, precision, scale, and optional description. The description can contain up to 4000 characters.
- If you connect the Union transformation to an upstream transformation that does not pass in any fields, the output fields are not initialized.
- At run time, the mapping passes null values to output fields that are not in a field mapping.

Field mappings

The Union transformation can merge data from multiple source pipelines. The sources can have the same set of fields, have some matching fields, or use parameterized field mappings.

When you work with field mappings in a Union transformation, note the following:

- You must use input groups where the fields have the identical name, type, precision, and scale.
- You can edit, remove, or manually add some of the output fields.
- As part of the field mapping, you choose an input group and specify the parameter from the input group.
- You can use parameters for fields in all input groups.
- You can parameterize the field mapping or map by field name for each input group. At run time, the task adds an exact copy of the fields from the input group as output fields.

If you want Data Integration to automatically link fields with the same name and you also want to manually map fields, select the **Manual** option and click **Automap**.

You can map fields in the following ways:

- **Exact Field Name.** Data Integration matches fields of the same name.
- **Smart Map.** Data Integration matches fields with similar names. For example, if you have an incoming field `Cust_Name` and a target field `Customer_Name`, Data Integration automatically links the `Cust_Name` field with the `Customer_Name` field.

You can use both Exact Field Name and Smart Map in the same field mapping. For example, use Exact Field Name to match fields with the same name and then use Smart Map to map fields with similar names.

You can undo all automapped field mappings by clicking **Automap > Undo Automap**. To unmap a single field, select the field to unmap and click **Actions > Unmap**.

Data Integration highlights newly mapped fields. For example, when you use Exact Field Name, Data Integration highlights the mapped fields. If you then use Smart Map, Data Integration only highlights the fields mapped using Smart Map.

Advanced properties

You can configure advanced properties for a Union transformation. The advanced properties control settings such as the tracing level for session log messages and whether the transformation is optional or required.

You can configure the following properties:

Property	Description
Tracing Level	Detail level of error and status messages that Data Integration writes in the session log. You can choose terse, normal, verbose initialization, or verbose data. Default is normal.
Optional	<p>Determines whether the transformation is optional. If a transformation is optional and there are no incoming fields, the mapping task can run and the data can go through another branch in the data flow. If a transformation is required and there are no incoming fields, the task fails.</p> <p>For example, you configure a parameter for the source connection. In one branch of the data flow, you add a transformation with a field rule so that only Date/Time data enters the transformation, and you specify that the transformation is optional. When you configure the mapping task, you select a source that does not have Date/Time data. The mapping task ignores the branch with the optional transformation, and the data flow continues through another branch of the mapping.</p>

Union Transformation example

You have demographic data about employees from two flat file sources and you want to merge that data.

You receive the following data in a .txt file:

```
employee_ID,first_name,last_name,location,email,phone
1211,John,Davis,Redwood City,jdavis@infa2.com,555-555-4444
0233,Miles,Simone,Barcelona,msimone@infa2.com,555-555-6666
1045,Billie,Coltrane,Philadelphia,bcoltrane@infa2.com,555-555-7777
0987,Django,Holiday,Paris,dholiday@infa3.com,444-444-4444
1199,Nina,Reinhardt,New York,nreinhardt@infa3.com,444-555-5555
```

A second file contains the following data:

```
ID,first,last,dept,e-mail,phone
0456,Joni,Smith,Marketing,j_smith@infa4.com,333-333-3333
1325,David,Mitchell,R&D,dmitchell@infa4.com,222-222-2222
1101,David,Harry,R&D,dharry@infa5.com,777-777-7777
0623,Debbie,Byrne,HR,dbyrne@infa5.com,888-888-8888
0777,Patti,Bowie,Sales,pbowie@infa5.com,999-999-9999
```

You want to merge those records into a single dataset in MySQL with the following columns:

- id
- last
- first
- email
- phone

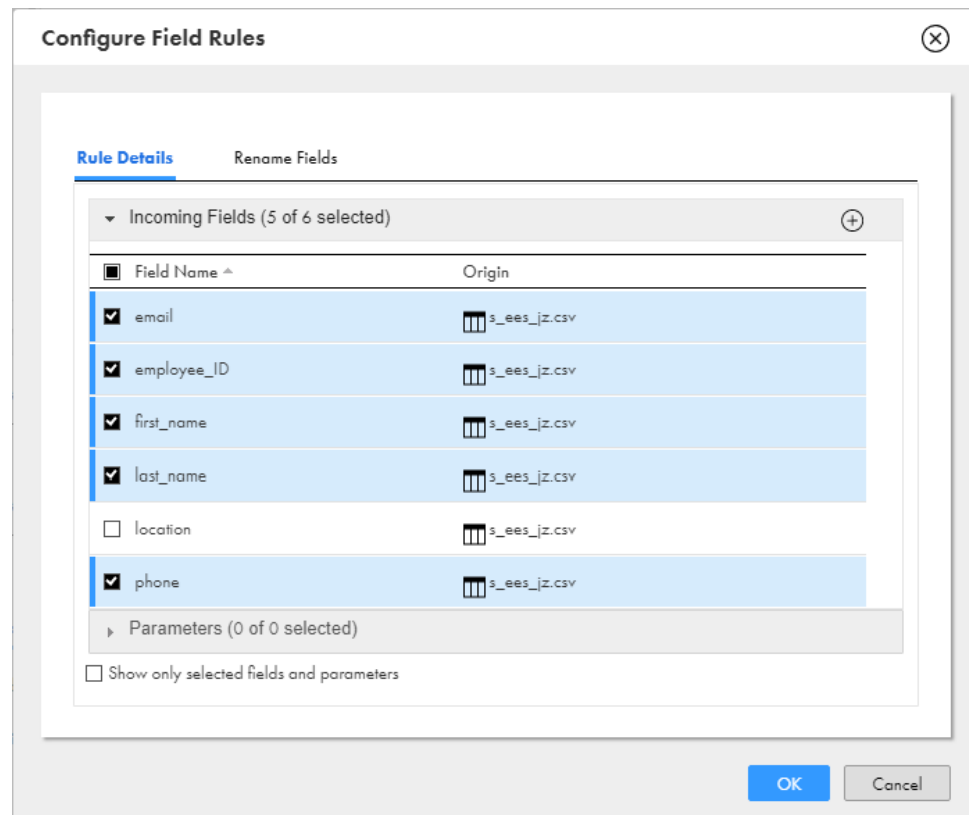
Note: Remember that the data to be merged with a Union transformation must have the same data type, precision, and scale.

To merge the files with a Union transformation, complete the following steps:

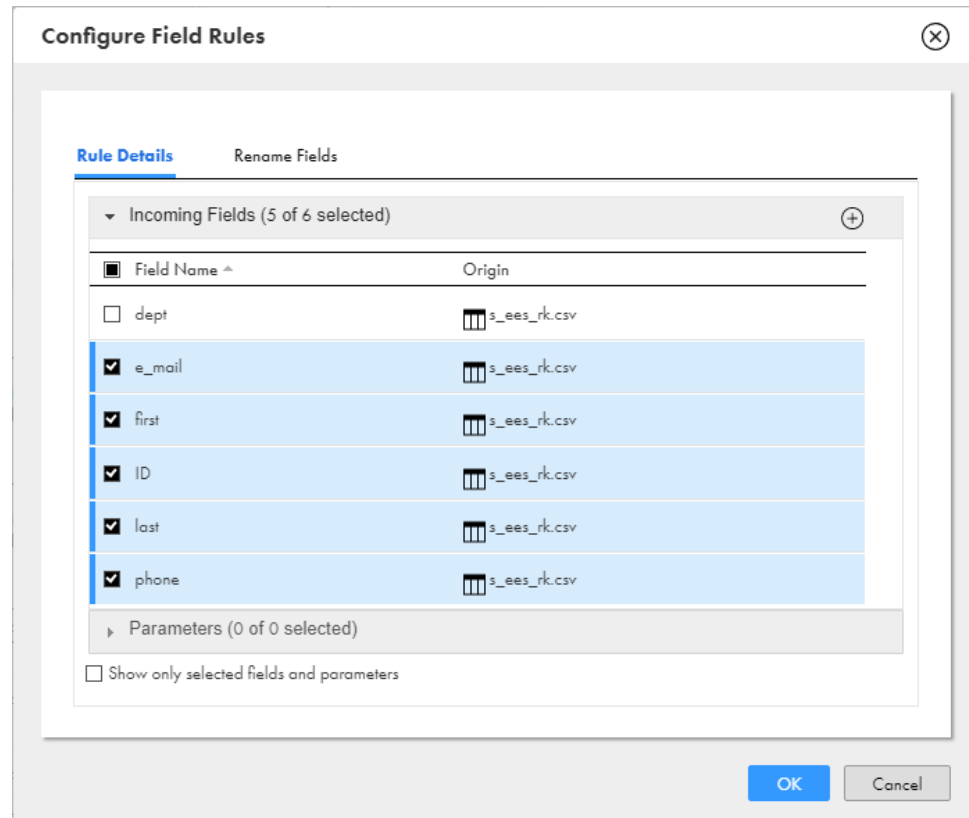
1. Ensure that the source files reside in a location accessible to your Secure Agent.

2. Define a connection to access the .csv files.
3. Create a mapping in the Mapping Designer.
4. Add two Source transformations to the mapping to connect to data in the .csv files.
5. Add a Union transformation and connect the Source transformations to it.
6. In the Union transformation Properties, perform the following steps for each input group:
 - a. In the Field Rules section, click the group you want to configure.
 - b. (Optional) For the incoming fields, select the fields you want to merge in the output.

The following image shows the selected fields in the first input group:

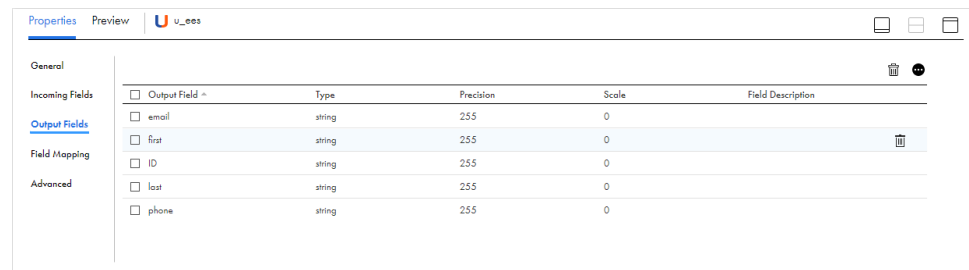


The following image shows the selected fields in the second input group:



If you do not specify a rule to exclude fields, at run time, the task ignores any fields that you do not map to the output fields.

- c. Edit the Output field names in the Union transformation, to correspond to the field names that you want in the target:



Note: You can also select fields, change metadata, add other fields, or convert the field types, for example, from integer to number.

- In the Field Mapping of the Union transformation, ensure that the fields are correctly mapped for each input group:

Properties Preview **u_ees**

General Input group: Input1

Incoming Fields Field map options: Manual Options

Output Fields Automatch

Field Name ^	Mapped Field
employee_ID	ID
first_name	first_name
last_name	last_name
email	email
phone	phone

- Add a Target transformation to the mapping.
- Connect the Union transformation to the Target transformation.
- In the Target transformation field mapping, select automatic field mapping:

t_ees Properties

General Field map options: Automatic Note: This option will automatically map any fields added later by name. Options

Incoming Fields Incoming Fields: (0 of 0 mapped)

Field Name ^
ID
last
first
email
phone

Target Fields Target Fields: (0 of 0 mapped)

Field Name ^	Mapped Field
There are no target fields to show as the object has not been specified or you have chosen to create a new target object.	

When complete, the mapping appears similar to the following image:



INDEX

A

- active transformations
 - Rank [112](#)
 - Router [120](#)
 - Sorter [130](#)
- aggregate fields
 - Aggregator transformations [58](#)
- aggregate functions
 - in Aggregator transformations [58](#)
 - nested [58](#)
- Aggregator transformation
 - advanced properties [59](#)
- Aggregator transformations
 - aggregate fields [58](#)
 - aggregate functions [58](#)
 - conditional clause example [59](#)
 - example in mapping [107](#)
 - group by fields [56](#)
 - overview [56](#)
 - using sorted data [57](#)
- AND
 - reserved word [64](#)

C

- caches
 - dynamic lookup cache [86](#)
 - Rank transformation [113](#), [117](#)
 - Sorter transformation [130](#)
- CHR function
 - inserting single quotation mark [63](#)
- Cloud Application Integration community
 - URL [8](#)
- Cloud Developer community
 - URL [8](#)
- comments
 - adding to field expressions [63](#)
- conditions
 - Router transformation [121](#)
- connected transformations
 - Rank [112](#)
 - Router [120](#)
- constants
 - definition [62](#)
- CURRVAL [126](#)

D

- data cache [19](#)
- Data Integration community
 - URL [8](#)
- data preview
 - sources, targets, and lookup objects [17](#)

- databases
 - configuration in mapping sources [38](#)
 - configuration in mapping targets [48](#)
 - target update override [50](#)
- DD_DELETE constant
 - reserved word [64](#)
- DD_INSERT constant
 - reserved word [64](#)
- DD_REJECT constant
 - reserved word [64](#)
- DD_UPDATE constant
 - reserved word [64](#)
- dynamic lookup cache
 - SQL overrides [90](#)

E

- examples
 - Joiner transformation [72](#)
 - lookup SQL override [83](#)
 - Rank transformation [118](#)
 - Router transformation [123](#)
- expression editor
 - expression transformation [62](#)
- expression macros in mappings
 - configuring a horizontal macro [27](#)
 - configuring a vertical macro [22](#)
 - horizontal expansion functions [26](#)
 - horizontal macro configuration [26](#)
 - hybrid macro configuration [30](#)
 - macro input field [21](#)
 - macro input field configuration for incoming fields in a horizontal macro [27](#)
 - macro input field configuration for vertical macros [22](#)
 - macro output field configuration for vertical macros [23](#)
 - macro types [21](#)
 - output field inclusion for vertical macros [24](#)
 - overview [21](#)
 - transformation output field for horizontal macros [29](#)
 - using constants in horizontal macros [28](#)
 - vertical macro configuration [21](#)
- Expression transformation
 - advanced properties [65](#)
 - variable fields [18](#)
- Expression transformations
 - expression fields [61](#)
 - in mappings [61](#)
- :EXT reference qualifier
 - reserved word [64](#)

F

- FALSE constant
 - reserved word [64](#)

- field expression [62](#)
- field expressions
 - comments, adding [63](#)
 - components [62](#)
 - literals [63](#)
 - reserved words [64](#)
 - syntax [63](#)
- field mapping
 - Output transformation [110](#)
- field mappings
 - in Maplet transformations [99](#)
 - in Target transformations [53](#)
 - in the Normalizer transformation [105](#)
- field name conflicts
 - resolving in mappings [14](#)
 - transformations [12](#)
- field rules
 - field selection criteria [13](#)
 - in mappings [13](#)
- field selection criteria [13](#)
- file lists
 - batch files [32](#)
 - command sample file [33](#)
 - commands [32](#)
 - in Lookup transformations [34](#)
 - in Source transformations [33](#)
 - manually created [31](#)
 - overview [31](#)
 - rules and guidelines [31](#)
 - shell scripts [32](#)
 - text file format [31](#)
- Filter transformation
 - advanced properties [67](#)
- Filter transformations
 - filter condition [66](#)
 - in mappings [66](#)
- flat files
 - command sample file [33](#)
 - configuration in mapping sources [36](#)
 - file lists [31](#)
- functions
 - definition [62](#)

G

- group by fields
 - Aggregator transformation [56](#)
- group filter condition
 - configuring [122](#)
 - Router transformation [121](#)
- groups
 - Router transformation [121](#)

H

- hierarchical data and multibyte characters [34](#), [43](#), [52](#)

I

- Incoming Fields are Sorted
 - Aggregator transformation [57](#)
- index cache [19](#)
- :INFA reference qualifier
 - reserved word [64](#)

- Informatica Global Customer Support
 - contact information [9](#)
- Informatica Intelligent Cloud Services
 - web site [8](#)
- input fields [68](#)
- Input transformation
 - input fields [68](#)

J

- Joiner transformation
 - advanced properties [70](#)
- Joiner transformations
 - comparison with Union [152](#)
 - creating [71](#)
 - example [72](#)
 - in mappings [69](#)
 - join condition [69](#)
 - join types [70](#)

L

- literals
 - definition [62](#)
 - single quotation mark requirement [63](#)
 - string and numeric [63](#)
- :LKP reference qualifier
 - reserved word [64](#)
- lookup caches
 - dynamic [86](#)
- lookup source filter
 - Lookup transformation [85](#)
- lookup SQL overrides
 - examples [83](#)
 - Lookup transformation [83](#)
 - query guidelines [84](#)
- Lookup transformation
 - advanced properties [80](#)
 - dynamic cache [86](#)
 - dynamic cache inserts and updates [87](#)
 - field mapping [89](#)
 - file name prefix [91](#)
 - generated key fields [89](#)
 - ignore fields in comparison [90](#)
 - insert else update [87](#)
 - lookup return fields [79](#)
 - lookup source filter [85](#)
 - lookup SQL override example [83](#)
 - lookup SQL override guidelines [84](#)
 - lookup SQL overrides [83](#)
 - NewLookupRow [87](#)
 - non-persistent lookup cache [91](#)
 - persistent lookup cache [91](#)
 - re-cache from lookup source [91](#)
 - rebuilding the lookup cache [91](#)
 - Sequence-ID field [89](#)
 - synchronizing lookup source with dynamic cache [88](#)
- Lookup transformations
 - :LKP expression syntax [93](#)
 - calling unconnected lookups [93](#)
 - configuring file lists [34](#)
 - configuring unconnected lookups [93](#)
 - custom lookup source queries [78](#)
 - data preview [17](#)
 - in mappings [75](#)
 - lookup condition [78](#)

Lookup transformations (*continued*)

- lookup object [76](#)
- lookup object configuration [76](#)
- lookup object properties [77](#)
- multiple match policy restrictions [77](#)
- unconnected lookup example [95](#)
- unconnected lookups [92](#)

M

- macro input fields
 - in mappings [21](#)
- maintenance outages [9](#)
- mapping designer
 - transformations [10](#)
- mappings
 - adding mapplets [98](#)
 - configuring aggregate calculations with the Aggregator transformation [56](#)
 - custom lookup source queries [78](#)
 - custom source queries [42](#)
 - example of using a Union transformation [155](#)
 - filtering data with a Filter transformation [66](#)
 - filtering source data [43](#)
 - joining heterogeneous sources with a Joiner transformation [69](#)
 - look up data with a Lookup transformation [75](#)
 - lookup object configuration [76](#)
 - Mapplet transformations [98](#)
 - normalizing data with the Normalizer transformation [103](#)
 - output fields in a Union transformation [154](#)
 - performing calculations with an Expression transformation [61](#)
 - planning to use a Union transformation [153](#)
 - sorting source data [43](#)
 - source configuration [35](#)
 - Source transformations [35](#)
 - SQL transformation [135](#)
 - target configuration [46](#)
 - Target transformations [46](#)
 - Union transformation [152](#)
 - using expression macros [21](#)
- mapplet
 - parameters [100](#)
- Mapplet transformations
 - configuring [98](#)
 - field mappings [99](#)
 - in mappings [98](#)
 - output fields [101](#)
 - purpose [98](#)
 - selecting a mapplet [99](#)
- mapplets
 - selecting in Mapplet transformations [99](#)
- :MCR reference qualifier
 - reserved word [64](#)
- metadata override
 - editing transformation data types [45](#)
 - source fields [44](#)
 - target fields [52](#)
- Microsoft SQL Server
 - configuration in mapping sources [38](#)
 - configuration in mapping targets [48](#)
- multibyte data configuration [34](#), [43](#), [52](#)
- MySQL
 - configuration in mapping sources [38](#)
 - configuration in mapping targets [48](#)

N

- NEXTVAL [126](#)
- normalized fields
 - Normalizer transformation [103](#)
- Normalizer transformation
 - advanced properties [106](#)
- Normalizer transformations
 - example in mapping [107](#)
 - field mapping [105](#)
 - field mapping options [105](#)
 - field rule for parameterized sources [106](#)
 - generated keys [104](#)
 - handling unmatched groups of multiple-occurring fields [104](#)
 - normalized fields [103](#)
 - occurs configuration [103](#)
 - overview [103](#)
 - target configuration [106](#)
- NOT
 - reserved word [64](#)
- NULL constant
 - reserved word [64](#)

O

- operators
 - definition [62](#)
- OR
 - reserved word [64](#)
- Oracle
 - configuration in mapping sources [38](#)
 - configuration in mapping targets [48](#)
- output fields
 - Output transformation [110](#)
- Output transformation
 - field mapping [110](#)
 - output fields [110](#)

P

- PROC_RESULT variable
 - reserved word [64](#)

Q

- quotation marks
 - inserting single using CHR function [63](#)

R

- Rank transformation
 - advanced properties [117](#)
 - caches [113](#)
 - case-sensitive string comparison [117](#)
 - configuring as optional [117](#)
 - configuring cache directory [117](#)
 - configuring cache sizes [117](#)
 - defining [114](#)
 - example [118](#)
 - fields [114](#)
 - overview [112](#)
 - rank groups [116](#)
 - rank index [114](#)
 - rank order [115](#)

Rank transformation (*continued*)

- RANKINDEX field [114](#)
- ranking string values [112](#)
- selecting rows to rank [115](#)
- steps to create [114](#)
- tracing level [117](#)
- transformation scope [117](#)

reserved words

- list [64](#)

Router transformation

- advanced properties [123](#)
- configuring a filter condition [122](#)
- examples [123](#)
- group filter condition [121](#)
- groups [121](#)
- output group guidelines [121](#)
- overview [120](#)

routing rows

- transformation for [120](#)

S

:SD reference qualifier

- reserved word [64](#)

:SEQ reference qualifier

- reserved word [64](#)

Sequence Generator transformation

- disable incoming fields [128](#)
- output fields [126](#)
- properties [127](#)
- rules and guidelines [129](#)

sorter cache

- description [130](#)

Sorter transformation

- advanced properties [131](#)
- cache size [131](#)
- caches [130](#)
- overview [130](#)
- sort conditions [130](#)
- work directory [131](#)

Source transformations

- advanced relationships [42](#)
- configuring file lists [33](#)
- custom source queries [42](#)
- data preview [17](#)
- database sources [38](#)
- editing transformation data types [45](#)
- filtering data [43](#)
- in mappings [35](#)
- joining related objects [39](#)
- sorting data [43](#)
- source configuration [35](#)
- source fields [44](#)

:SP reference qualifier

- reserved word [64](#)

SPOUTPUT

- reserved word [64](#)

SQL overrides

- dynamic lookup cache [90](#)

SQL queries

- SQL transformations [139](#)

SQL transformation

- advanced properties [149](#)
- call from an expression [138](#)
- unconnected [137–139](#)
- unconnected SQL transformation [137](#)

SQL transformations

- configuration [144](#)
- configuring the SQL type [145](#)
- dynamic queries [141](#)
- entering a query [146](#)
- field mapping [146](#)
- NumRowsAffected field [147](#)
- output fields [147](#)
- overview [135](#)
- parameterizing a query [146](#)
- passing the full query [141](#)
- passive mode [142](#)
- query guidelines [143](#)
- query processing [139](#)
- selecting a stored procedure or function [145](#)
- selecting multiple rows [140](#)
- SQL statements for queries [143](#)
- SQLException field [147](#)
- static queries [140](#)
- stored function processing [135](#)
- stored procedure processing [135, 137–139](#)
- substituting the table name [142](#)

status

- Informatica Intelligent Cloud Services [9](#)

stored functions

- SQL transformation [135](#)

stored procedures

- SQL transformation [135, 137](#)

string literals

- single quotation mark requirement [63](#)

strings

- ranking string values [112](#)

synchronization

- source fields [44](#)
- target fields [52](#)

syntax

- for field expressions [63](#)

system status [9](#)

T

Target transformations

- creating a database target at run time [49](#)
- data preview [17](#)
- database targets [47](#)
- database targets created at run time [49](#)
- entering a target update statement [52](#)
- field mappings [53](#)
- in mappings [46](#)
- specifying targets [54](#)
- target configuration [46](#)
- target fields [52](#)
- target update override [50](#)
- target update override guidelines [51](#)
- update columns [50](#)

:TD reference qualifier

- reserved word [64](#)

transformation cache

- tuning [20](#)

transformation caches [19](#)

transformations

- active and passive [10](#)
- connecting [10](#)
- field name conflicts [12](#)
- field rules [13](#)
- incoming fields [12](#)
- overview [10](#)

transformations (*continued*)

 previewing fields [12](#)

 Rank [112](#)

 renaming fields [14](#)

 Router [120](#)

 types [11](#)

TRUE constant

 reserved word [64](#)

trust site

 description [9](#)

U

Union transformation

 advanced properties [155](#)

 example [155](#)

 field mappings [154](#)

 output fields [154](#)

 overview [152](#)

Union transformations

 comparison with Joiner [152](#)

 guidelines [153](#)

Union transformations (*continued*)

 input group guidelines [153](#)

 update columns

 configuring [50](#)

 in Target transformations [50](#)

 upgrade notifications [9](#)

V

variable fields

 in Expression transformations [18](#)

W

web site [8](#)

work directory

 Sorter Transformation [131](#)

WORKFLOWSTARTTIME variable

 reserved word [64](#)