



Informatica® Data Integration - Free & PayGo
April 2023

Components

© 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: 2023-04-03

Table of Contents

Preface	5
Informatica Resources.	5
Informatica Documentation.	5
Informatica Intelligent Cloud Services web site.	5
Informatica Intelligent Cloud Services Communities.	5
Informatica Intelligent Cloud Services Marketplace.	5
Data Integration connector documentation.	6
Informatica Knowledge Base.	6
Informatica Intelligent Cloud Services Trust Center.	6
Informatica Global Customer Support.	6
 Chapter 1: Components.....	 7
 Chapter 2: Mapplets.....	 8
Active and passive mapplets.	8
Mapplet input and output.	8
Mapplet input.	8
Mapplet output.	9
Transformation names.	9
Parameters in mapplets.	10
Creating a mapplet.	10
Editing a mapplet.	11
Changes that affect dependencies.	11
Synchronizing a mapplet.	11
 Chapter 3: Shared sequences.....	 12
Shared sequence properties.	12
Number of reserved values.	13
Creating a shared sequence.	14
Using a shared sequence.	14
Resetting a shared sequence.	14
 Chapter 4: User-defined functions.....	 15
Creating user-defined functions.	15
User-defined function general properties.	16
User-defined function arguments.	16
User-defined function expression.	17
Editing and deleting user-defined functions.	17
Creating expressions with user-defined functions.	17

Index..... 19

Preface

Use *Components* to learn about reusable assets that you can create in Data Integration. Learn how to create components and add them to transformations, mappings, and tasks.

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

Components

Components are assets that support mappings and tasks.

You can use the following components:

Mapplets

Define reusable transformation logic to use in Mapplet transformations in a mapping.

For more information about mapplets, see [Chapter 2, “Mapplets” on page 8](#).

Shared sequences

Define a reusable sequence to use in multiple Sequence Generator transformations.

For more information about shared sequences, see [Chapter 3, “Shared sequences” on page 12](#).

User-defined functions

Create reusable functions that you can use in transformation expressions and in other user-defined functions.

For more information about user-defined functions, see [Chapter 4, “User-defined functions” on page 15](#).

Create components on the **New Asset** page.

CHAPTER 2

Mapplets

A mapplet is reusable transformation logic that you can use to transform source data before it is loaded into the target.

Create mapplets in Data Integration in the Mapplet Designer in the same way that you create mappings in the Mapping Designer. After you create a mapplet, you can add it to a Mapplet transformation to use its transformation logic. You can use any transformation in a mapplet.

You can use a mapplet in another mapplet. However, you cannot cyclically reference a mapplet. For example, if Mapplet A uses Mapplet B, Mapplet B cannot also use Mapplet A.

Active and passive mapplets

Mapplets can be either active or passive.

Passive mapplets contain a single input group, a single output group, and only passive transformations.

Active mapplets contain at least one active transformation.

When you use a mapplet in a Mapplet transformation, the mapplet type displays on the **Mapplet** tab.

Mapplet input and output

To use a mapplet in a Mapplet transformation, you must configure the mapplet input and output.

A mapplet receives input from an upstream transformation through an Input transformation, from a Source transformation, or both.

A mapplet passes output to a downstream transformation through an Output transformation, to a Target transformation, or both.

A mapplet must contain at least one Input transformation or one Output transformation.

Mapplet input

Mapplet input can be an Input transformation, a Source transformation, or both.

Use an Input transformation when you want the mapplet to receive input data from one or more upstream transformations. You can use multiple Input transformations in a mapplet. When you use the mapplet in a Mapplet transformation, each Input transformation becomes an input group. Use multiple Input

transformations when you have multiple pipelines in a mapplet, or when you want the mapplet to receive input from multiple upstream transformations.

You include one or more Source transformations in a mapplet to provide source data. When you use only Source transformations for mapplet input, the mapplet is the first object in the mapping pipeline and contains no input groups. If the Source transformation reads hierarchical data, you must specify the validation type in the Source transformation.

A mapplet must contain at least one Input transformation or Source transformation.

Note: Not all source types can be used in a Source transformation in a mapplet. For more information, see the help for the appropriate connector.

For information about configuring Input and Source transformations, see *Transformations*.

Mapplet output

Mapplet output can be to an Output transformation, a Target transformation, or both.

Use an Output transformation when you want the mapplet to pass data to one or more downstream transformations. When you use the mapplet in a Mapplet transformation, each Output transformation becomes an output group. Each output group can pass data to one or more pipelines in a mapping.

Use a Target transformation when you want the mapplet to write data to a target. When you use a Target transformation with no Output transformation, the mapplet is the last object in the mapping pipeline.

A mapplet must contain at least one Output transformation or Target transformation.

For information about configuring Output and Target transformations, see *Transformations*.

Transformation names

When you use a mapplet in a Mapplet transformation, Data Integration renames the transformations in the mapplet at run time.

When you use a mapplet in a Mapplet transformation, the mapplet might contain transformations with names that conflict with other transformation names in the mapping. To avoid name conflicts, Data Integration prefixes the names of the transformations in the mapplet with the Mapplet transformation name.

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 a 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.

Data Integration truncates transformation names that contain more than 80 characters.

Parameters in mapplets

You can use input parameters and in-out parameters in a mapplet. You specify the value of the parameters when you configure the mapping task.

You configure parameters in mapplets the same way that you configure parameters in mappings. For information about configuring parameters, see *Mappings*.

When you include parameters in a mapplet, Data Integration renames the parameters when you use the mapplet in a Mapplet transformation. The parameter names are prefixed with the name of the Mapplet transformation. You can view the corresponding names on the **Parameters** tab of the Mapplet transformation Properties panel.

For example, you use a Filter transformation in a mapplet and select **Completely Parameterized** as the filter condition. You create a string parameter called "Filter_Param." You use the mapplet in a Mapplet transformation named "MyMappletTx." In the Mapplet transformation, the filter parameter is renamed to "MyMappletTx_Filter_Param."

Creating a mapplet

Create a mapplet in the Mapplet Designer.

The Mapplet Designer contains the same design areas and functionality as the Mapping Designer. For information about using the Mapping Designer, see *Mappings*.

1. Click **New > Mapplets > Mapplet**.

The Mapplet Designer appears with an Input transformation and an Output transformation on the canvas.

2. Specify the mapplet name and location.

The default name for the mapplet is `Mapplet` followed by a sequential number.

If the **Explore** page is currently active and a project or folder is selected, the default location for the asset is the selected project or folder. Otherwise, the default location is the location of the most recently saved asset.

You can change the name or location after you save the mapping using the **Explore** page.

3. Optionally, enter a description for the mapplet.

When you use the mapplet in a Mapplet transformation, the description appears in the **Mapplet** tab.

4. Optionally, filter the transformations in the transformation palette.

5. Configure the mapplet input.

Input can be a Source transformation, an Input transformation, or both. If you use an Input transformation, add input fields for the fields you want to pass to the mapplet.

6. Add transformations to the mapplet.

7. Configure the mapplet output.

Output can be a Target transformation, an Output transformation, or both. If you use an Output transformation, add output fields for the data you want to pass to the downstream transformation.

8. Validate the mapplet.

9. Resolve errors on the **Mapping** tab in the **Validation** panel.

10. Save the mapplet.

Editing a mapplet

Edit a mapplet in the Mapplet Designer. To edit a mapplet, open the mapplet from the Explore page.

When you edit a mapplet, the Mapplet Designer validates the transformation logic. If a mapplet is valid when you save it, Data Integration deploys the changes to each mapping and mapplet that uses the mapplet.

Use caution when you edit mapplets that are used in mappings and other mapplets. If you edit the mapplet interface, mappings and mapplets that use the mapplet will have validation errors. To resolve the validation errors, you must synchronize the Mapplet transformations that use the mapplet to get the latest changes. If you do not synchronize the mapplet transformation and you run the task, the task fails.

To see the assets that use the mapplet, in the Mapplet Designer, open the **Actions** menu and select **Show Dependencies**.

Changes that affect dependencies

Changes to a mapplet might affect the mappings and mapplets that use the mapplet.

The following changes to a mapplet do not affect the assets that use the mapplet:

- Change transformation names, descriptions, or properties.
- Add or remove transformations in the mapplet, as long as you do not change the mapplet type from active to passive or from passive to active.

The following changes to a mapplet change the interface of the mapplet and affect the assets that use the mapplet:

- Add or remove transformations that change the mapplet type from active to passive or from passive to active.
- Change the data type, precision, or scale of a connected input or output field.
- Add or remove input or output fields
- Add or remove Input or Output transformations

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 3

Shared sequences

Shared sequences are reusable sequences that you can use in multiple Sequence Generator transformations. When you use a shared sequence, the Sequence Generator transformation uses the properties of the shared sequence to generate values. Use a shared sequence when you want to assign numeric values within the same sequence to data in multiple mapping tasks.

Multiple mappings and mapplets can use the same shared sequence. When you run the mapping task, Data Integration reserves a set of values in the sequence so that each mapping task generates unique values.

For example, you want to assign a unique ID to entries in your customer tables. You have two mappings that load customer data to the same target table, and the mappings are scheduled to run at the same time. Use a shared sequence to ensure that you do not get duplicate IDs in the target.

You create shared sequences on the **New Assets** page. You can create non-shared sequences in a Sequence Generator transformation. For more information about creating non-shared sequences, see *Transformations*.

Shared sequence properties

When you create a shared sequence, you define general properties such as the sequence name, and properties that define the sequence such as initial value, increment value, and end value.

The following table lists the properties you define for a shared sequence:

Property	Description
Name	Required. Name of the shared sequence. Can contain alphanumeric characters and underscores (_). Maximum length is 200 characters.
Location	Project or folder in which the sequence resides. If the Explore page is currently active and a project or folder is selected, the default location for the asset is the selected project or folder. Otherwise, the default location is the location of the most recently saved asset.
Description	Optional. Description for the shared sequence.
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 is 2,147,483,647.

Property	Description
End Value	Maximum value that the mapping task generates. If the sequence reaches this value during the task run with rows left to process and the sequence is not configured to cycle, the run fails. If the sequence is configured to cycle, the sequence uses this value and then starts over at the Cycle Start Value. Maximum is 9,223,372,036,854,775,807.
Initial Value	The value you want the mapping task to use as the initial value in the sequence the first time that the sequence is used. If you want to cycle through a series of values, this value must be greater than or equal to the Cycle Start Value and less than the End Value. If you reset the shared sequence, the sequence is reset to this value. Default is 0.
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 when the sequence is configured to cycle. When the sequence reaches the end value, it cycles back to this value. Default is 0.
Number of Reserved Values	The number of sequence values the mapping task caches at one time. Must be greater than or equal to 1000.
Current Value	Displays the current start value in the sequence.

Number of reserved values

The number of reserved values determines the number of values in the sequence that the mapping task caches at one time during each task run.

When multiple Sequence Generator transformations use the same shared sequence, multiple instances of the shared sequence can exist at the same time. To avoid generating duplicate values, reserve a range of sequence values for each mapping by configuring the number of reserved values.

For example, you configure a shared sequence as follows: Number of Reserved Values = 1000, Initial Value = 1, Increment By = 1. Two mappings use the shared sequence. When the first mapping task runs, Data Integration reserves 1000 values (1-1000) for the mapping task and updates the current value in the repository to 1001. When the second mapping task runs, Data Integration reserves the next 1000 values (1001-2000) and updates the current value to 2001. When either mapping task uses all of its reserved values, Data Integration reserves a new set of values and updates the current value.

The number of values that Data Integration reserves in a sequence is determined by multiplying the Number of Reserved Values by the Increment By value.

For example, if you have a shared sequence that begins at 2, increments by 2 and the number of reserved values is 1000, Data Integration reserves 1000 values in the sequence, or numerical values 2-2002, and updates the current value to 2003.

By default, the number of reserved values is 1000. To increase performance, you can increase the number of reserved values. This reduces the number of calls that Data Integration must make to the repository. The number of reserved values cannot be less than 1000.

Values that are reserved for a task run but not used in the task are lost when the task completes. For example, you generate IDs in a table with 1250 rows. You configure the shared sequence to generate values in increments of 1 and set the number of reserved values to 2000. When the task runs, Data Integration reserves 2000 values for the task and updates the current value to 2001. When the next task runs, the sequence begins at 2001, and values 1251-2000 are lost.

Creating a shared sequence

Create a shared sequence on the **New Asset** page.

1. Click **New > Components > Shared Sequence** and then click **Create**.
2. Define the sequence properties.
3. Save the sequence.

Using a shared sequence

You can use a shared sequence in a Sequence Generator transformation in a mapping or maplet.

1. Open the mapping or maplet.
2. Add a Sequence Generator transformation to the canvas and connect it to a downstream transformation.
3. On the **Sequence** tab, select **Use Shared Sequence**.
4. Select the sequence to use.
5. Configure the Sequence Generator advanced properties and output fields.
For more information about configuring a Sequence Generator transformation, see *Transformations*.

Note: If you do not select **Use Shared Sequence**, the Sequence Generator transformation generates a non-shared sequence.

Resetting a shared sequence

Reset a shared sequence to update the current value of the sequence to the defined initial value.

1. Open the shared sequence.
2. Click **Edit**.
3. Click **Reset Current Value** in the **Sequence Properties** area.
4. Save the shared sequence.

CHAPTER 4

User-defined functions

User-defined functions are reusable functions that you can use in expressions. Create user-defined functions to build complex expressions using the Informatica Intelligent Cloud Services transformation language. User-defined functions use the same syntax and can use the same transformation language components as transformation and field expressions.

You can include a user-defined function in a transformation expression in a mapping or mapplet, in a field expression in a mapping task, or in another user-defined function. To use a user-defined function in an expression, you select the user-defined function in the expression editor and enter the required arguments.

Example

You want to remove leading and trailing spaces from a text string such as a name or address. You create a user-defined function called `RemoveSpaces` that takes a text string as an argument and performs the `LTRIM` and `RTRIM` functions. When you configure the user-defined function, you enter the following expression:

```
LTrim( RTrim(TextString) )
```

After you create the function, you use it in an Expression transformation to remove leading and trailing spaces from the incoming field, `LAST_NAME`. The expression field contains the following expression:

```
:UDF.RemoveSpaces (LAST_NAME)
```

In the expression, the user-defined function name is prefaced with `:UDF`. The incoming field `LAST_NAME` is passed to the function as the argument.

Creating user-defined functions

Create a user-defined function on the **New Asset** page. The user-defined functions that you create are available to all users in your organization based on their privileges.

1. Click **New > Components > User-Defined Function** and then click **Create**.
2. Configure general properties such as the function name, location, and return type.
3. Optionally, create arguments for the function.
When you create arguments, configure the name, data type, and description for each argument.
4. Create the function expression.
5. Validate and save the function.

User-defined function general properties

When you create a user-defined function, you must define general properties such as the function name and return type. Define general properties on the **General** tab.

The following table describes the properties:

Property	Description
Name	Name of the function. Must be unique within an organization. The name must begin with a letter and can contain letters, numbers, and the following special characters: _ @ \$ # The name cannot exceed 100 characters and cannot contain spaces.
Location	Location of the user-defined function. Browse to the folder where you want to store the user-defined function or use the default location. The default location is the location of the most recently saved asset.
Description	Description of the user-defined function. The description appears in the expression editor when you select the function in a transformation expression, a field expression, or another user-defined function.
Return Type	Data type of the values that the function returns. Can be binary, date, numeric, or string.

User-defined function arguments

When you create a user-defined function, you can include arguments to define the values that you pass to the function. Create arguments on the **Arguments** tab. You can create up to 10 arguments.

To create an argument, click **Add**, and enter the following information:

Property	Description
Name	Name of the argument. Must be unique within the function. The name must begin with a letter and can contain letters, numbers, and underscore characters (_). It cannot exceed 100 characters and cannot contain spaces.
Type	Data type of the argument. Can be binary, date, numeric, or string.
Description	Description of the argument. The description appears in the Arguments and Functions area on the Expression tab when you select the argument. It also appears in the expression editor when you use the function in a transformation or field expression.

Arguments are passed to the function in the order in which they appear on the **Arguments** tab. To rearrange the order, select an argument and click **Move up** or **Move down**.

To delete an argument, click **Delete** in the row that contains the argument.

User-defined function expression

Create the expression that defines the function on the **Expression** tab. The function expression can contain constants, operators, built-in functions, and other user-defined functions. You can create a complex expression by nesting functions within functions.

To create an expression, enter the expression in the expression editor. The expression that you create must follow the same guidelines and use the same syntax as any transformation or field expression. For more information about creating expressions, see *Tasks*.

You can add arguments, built-in functions, and other user-defined functions to the expression by selecting them in the Arguments and Functions area of the expression editor and clicking **Add**. Alternatively, press the **Ctrl + Space** keys to see a list of recommended arguments and functions in-line. You can also type in the expression manually.

An expression cannot contain cyclic references. For example, the expression for user-defined function Function_A cannot include Function_A. Also, if user-defined function Function_A calls user-defined function Function_B, then the expression for Function_B cannot call Function_A.

To validate the expression, click **Validate**. Data Integration validates the expression. It does not validate the user-defined functions and expressions that use the function.

Editing and deleting user-defined functions

Certain restrictions apply when you edit or delete a user-defined function.

Editing a user-defined function can affect the expressions and functions that use it. Consider the following guidelines when you edit a user-defined function:

- If you edit a user-defined function and the updated function is valid, Data Integration propagates the changes to all expressions and user-defined functions that use the function.

Sometimes this can make the expressions and functions that use the user-defined function invalid. For example, you edit a function, change the number of arguments from one to two, and validate the function. Even though the updated function is valid, expressions and functions that use the user-defined function become invalid if they pass one argument to the function.
- If you edit and save a user-defined function but the updated function is not valid, Data Integration does not propagate the changes to the expressions and user-defined functions that use it.
- You cannot rename a user-defined function after you save it.

To delete a user-defined function, you must first remove it from all expressions and functions that use it. You cannot delete a user-defined function that is used in an expression or in another user-defined function.

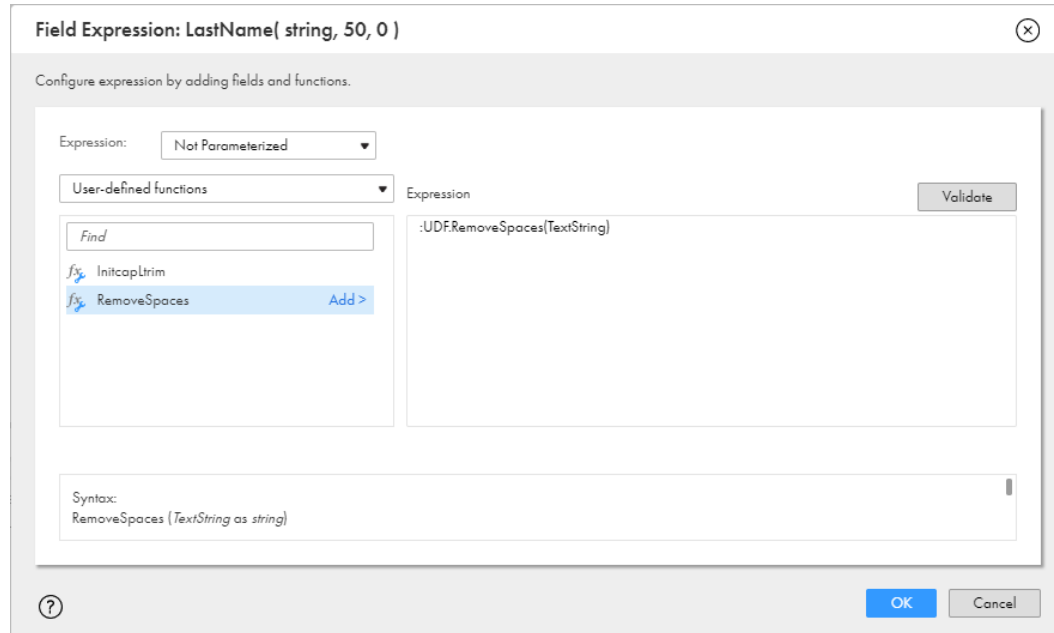
Tip: View the dependencies for the user-defined function to find the assets that use the function.

Creating expressions with user-defined functions

You can add a user-defined function to a transformation or field expression.

When you create an expression, valid user-defined functions appear in the expression editor. If you type in an expression manually, precede each user-defined function with :UDF.

The following image shows a user-defined function selected in the expression editor of an Expression transformation:



When you select a user-defined function, the expression editor shows the function syntax in the following format:

<function name> (<argument 1> as <data type>, <argument N> as <data type>)

For example:

```
RemoveSpaces(TextString as string)
```

When you add the function to the expression, the function includes the prefix :UDF, as shown in the following example:

```
:UDF.RemoveSpaces(TextString)
```

After you add the function to the expression, replace the arguments with field names or parameters. Do not include the table name in the argument. Use only the field name as shown in the following example:

```
:UDF.RemoveSpaces(NAME)
```

For more information about creating expressions, see *Tasks*.

When you validate the expression, Data Integration does not validate the user-defined function. It only validates the expression.

INDEX

A

active mapplets
description [8](#)
asset components [7](#)

C

Cloud Application Integration community
URL [5](#)
Cloud Developer community
URL [5](#)
components
mapplets [8](#)
shared sequence [12](#)

D

Data Integration community
URL [5](#)

I

Informatica Global Customer Support
contact information [6](#)
Informatica Intelligent Cloud Services
web site [5](#)

M

maintenance outages [6](#)
mapplet
output [8](#)
Mapplet
input [8](#)
mapplet input [8](#)
mapplet output [9](#)
mapplet parameters [10](#)
mapplets
active and passive [8](#)
overview [8](#)

N

number of reserved values [13](#)

P

parameters
in mapplets [10](#)
passive mapplets
description [8](#)
properties
shared sequence [12](#)

S

shared sequence
creating a shared sequence [14](#)
number of reserved values [13](#)
resetting a shared sequence [14](#)
status
Informatica Intelligent Cloud Services [6](#)
system status [6](#)

T

trust site
description [6](#)

U

upgrade notifications [6](#)
user-defined functions
arguments [16](#)
creating [15](#)
deleting [17](#)
editing [17](#)
expression [17](#)
general properties [16](#)
overview [15](#)
return type [16](#)

W

web site [5](#)