Oracle® Cloud

Using the Oracle Mapper with Oracle Integration

E85415-11
November 2020

Oracle Cloud Using the Oracle Mapper with Oracle Integration,

E85415-11

Primary Author: Oracle Corporation

Contents

Preface

Using the Oracle Mapper with Oracle Integration describes how to use the mapper to map source data structures to target data structures.

> **Note:**
>
> The information in this guide applies to all of your Oracle Integration instances. It doesn't matter which edition you're using, what features you have, or who manages your cloud environment. You'll find what you need here, including notes about any differences between the various flavors of Oracle Integration when necessary.

Topics

Audience

Using the Oracle Mapper with Oracle Integration is intended for users who want to use the mapper to map source data structures to target data structures.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Resources

See these Oracle resources:

- Oracle Cloud

  `http://cloud.oracle.com`
- *Using Integrations in Oracle Integration*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Get Started with the Mapper

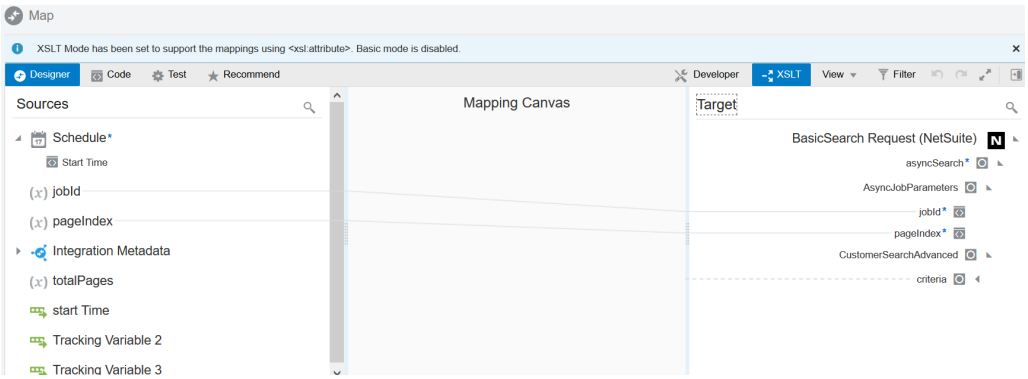
Review the following topics for an overview of how to use the mapper to map source data structures to target data structures.

Topics

About Mappings

One of the key tasks to any integration is defining how data is transferred, or *mapped*, between two applications.

In most cases, the messages you want to transfer between the applications in an integration have different data structures. A visual mapper enables you to map element nodes between applications by dragging source element nodes onto target element nodes. When you open the mapper for a request or response message in an integration, the data structures are automatically populated with the information pulled from the source and target connections. You can expand and load data structure levels on demand to display additional levels. There is no limit on the levels of display.



The maps you create are called transformation maps, and use the eXtensible Stylesheet Language (XSL) to describe the data mappings, which let you perform complex data manipulation and transformation. A standard set of XSLT constructs are provided (for example, `xsl:if`, `xsl:for-each`, and others). A specialized function is also provided for you to reference lookups directly from the mapper.

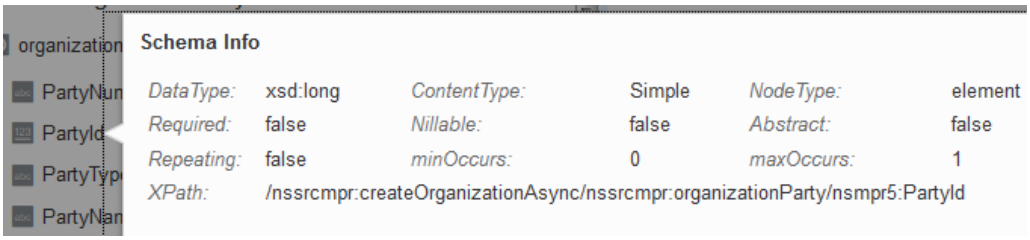
The mapper supports both qualified and unqualified schemas (that is, schemas without `elementFormDefault="qualified"`). Elements and attributes with and without namespace prefixes are also supported.

Substitution groups in schemas are supported. You can see all the substitutable elements in a base element in the mapper, and select the one to use.

Extended data types are also supported.

Elements and attributes for which mapping is required are identified by a blue asterisk (*) to the left of their names. To display only required fields, click the **Filter** icon in the mapper toolbar, select **Required Fields**, and click **Apply**.

You can also right-click elements and attributes and select **Node Info** to show specific schema details such as the data type, if mapping is required, and so on.



Additional custom annotations can also be displayed. These annotations are currently only available with the Oracle Sales Cloud Adapter. The Oracle Sales Cloud Adapter obtains this information from the applications and annotates it in the integration WSDL. This information is then read and made visible as annotations in the mapper (for example, title and description). This information can help you better understand what data is being mapped.

The mapper toolbar provides the following functionality.

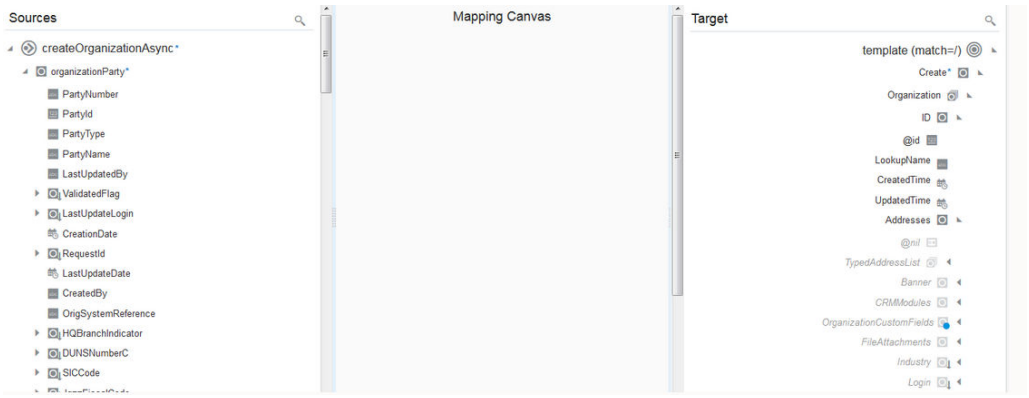
Element	Description
Designer	Click to return to the mapping canvas when you are inside the Code, Test, or Recommend page.
Code	You can view the XSLT code being created as you design your mappings.
Test	Once you complete designing your mappings, you can test them by entering sample content of the message to process in the mapping tester.
Recommend	If you enable the recommendations engine, you can accept the target element recommendations of the engine when creating mappings. This eliminates the need to analyze and perform each individual source-to-target mapping.
Developer	Click to disable user-friendly, source and target element names in the mapper. By default, user-friendly element names are shown.
XSLT	Click to show the XSLT functions.

Element	Description
View	You can select the following options: • Select to show the namespace prefixes on source and target element nodes. • Select to show the types (prefixes and data types) on source and target element nodes.
Filter	You can filter the display of element nodes, error messages, and warnings in the source or target data structures.
	You can select to undo the previous action performed in the mapper. For example, if you perform a mapping, then press this button, the mapping is removed. The link is disabled when all actions have been undone.
	You can redo the action that was undone.
	You can maximize the size of the mapper. This is useful when working with large schemas.
	You can add functions, operators, and XSLT expressions to your mappings.

About Mapping Data Between Applications

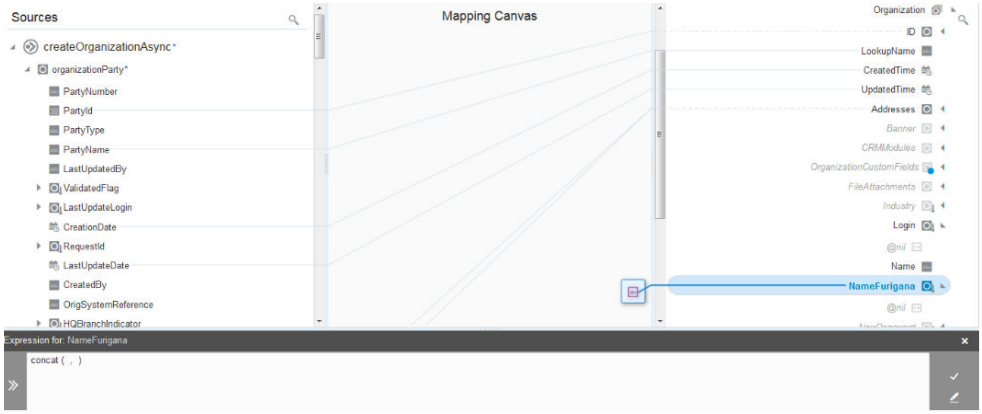
Once you create an integration and have the source and target connections in place, you can define how data is mapped between the element nodes in the two data structures.

The mapper appears with the element nodes of the source data structure on the left and the target data structure on the right.



1. To map fields directly, click a source element nodes and drag it to the corresponding field in the target element node.

   A blue line connects the two nodes. An Expression Builder below the mapper is displayed to show the XPath expression.



2. To use functions, operators, or XSLT statements in your mapping, see Work with Functions, Operators, and XSLT Statements.
3. When you are done mapping data, click **Close**, then click **Apply** to save your changes when prompted. You can also click **Validate** to save your changes.

View User-Friendly Element Names

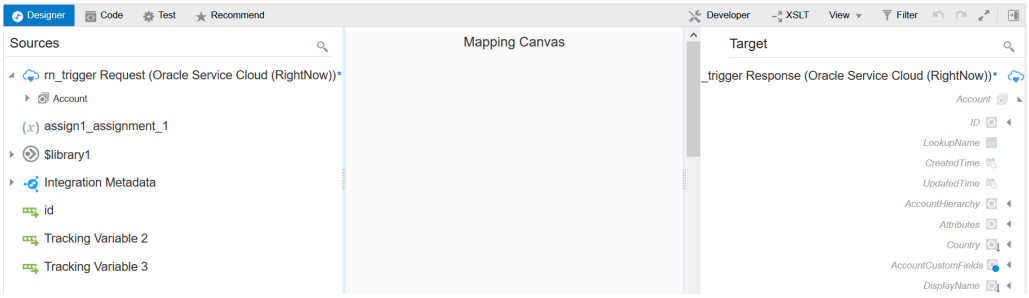
You can view user-friendly display names instead of technical names for source and target elements in the mapper tree and for expressions in the Expression Builder. This eliminates the need to try and understand the technical, often cryptic, names that are difficult to correlate to the user-friendly display names you see in the endpoint application's user interface. User-friendly names are displayed by default, but you can also toggle to the technical names.

- Toggle Between User-Friendly Names and Technical Names
- Adapter Names
- Root Elements in Source and Target Trees
- Child Elements in Source and Target Trees
- Search For Data in the Source and Target Trees
- User-Friendly Expression for Mapping
- Expression Builder
- Other Sections of the User Interface

Toggle Between User-Friendly Names and Technical Names

By default, user-friendly names are displayed in the source and target mapper trees when you open the mapper. Name display is controlled by the

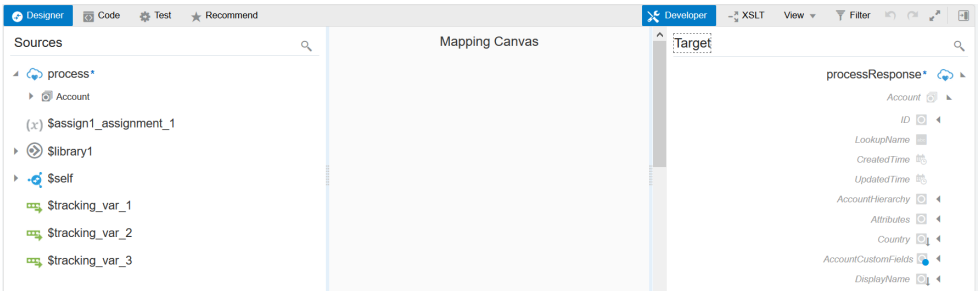
button at the top of the mapper.



1. Click

   Developer

   to switch to technical names.
   The button changes colors to blue and technical names for the source and target elements are displayed.



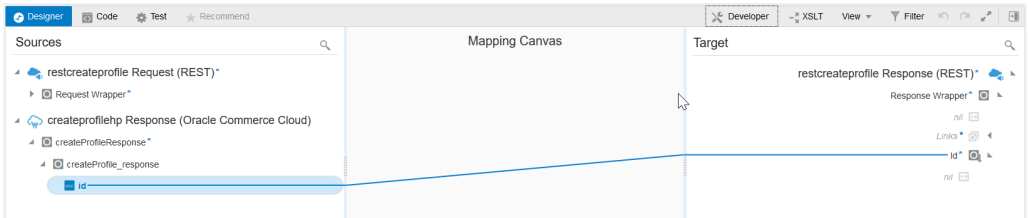
2. Click

   Developer

   to switch back to user-friendly names.

Adapter Names

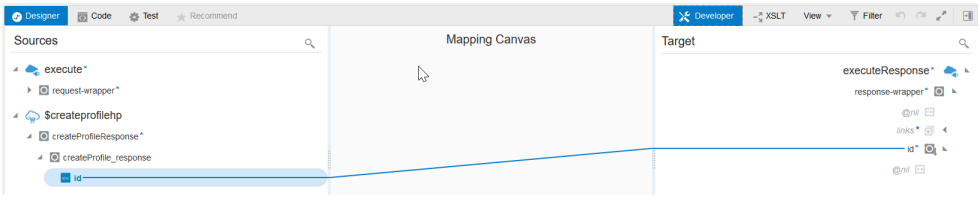
In user-friendly mode, adapter names are displayed along with the adapter's associated icon and the type of payload (request/response). For this example, the source REST Adapter and Oracle Commerce Cloud Adapter and the target REST Adapter are displayed.



1. Click

   Developer

   to switch to technical names. The adapter names are removed.



Root Elements in Source and Target Trees

User-friendly names for the root elements of the different payloads enable you to easily correlate them with the associated invoke/trigger connection, the adapter used, and the type of payload (request/response). The icon of the root element corresponds to the associated adapter.

The format of user-friendly names for the root elements differs based on the variable type or the associated adapter. The following table lists the format of user-friendly names for the root elements for different variable types.

Adapter/Variable Type	Format of User-Friendly Name	Example
Application Adapter	*trigger/invoke_action_name payload_type (request/response) (Associated_Adapter_Name)*	SendInventoryAdjustments Request (SOAP)
System Adapter	See the Example column.	Schedule - Schedule $self (for technical mode) or Integration Metadata (for user-friendly mode)
Tracking Variables	If a user-friendly name is entered for the tracking variable in the Business Identifiers For Tracking page, that becomes the user-friendly name for the variable in the mapper. If the **Tracking Name** field in the Business Identifiers For Tracking page is not populated for the variable, the system constructs the user-friendly name for the tracking variable in the format of Tracking Variable 1/2/3.	My Business Identifier Tracking Variable 1 Tracking Variable 2 Tracking Variable 3
Other Variables	For all other variables (that is, simple variables and the root element of the complex variables), the user-friendly name is automatically constructed using the name with which the variable was created (without the $ prefix).	counter studentName

Child Elements in Source and Target Trees

The user-friendly names for the child elements in the source and target trees are derived from the associated schema files. If the schema files are generated with user-friendly names for the elements, the elements get rendered with those names in user-friendly mode in the mapper.

If the schema files do not contain user-friendly names for the elements defined, the child elements are displayed with the technical name in both user-friendly mode and technical mode.

Figure 1-1 Child Elements Shown with User-Friendly Names

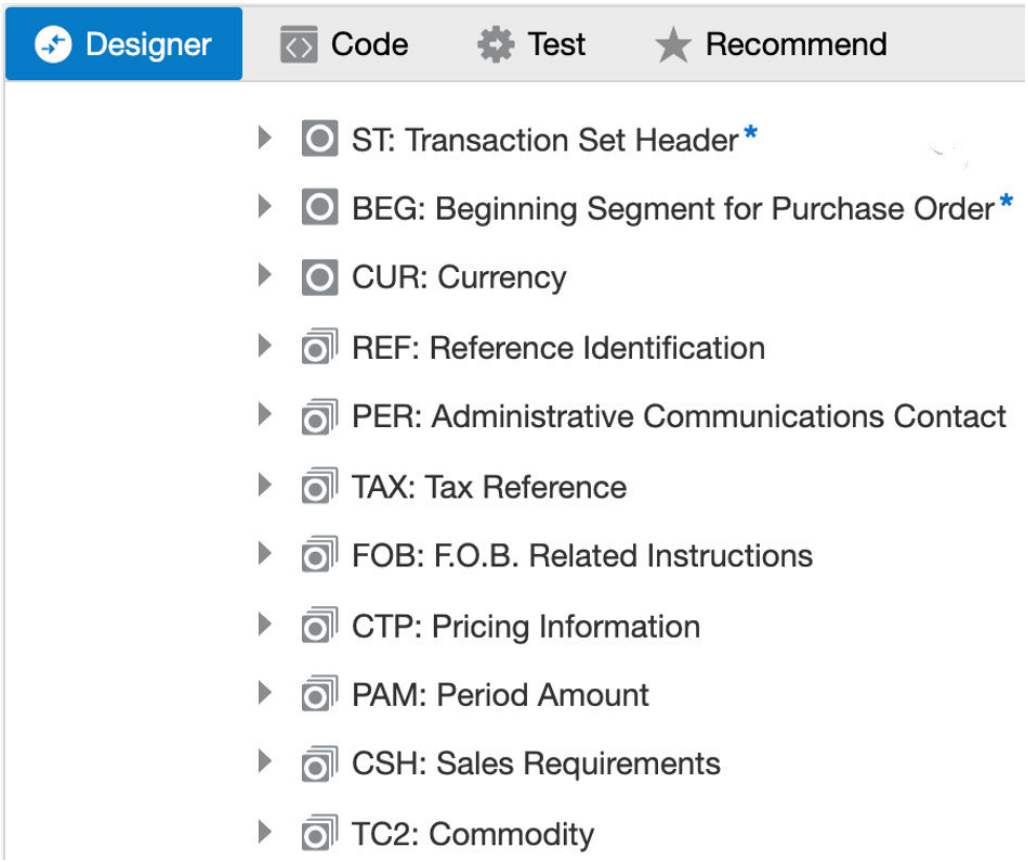
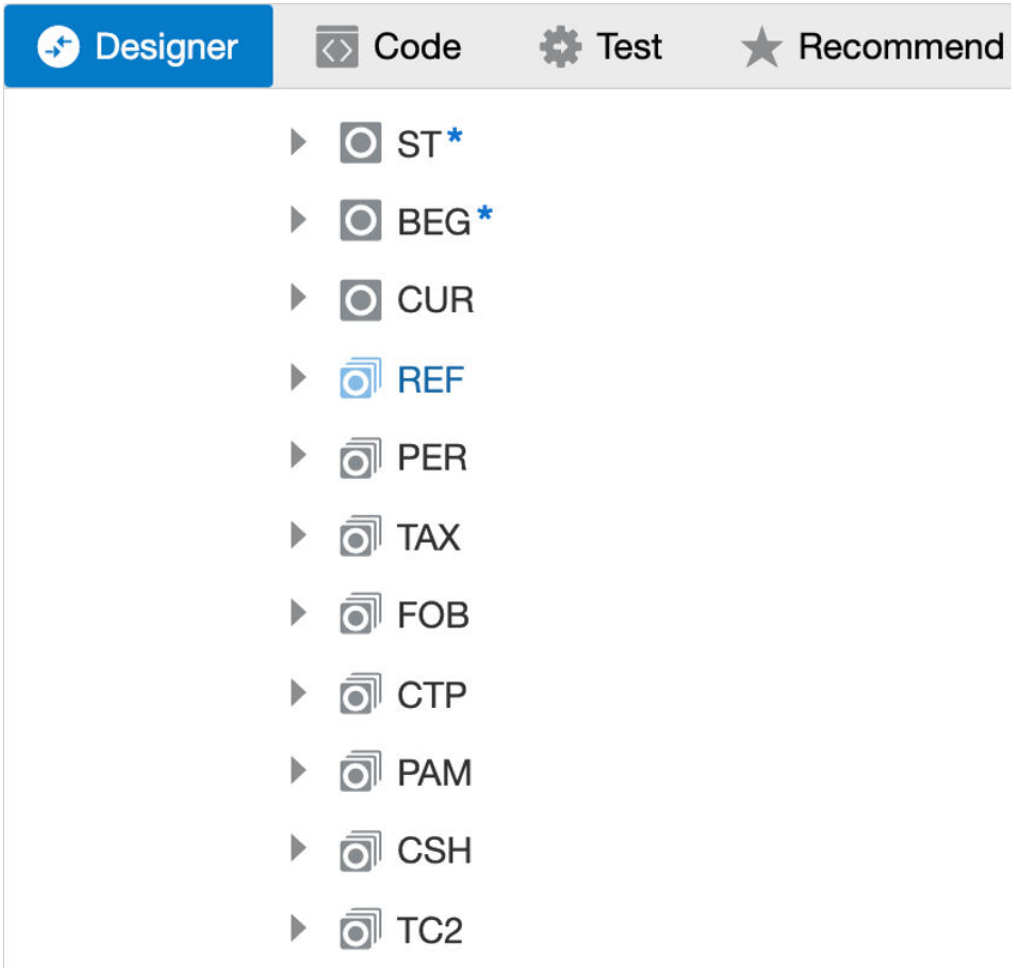


Figure 1-2 Child Elements Shown with Technical Names



The attributes of the schema elements are rendered with the `@` prefix followed by the attribute name in the mapper. With user-friendly names, the `@` prefix is not appended to the front of the name or in user-friendly mode. In technical mode, the attributes are shown appended with the `@` prefix.

User-friendly names do not include the namespace prefix. The option to view element names with the prefix **Show prefixes** available in the **View** menu of the mapper is disabled when the mapper is in user-friendly mode. The option is enabled once you switch to technical mode.

Search For Data in the Source and Target Trees

The source and target trees can be searched with the element name in either user-friendly mode or technical mode.

For example, assume the mapper is in user-friendly mode and an element exists whose user-friendly name is `BEG: Beginning Segment for Purchase Order` and technical name is `BegSegPO`. If `SegPO` is the search string used to search for the element, the search highlights the element irrespective of your current mode.

User-Friendly Expression for Mapping

Just as the source and target element technical names are simplified by their user-friendly names, the mapping expression created is represented in a simplified form.

This is a user interface-only entity. That is, the user friendly expression for a mapping is displayed in the mapper. However, it does not get saved in the XSL file. Click the **Code** tab of the mapper after creating a mapping. The **Code** tab shows the XSL file that is generated behind the scenes. Note that the file contains only the technical mapping, and not the user-friendly expression. The mappings work as they always have at runtime. At design time, the mapper displays the mappings as user-friendly expressions in user-friendly mode and as technical expressions in technical mode.

The user-friendly expression for a mapping is created when a mapping is constructed in the mapper. The user-friendly expression is created based on the user-friendly name for the components in the mapping.

Consider the following mapping:

```
concat($EDI-Translate/nsmpr0:executeResponse/ns31:TranslateOutput/
ns31:translation-status,
$EDI-Translate/nsmpr0:executeResponse/ns31:TranslateOutput/
ns31:tracking-info)
```

This mapping refers to a `concat` function whose parameters are two elements from the payload. The user-friendly expression for this mapping is as follows:

```
concat( translation-status, tracking-info)
```

where:

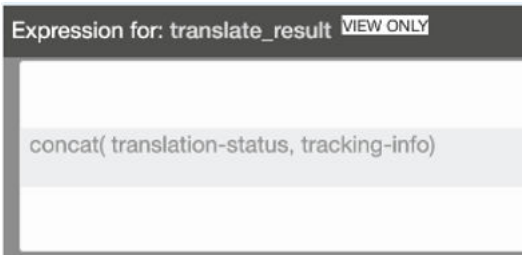
- `translation-status` is the user-friendly name of the element `$EDI-Translate/nsmpr0:executeResponse/ns31:TranslateOutput/ns31:translation-status`
- `tracking-info` is the user-friendly name of the element `$EDI-Translate/nsmpr0:executeResponse/ns31:TranslateOutput/ns31:tracking-info`

Expression Builder

When you navigate to the mapper, the Expression Builder launches in user-friendly mode by default when you select a target element.

The Expression Builder shows the mapping for the target element selected. As with the mapper, the Expression Builder also has two modes. User-friendly mode shows the mapping as a user-friendly expression.

Figure 1-3 User-Friendly Names in Expression Builder



To toggle the Expression Builder between the two modes, click the toggle button available on the right side of the Expression Builder. You can manually edit the existing mapping in the Expression Builder.

Figure 1-4 Technical Names in the Expression Builder

Other Sections of the User Interface

Other sections of the mapper in which the source and target elements are displayed all show the names in synchronization with the mode that is selected for the mapper (user-friendly or technical). For example:

- The **Test** button (where the root elements of each source are displayed as the headers of the tabs)
- The **Filter** button (where one of the options to filter the tree data is by source name, which shows the root elements of the different sources)

This means that if the mapper is in user-friendly mode, these sections of the user interface also show the user-friendly names of the elements. If the mapper is in technical mode, these sections show the technical names of the elements.

About the Expression Builder

Use the Expression Builder to view and edit your XPath expressions. This section provides an overview of the Expression Builder.

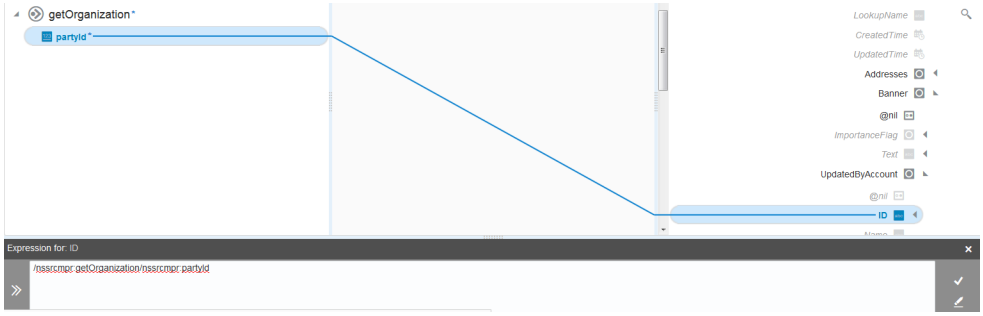
Displaying the Expression Builder

1. Click a target element node.

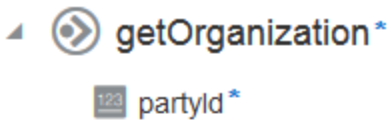
   The Expression Builder is displayed. A shuttle button ( ) is displayed on the left side of the field. Save ( ) and erase ( ) buttons are displayed on the right side of the field.

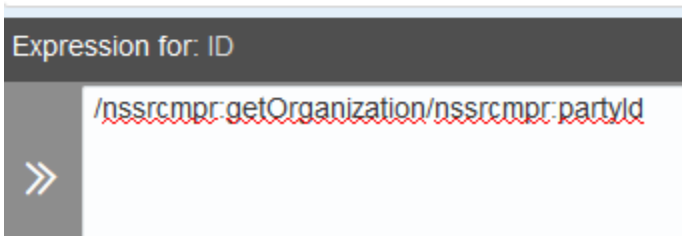
2. Drag a source element node to a target element node.

   The XPath expression is added to the Expression Builder.



3. If you want to remove the value, click  , then click  to completely remove the mapping.
4. Drag the source element node to the Expression Builder. You can also highlight the source element node and click  to add a value to the Expression Builder.





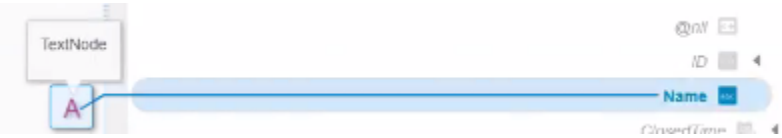
5. Click  to save the mapping.

Using Set Text Mode

When there is no mapping in the Expression Builder, there is an  button. This option enables you to enter text in an element node. You can only have XPath expression or text in the Expression Builder. You cannot have both types.

1. Highlight a target element node and click  in the Expression Builder to enter set text mode.
2. Enter text in the Expression Builder.

   A letter icon is added to the node. When you place your cursor over the icon, the text you entered is displayed.



> **Note:**
>
> If you drag a source target node into the Expression Builder while in set text mode, the mapping value is literally added as text, and not as an XPath expression.

Entering Literal Values

You can enter literal values in the Expression Builder when you are *not* in set text mode.

1. Enter text in the Expression Builder.

   This creates a `value-of` expression in the XSTL file instead of straight text. See View the XSLT Code.

Access the Mapper

To create mappings in an integration, you need to first access the mapper. The method for accessing the mapper is based on the integration pattern you are using.

To create mappings in App Driven Orchestration integrations and Scheduled Orchestration integrations:
As you add triggers and invokes to an App Driven Orchestration integrations, a map icon is automatically added. You can also add ad-hoc mappings to this type of integration, such as adding a mapper to a switch action.

1. Click an existing mapper icon or drag a mapper into your integration from the **Actions** panel to the appropriate location in your integration.
2. Click **Edit**.

If you click the **View** icon, note the following details:

- You cannot add or edit mappings.
- You cannot validate mappings.
- You cannot save or erase the XPath expression in the Expression Builder.
- You cannot create or delete elements or mappings in the target context menus.
- You cannot drag source element nodes to target element nodes.
- You can view XSLT code and test your mappings.

3. See Creating Integrations.

To create mappings in Basic Routing integrations:

1. In the middle of the integration, click the **Mapper** icon for the request, response, or fault map to edit.
2. Click **Edit**.

See Creating Integrations.

2

Map Data

Use the mapper to drag element nodes in the source structure to element nodes in the target structure.

Topics

Accept Mapping Recommendations with the Recommendations Engine

You can accept the target element node recommendations of the recommendations engine when creating mappings. This eliminates the need to analyze and perform each individual source-to-target mapping. The findings of the recommendations engine are particularly useful when you have a new integration in which mapping has not yet been created. You can also use the recommendations engine with previously-created mappings.

Topics

Disable and Enable the Oracle Recommendations Engine

By default, the recommendations engine is enabled. When enabled, all integrations on this instance are published to the recommendations engine. If you want, you can disable this feature.

To disable or re-enable the recommendations engine:

1. In the left navigation pane, click **Home** > **Settings** > **Recommendations**.

2. Deselect the **Contribute integration mappings to Oracle Recommends.** check box, then click **Save** in the upper right corner.
3. To re-enable, select the **Contribute integration mappings to Oracle Recommends.** check box, then click **Save** in the upper right corner.

Accept Target Element Mapping Recommendations

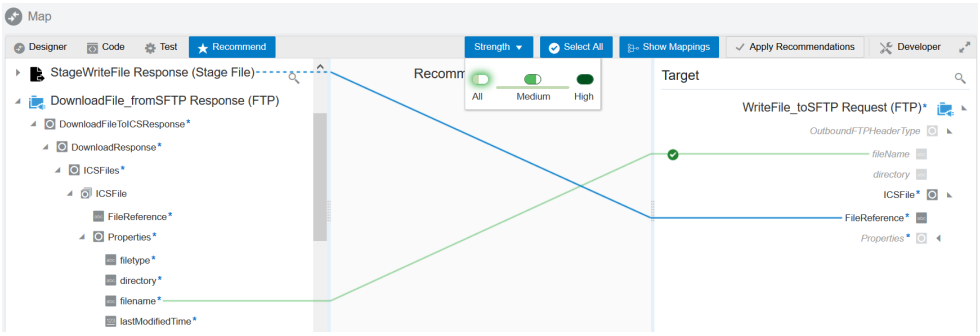
The mapper includes a recommendations engine for creating mappings. This eliminates the need to analyze and perform each individual source-to-target mapping. The findings of the recommendations engine are particularly useful when you have a new integration in which mapping has not yet been created. You can also use the recommendations engine with previously-created mappings.

> **Note:**
>
> Ensure that you first enable the recommendations engine. See Disable and Enable the Oracle Recommendations Engine.

To use the mapping recommendations of the recommendations engine:

1. Go to the Integrations page, and find the integration in which to use the mapping recommendations of the recommendations engine.
2. Open the mapper.
3. Click **Recommend** in the upper left corner.

   The page shows the recommended source and target element nodes mappings.



4. Click **Strength**. The strength of each recommended mapping is displayed at the top.
5. If you want to accept the all recommendations, click **Select All**.
6. To deselect a mapping, click the right mark or click the mapping line and click **Select**.
7. Perform one of the following steps:
   a. Click **Designer** to exit the Recommendations page and return to the mapper.
   b. Click **Apply Recommendations** to apply the selected mappings. The recommendations you selected are displayed in the mapper. Click **Validate** to save the changes.

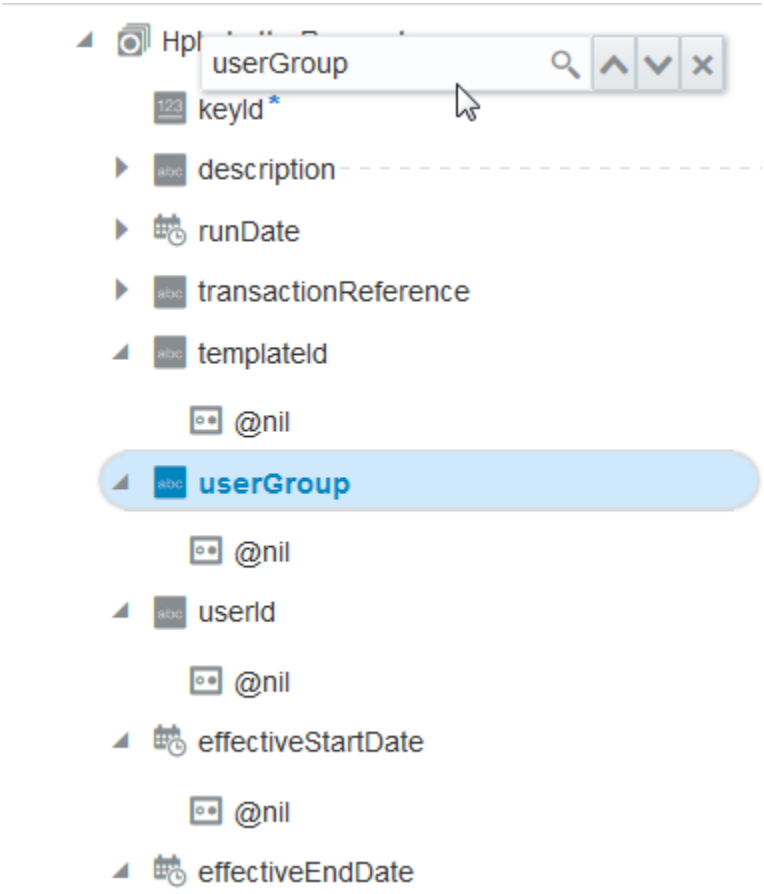
c. Click **Close** and select **Apply** to save the mapping and exit the page.

Search Data Fields

The mapper displays the source data structure on the left and the target data structure on the right. You can search for specific element nodes or attributes (identified by the @ prefix) in either the source or target structure.

To search data fields:

1. In the **Sources** or **Target** section, click  .
2. Enter the full or partial name, and click  .



The tree is automatically expanded and scrolls to the first match. If you entered straight text (for example, `country`), any element nodes and attributes of the same name are found. If you search by attribute (for example, `@country`), only the attributes of the same name are displayed.

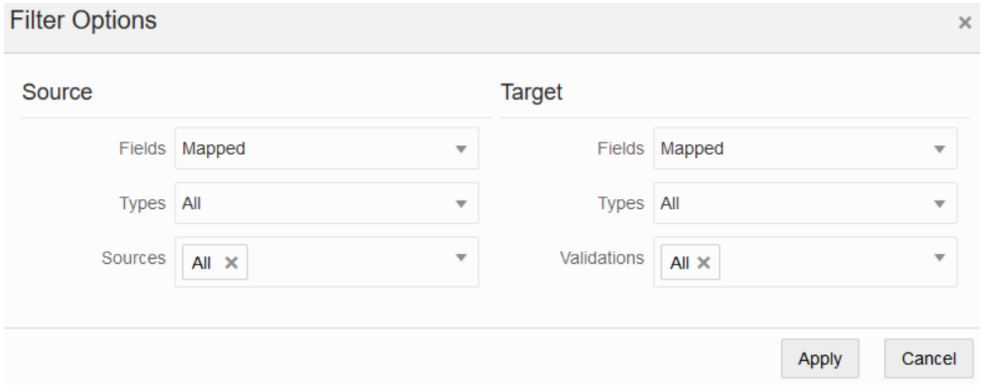
3. Click the **V** icon to scroll to the next match.
4. When done, click the **X** icon to dismiss the search facility.

Filter the Source or Target Data Structures

You can filter the display of the source and target structures. This enables you to show only the details in which you are interested.

To filter the source or target data structures:

1. Click Filter in the **Target** section of the mapping toolbar.
2. Specify map filtering options based on the following criteria.
   - View the mapped element nodes, unmapped element nodes, or both.
   - View all element node types (required element nodes and custom element nodes you created in a prebuilt Oracle integration that was edited in customization mode).
   - View the source data structures in the integration (main source and secondary sources).
   - View validation details (view only errors, only warnings, or only mappings with no issues).



3. Click **Apply**.

   Based on your selections, icons are displayed in the mapper toolbar. For example, Mapped is displayed for both data structures if you selected to show mapped element nodes in both the **Sources** and **Target** sections.

4. To remove the selected filtering, click Mapped.

View the XSLT Code

You can view the XSLT code being created as you map source data structures to target data structures.

To view the XSLT code:

1. Click **Code**.

The current XSLT code for your mappings is displayed.

2. When complete, click **Designer** to return to the mapper.

Test Your Mappings

Once you complete designing your mappings, you can test them by entering sample content of the message to process in the mapping tester. When you execute the test, output is generated from the sample content.

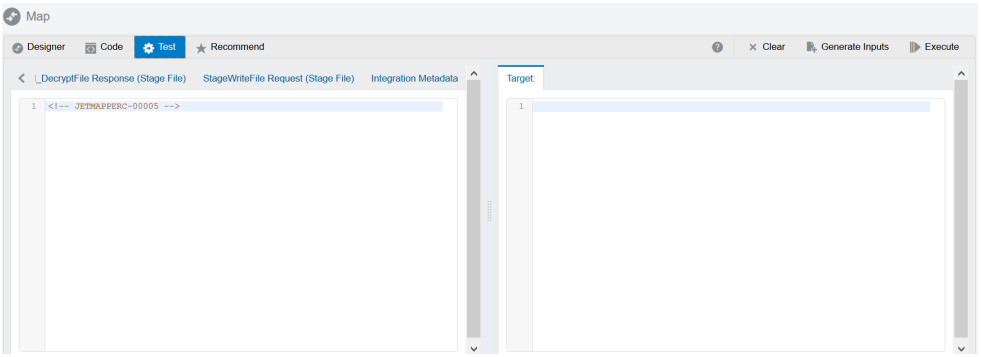
To test a mapping:

> **Note:**
>
> Custom JavaScript functions cannot be tested.

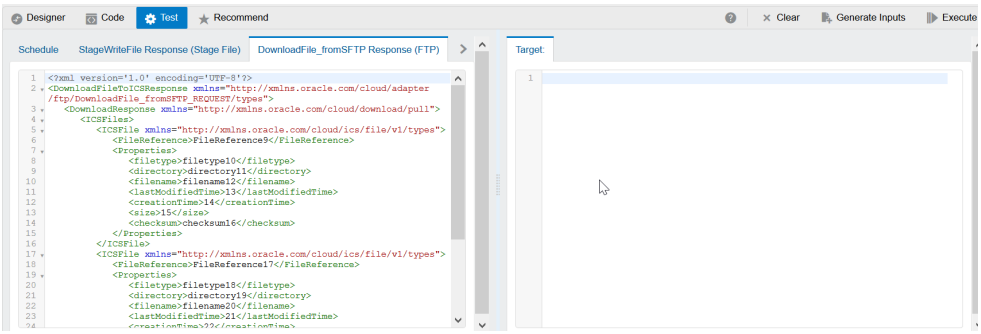
Two elements are generated when the element is defined as repeating. This is useful for generating payloads containing at least two repeating elements to test for-each loops in the mapper.

1. In the mapper toolbar, click **Test**.

   The mapping tester appears. The names of the source data structures for your mapping are displayed at the top. Two instances are also generated and displayed for repeating nodes.



2. In the **Input** panel, you can manually enter the payload, copy and paste the payload, or click **Generate Inputs** to automatically generate and test the payload. Payloads for scalar parameters are not created.

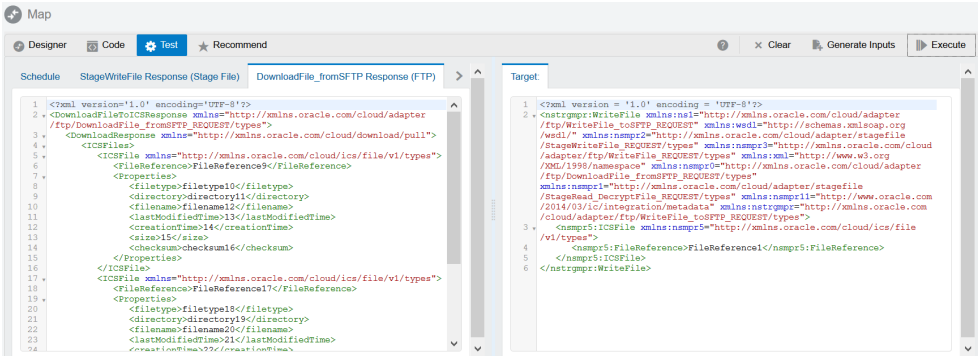


If your mapping includes multiple source data structures, both names are displayed. Payloads for both sources can be generated.

> **Note:**
>
> If the payload is very large, it is not automatically generated and you receive the following error message:
>
> ```
> Payload could not be generated for the
> ''$SourceApplicationObject'' schema due to excessive size
> and a lack of system memory
> ```

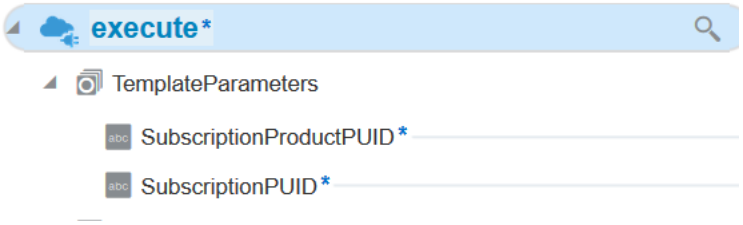
3. Scroll through the input payload and note the following details:
   - Unbounded, repeating elements are displayed multiple times.
   - Schemas of up to 20 levels in depth can be displayed.
   - Random values are automatically generated for payload elements. Based on the data type of the element, the correct values (for example, numerical or string values) are generated.
   - You can manually edit the randomly-generated values, as necessary.
4. Click **Execute** to generate results in the **Target** panel.

5. Review the results in the **Target** panel to ensure that your input payload was processed correctly.



6. Test your mapping and, as necessary, return to the mapper to make mapping changes, such as changing the XSLT statements or functions used.
7. To clear the **Input** and **Target** panels, click **Clear**.
8. When testing is complete, click **Designer** to return to the mapper.

   Multiple entries are generated for template parameters. There should be only be one instance of each template parameter. This is the expected behavior. For example, `/subscriptions/{SubscriptionPUID}/child/products/{SubscriptionProductPUID}` generates repeating elements for template parameters `SubscriptionPUID` and `SubscriptionProductPUID`.



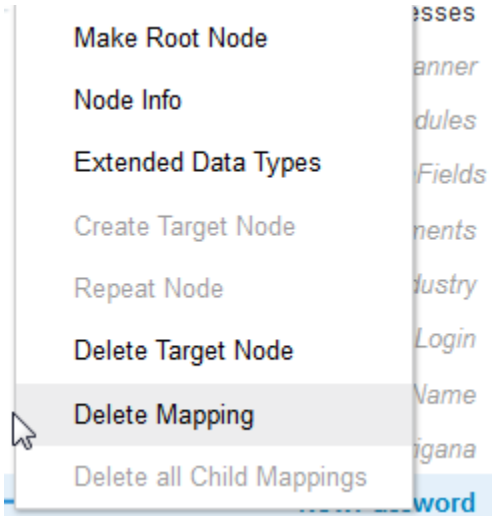
Delete Mappings and Target Element Nodes

You can delete mappings and target element nodes from a context menu. You can select this option for a parent to delete all children. For example, if you select the root, all mappings are deleted.

Deleting Mappings

To delete mappings:

1. Find the source-to-target mapping to delete.
2. Right-click the target element node name, and select **Delete Mapping**.



Deleting Target Nodes

To delete target nodes:

1. Find the source-to-target mapping.
2. Right-click the target element node name to delete, and select **Delete Target Node**.

   This action deletes the mapping *and* the target element. The element node is now grayed out (considered a ghost node). If you click **Code** and view the XSLT file of the mapping, note that this element does not exist. However, you can still map to it.
3. If you want to create this target element node, select **Create Target Node** to create it again in the XSLT file. As a short cut, you can also create a target element node by simply dragging a source element node to it.

> **Note:**
>
> - If you delete a parent element node, all of its child element nodes and any of their mappings are also deleted.
> - If you drag an XSLT statement to a target element node, the node must already exist (cannot be a ghost node). In those cases, you must first right-click the target element node and select **Create Target Node**.

Troubleshoot Errors

Your mappings can contain errors. These errors must be resolved before you can activate your integration. These errors may become visible when you click **Validate** during mapping design. Errors may also become visible when you complete

your mapping without errors, but make changes in the overall integration such as regenerating a WSDL. When you return to the mapper, these errors are visible.

Error messages are identified by red icons and warning messages are identified by yellow icons above the **Sources** section of the mapper.

To troubleshoot errors:

1. Expand the numbers in the red and yellow icons to show additional messages.

2. Click the message to access the error or warning in your mappings.

   For this example, there are two invalid target errors. The targets are in the XSLT file, but not in the schema. This may have occurred because the WSDL was regenerated after you previously completed mapping.

   When adding functions to your mappings, you can also receive errors if you do not enter all the parameters in the Expression Builder. For example, you add a **concat** function to your mapping, but forget to add one or both parameters to the function.

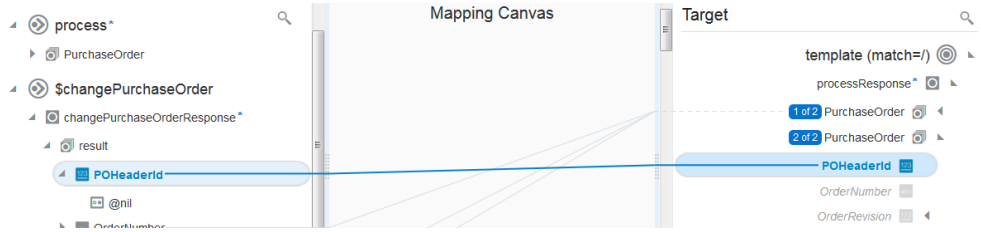
3. To show only the mappings with errors and warnings, click Filter and select **Errors** and **Warnings**. See Filter the Source or Target Data Structures.

Repeat a Target Element to Map to Different Sources

You can repeat a target element in the mapper. This enables you to map different sources to the same target element. Elements defined in the target schema with the `maxOccurs` attribute set to a value greater than one can be repeated.

To repeat a target element to map to different sources:

1. In the target data structure, right-click the element node to repeat, and select **Repeat Node**. This option is only available on elements that you can repeat. Elements that can be repeated are identified by a special icon with two bars to the left of the name. When you place your cursor over these elements, the words **Repeating: true** are displayed in the information text.



The element is repeated and displayed below the existing element. Elements that are repeated show the count (for example, **1 of 2** for the existing element and **2 of 2** for the repeated element. You can repeat an element multiple times.

2. Expand the existing and repeated elements to see that the attributes in each element are repeated.

3. Drag appropriate source mappings to the repeated targets.

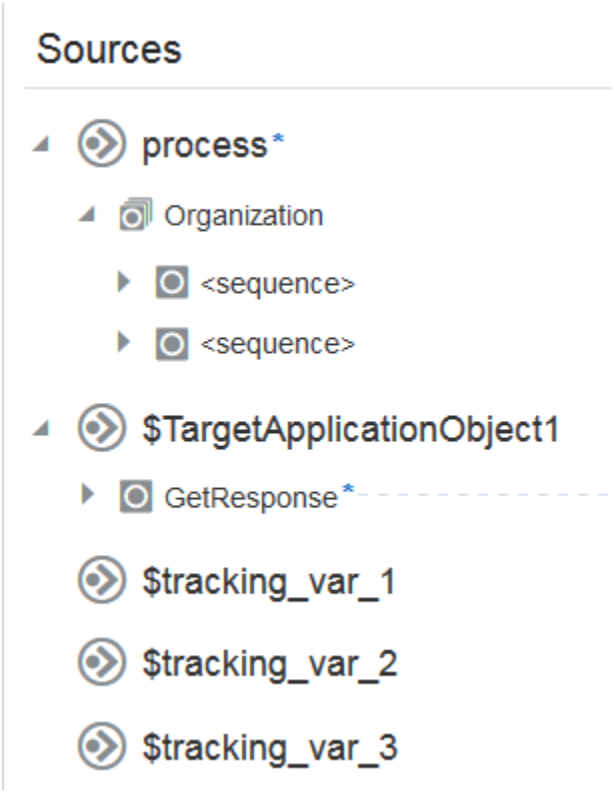
> **Note:**
>
> If you create a repeatable element in which you do not do any mapping, click **Close**, and apply your changes when prompted, the empty element is not saved.

Map Multiple Source Structures to a Target Structure

You can map fields from multiple source structures to a single target structure in certain parts of integrations (for example, integrations in which message enrichment points have been added or integrations with a response mapping). This action applies to the creation of new maps.

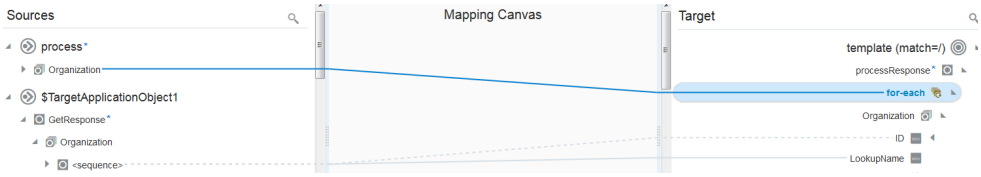
To map multiple source structures to a target structure:

1. In the mapper, note that two source structures are displayed:
   - The initial request mapping source (for this example, **process**)
   - The secondary request (for this example, **$TargetApplicationObject1**)



2. Expand the initial source data structure and drag appropriate source element nodes to target element nodes.

3. Expand the secondary source data structure and drag appropriate source element nodes to target element nodes.



4. To test the mappings, see Test Your Mappings.

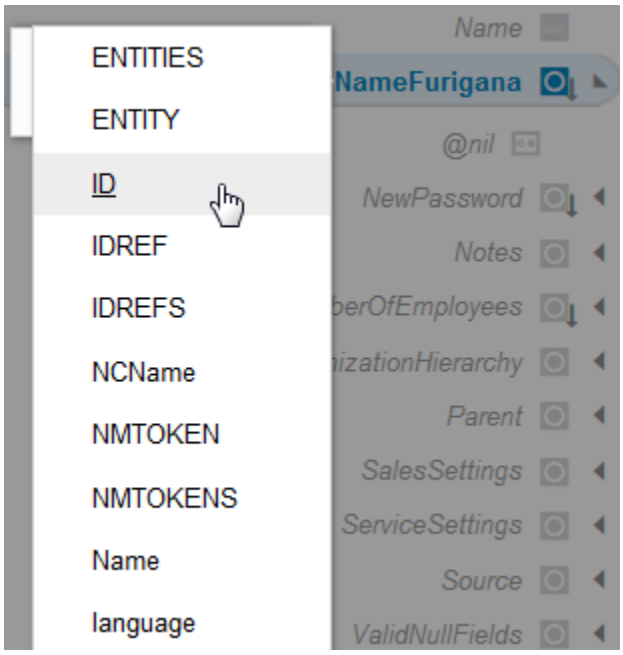
5. When complete, click **Close**, then apply your changes when prompted.

Extend a Data Type

You can extend a data type in the mapper. An extended data type is a primitive data type or container with a supplementary name and some additional properties. Extended data types are user-defined types based on the primitive data types boolean, integer, real, string, and date, and the composite type container.

1. Right click a target element and select **Extended Data Types**.

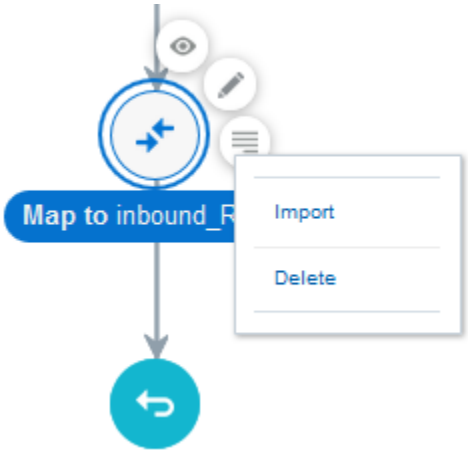
2. From the **Ext Datatypes** list, select the data type to extend.



Import a Map File into an Orchestrated Integration

You can import an XSL map file that was previously exported from the *same* integration. This action overwrites the existing mapping file. Once imported, the map file cannot be edited. For example, you can export the map from a specific integration, edit the XSL file as per a user requirement, save it, and import it back into the same integration. You cannot import an XSL map file into an orchestrated integration that was exported from a different integration in Oracle Integration Cloud or from an application in Oracle JDeveloper.

1. Right-click the map in which you want to import an integration, and select **More Actions** > **Import**.



2. Browse for the map file to import, then click **Import**. You only import the map file of an exported integration into Oracle Integration Cloud. You do *not* import the entire integration in which the map file is included into Oracle Integration Cloud.

3

Work with Functions, Operators, and XSLT Statements

You can add functions, operators, and XSLT statements to your mappings.

Topics

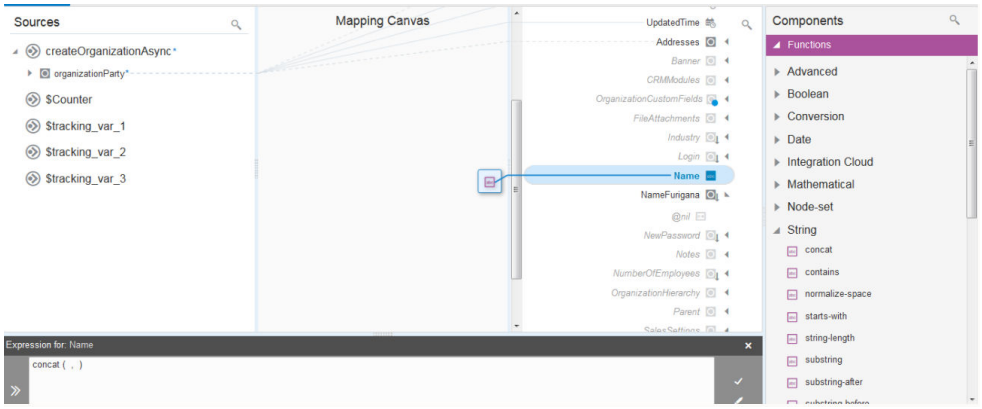
Add Functions, Operators, and XSLT Statements

You can add functions, operators, and XSLT statements to your mappings.

Working with Functions

1. In the **Target** section, highlight the element node to which to connect.
2. In the upper right corner, click  to launch the **Components** panel.
3. Expand **Functions**.
4. Select a function. For this example, **String** is expanded and **concat** is dragged to the target element node. The element can be an existing or ghost (not yet created) element.

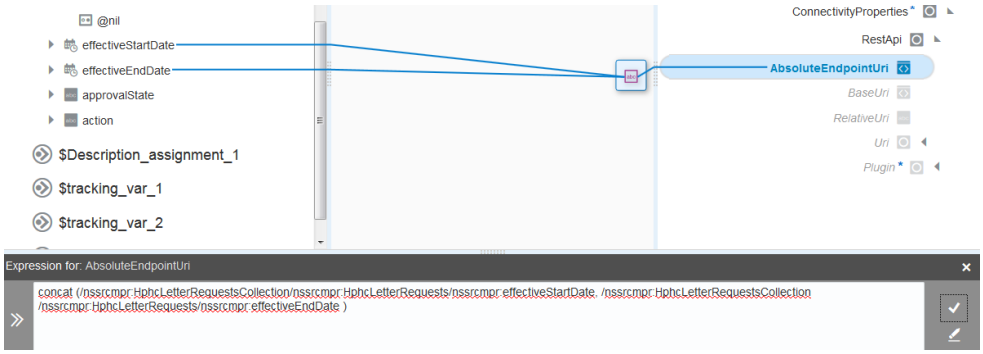
   A **function** icon is added to the **Mapping Canvas** section for the target element node and the function XPath expression is added to the Expression Builder at the bottom of the page. This icon indicates that a function is used in this mapping.



> **Note:**
>
> You can also initially drag functions to the Expression Builder and then connect the source element(s) to the function.

5. In the **Sources** section, drag the source element nodes to the function in the Expression Builder. For this example, **effectiveStartDate** and **effectiveEndDate** are dragged to the two sides of the comma in the **concat(,)** function in the Expression Builder. Do *not* drag source element nodes to the **function** icon in the **Mapping Canvas** section.

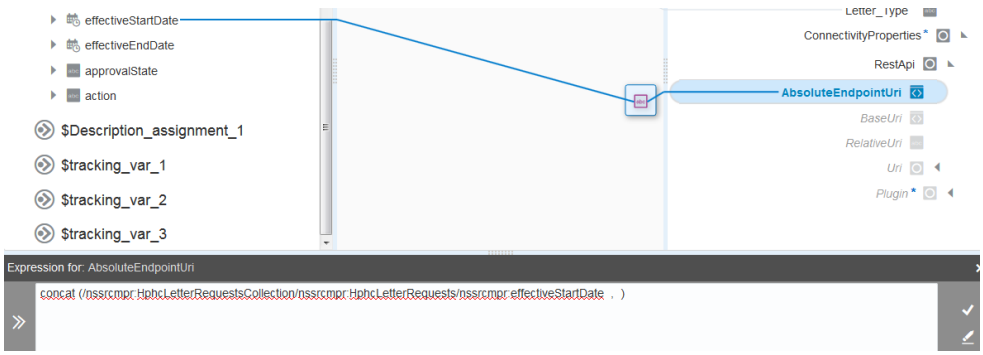
6. Click  to save your updates.



You can also drag functions onto existing mappings. Assume you have the following simple mapping.

7. Drag a function (for this example, **concat**) onto the line in the **Mapping Canvas** that connects the two elements.

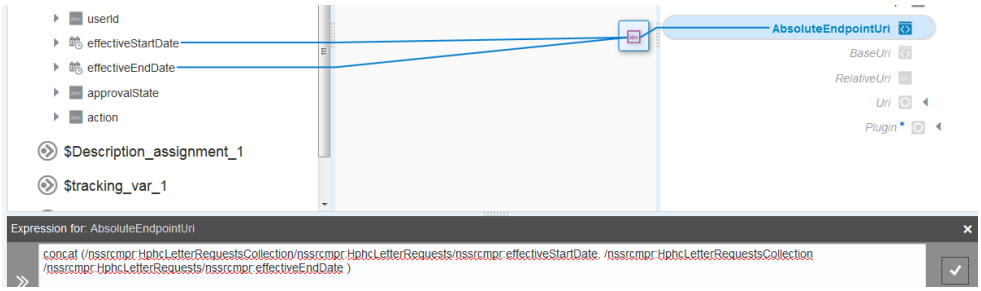
   This action adds the function to the line and shows the **concat** function in the Expression Builder. The existing source element mapping is added to the left side of the comma.



8. Drag the second source element to the right of the comma.

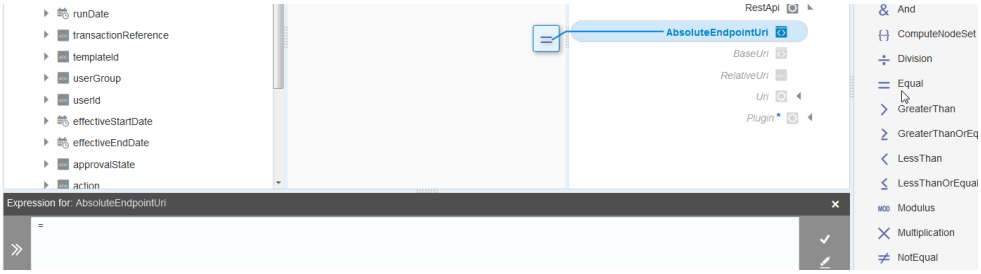
9. Click  to save the function.

   The **concat** function is shown as complete.



Working with Operators

1. Expand the **Operators** section.
2. Drag an operator to the target element node (for this example, a **=** is added). The **=** operator is also added to the Expression Builder. The element node can be a created or ghost element node.



3. Drag appropriate source elements to both sides of the operator or manually enter values.
4. Click  to save the operation.

   The operator icon is displayed in the **Mapping Canvas**.

Working with XSLT Statements

1. Click **Advanced**.

   An **XSLT** header is added to the **Components** panel.

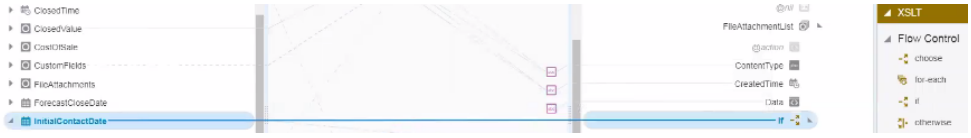
2. Expand **XSLT**.

3. Browse for and drag the appropriate XSLT statement onto the target element node or use the search facility to manually enter and search for the XSLT statement.

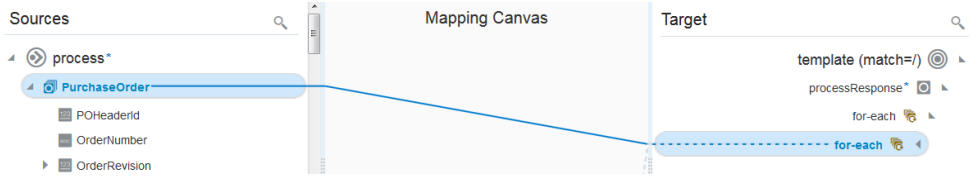
   Note the following conventions:

   - You can drag statements onto parent or child elements. Note the following conventions about dragging XSLT statements:
     - A green icon is displayed when you drag the XSLT statement to the front or the back of the element.
     - If a green icon is not displayed, you cannot insert as a parent.
     - Drag the statement to the end of the name to insert it as a parent.
     - Drag the statement to the front of the name to insert it as a child.
   - You can only drag XSLT statements onto created elements. If the element on which you want to drag the statement is grayed out (is a ghost node), right-click the element and select **Create Target Node**.

   For example, drag an **if** statement to the target element, then map a source element to the target element.



   Or drag a **for-each** statement to a repeatable element.



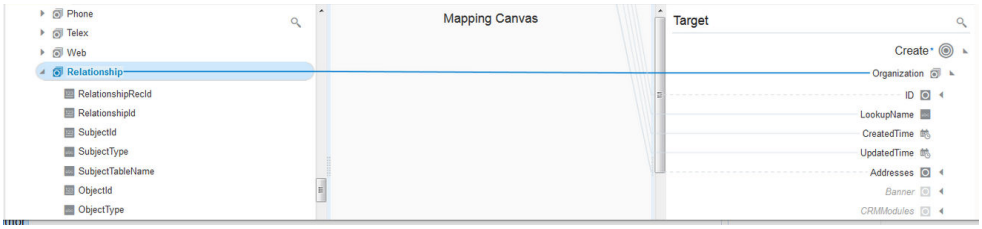
Automatically Create for-each Statements

You can automatically create for-each statements when mapping between repeatable source and target elements in the mapper.

To automatically create for-each statements:

1. In the **Source** section, identify the repeatable source and target elements to which to map. Repeatable elements are identified by the  icon to the left of the name. When you right-click these elements and select **Node Info**, **Repeating: true** is displayed in the message details about the element.

2. In the **Source** section, map the child repeatable element to the child target repeatable element. You cannot map repeatable elements to nonrepeatable elements.



The mapper creates a for-each statement to loop through the source **Relationship** element and place the mapping into the target **Organization** element. This statement does not include a value to select because parent elements do not typically contain attributes to map.

3. Click **Code** to view the for-each statement.

```
<xsl:for-each select="/nssrcmpr:createOrganizationAsync/
nssrcmpr:organizationParty/nsmpr5:Relationship">
              <nstrgmpr:Organization xml:id="id_27">
                    <rnb_v1_2:ID
xml:id="id_28" id="{/nssrcmpr:createOrganizationAsync/
nssrcmpr:organizationParty/nsmpr5:PartyId}"/>
                    <rnb_v1_2:LookupName xml:id="id_30">
                          <xsl:value-of
xml:id="id_31" select="/nssrcmpr:createOrganizationAsync/
nssrcmpr:organizationParty/nsmpr5:PartyName"/>
                    </rnb_v1_2:LookupName>
                    <rnb_v1_2:CreatedTime xml:id="id_34">
                          <xsl:value-of
xml:id="id_35" select="/nssrcmpr:createOrganizationAsync/
nssrcmpr:organizationParty/nsmpr5:CreationDate"/>
                    </rnb_v1_2:CreatedTime>
                    <rnb_v1_2:UpdatedTime xml:id="id_32">
                          <xsl:value-of
xml:id="id_36" select="/nssrcmpr:createOrganizationAsync/
nssrcmpr:organizationParty/nsmpr5:LastUpdateDate"/>
                    </rnb_v1_2:UpdatedTime>
                    <rno_v1_2:Addresses xml:id="id_37">
                          <rno_v1_2:TypedAddressList
xml:id="id_38">
                                <rno_v1_2:Country xml:id="id_41">
                                      <rnb_v1_2:ID
xml:id="id_42" id="{/nssrcmpr:createOrganizationAsync/
nssrcmpr:organizationParty/nsmpr5:Country}"/>
                                </rno_v1_2:Country>
                                <rno_v1_2:Street xml:id="id_39">
                                      <xsl:value-of
xml:id="id_40" select="/nssrcmpr:createOrganizationAsync/
nssrcmpr:organizationParty/nsmpr5:Address1"/>
                                </rno_v1_2:Street>
                          </rno_v1_2:TypedAddressList>
                    </rno_v1_2:Addresses>
```

```
                        <rno_v1_2:NameFurigana>
                          <xsl:value-
of select="/nssrcmpr:createOrganizationAsync/
nssrcmpr:organizationParty/nsmpr5:HQBranchIndicator"/>
                    </rno_v1_2:NameFurigana>
              </nstrgmpr:Organization>
</xsl:for-each>
```

Create Conditional Mappings

The if and choose statements are two ways to create conditions. If statements allow you to specify a single condition. Choose/when/otherwise statements allow you to specify multiple conditions, similar to if/then/else.

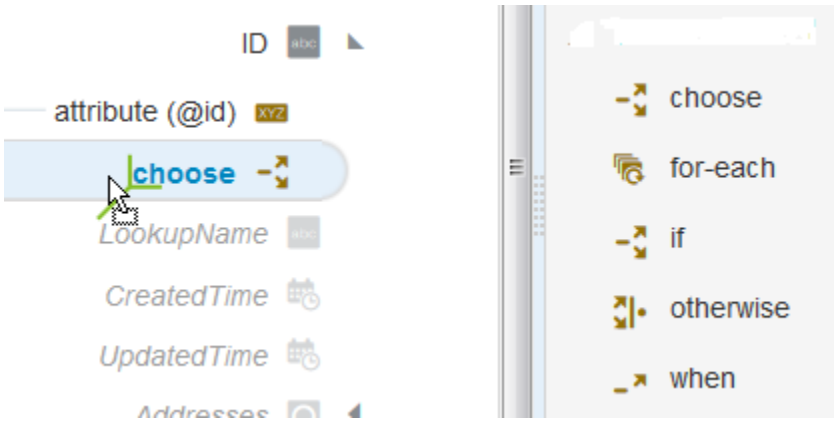
To create conditional mapping:

1. Drag a source to a target to create a mapping.
2. Click View, then select **Advanced**.
3. In the upper right corner, click  to launch the **Components** panel.
4. Expand **XSLT**, and drag appropriate XSLT statements onto the target element.

   You can either search or browse for the function.
5. Drag the **if** or **choose** function onto the target element. (for this example, an **if** statement is dragged to an ID element).

The **if** statement is displayed.

6. If you add a **choose** statement, you may specify additional **when** and **otherwise** conditions.



- Highlight the **choose** action, then drag and drop a **when** or **otherwise** statement.

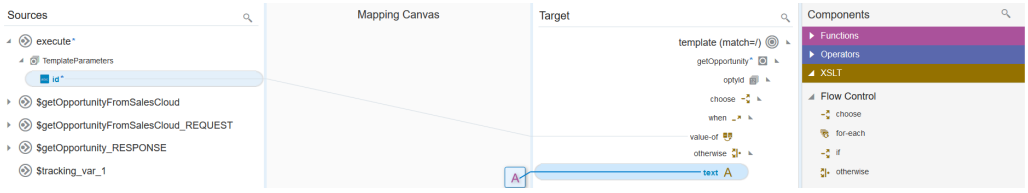
7. Click **Close**, then apply your changes when prompted.

See Use Conditional Mappings.

Set Default Values in the Mapper

You may have scenarios in which you need to set some fields to default values. The mapper contains a set of functions that you can use to set default values (for example, the **when** function that you can use to set default values).

For example, the following conditional mapping is performed.



In the payload, you can set the default value in the mapper.

```
<nstrgmpr:getOpportunity xml:id="id_12">
      <nstrgmpr:optyId>
            <xsl:choose>
                  <xsl:when test="">
                        <xsl:value-of select="/nssrcmpr:execute/nssrcmpr:TemplateParameters/nsmpr1:id"/>
                  </xsl:when>
                  <xsl:otherwise>
                        <xsl:text>1000</xsl:text>
                  </xsl:otherwise>
            </xsl:choose>
      </nstrgmpr:optyId>
</nstrgmpr:getOpportunity>
```

This syntax checks if the **iD** node is present in the payload. If so, it assigns that value. Otherwise, it adds the default value, which in this case is **1000**.

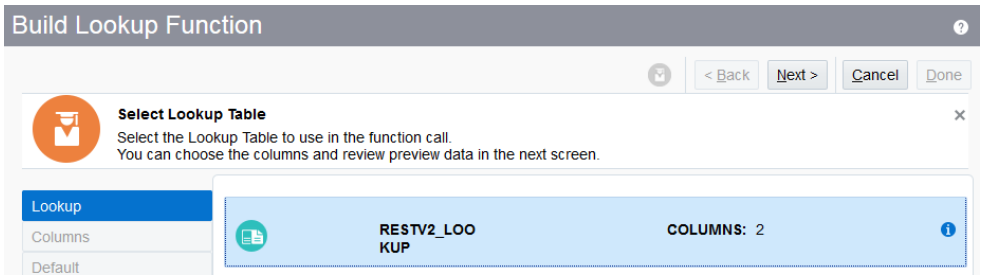
Reference Lookups

A special lookup function in the mapper enables you to call a lookup from a mapping to determine the value to populate into a field when transferring data between applications.

To reference a lookup from a data mapping:

1. Drag the source element node to a target element node to create a simple mapping.
2. In the upper right corner, click  to launch the **Components** panel.
3. Type `lookupValue` in the **Search** field, and click **Search**.
4. Drag the function onto the target element node.

   The mapper prompts you to select a previously created lookup.



5. Search by lookup name.

   When you select a lookup in the **Lookup Tables** column, you see preview data for that lookup on the right based on the connections you selected as the source and target.
6. Select a lookup and click **Use**.
7. The system automatically populates the following parameters in the lookup function:
   - **dvmLocation**: with the lookup name you selected
   - **srcColumn**: with the source application type
   - **srcValue**: with the existing mapping expression
   - **targetColumn**: with the target application type

8. Enter a **defaultValue** parameter. This is the value sent to the target if the lookup function is unable to find a match for the value passed from the source.
9. Click **Save** and **Close**.

> **Note:**
>
> Lookups referenced using the **lookupValue** function in the Expression Builder are included in an exported integration JAR file. When you import the integration, the referenced lookups are also imported and are visible in the Expression Builder. For information exporting integrations, see Exporting an Integration.

Create the lookupValue Function

You can create the parameter values for the **lookupValue** function with the Build Lookup Function wizard. This wizard enables you to define the lookup table, source column, target column, and default value to use in the function. For these parameter values to be selectable in the wizard, you must have already created a lookup on the Lookups page.

Topics

Access the Build Lookup Function Wizard

The Build Lookup Function wizard for creating the **lookupValue** function parameter values is accessible from the mapper.

To access the Build Lookup Function wizard:

> **Note:**
>
> You must already have created lookups to use this wizard. See Creating a Lookup of *Using Integrations in Oracle Integration*.

1. In the upper right corner, click  to launch the **Components** panel.
2. Expand **Functions** > **Integration Cloud**.
3. Drag the **dvm:lookupValue** function on the line in the **Mapping Canvas** section of an existing mapping.

> **Note:**
>
> If you drag the function to a ghost (not yet created) element, the element is first created.

The Build Lookup Function wizard is displayed. To create the function parameter values, see section Select the Lookup Table.

Select the Lookup Table

Select the lookup table to use in the `lookupValue` function.

> **Note:**
>
> You must already have created a lookup. Otherwise, no lookups are displayed for selection.

Element	Description
Lookup Table	Select the lookup table to use in the function. You can view the lookup description by clicking the information icon in the table. This can guide you in selecting the required lookup table. The number of columns defined in the lookup is also displayed.

Select the Source and Target Columns

Select the source and target columns to use in the `lookupValue` function.

The `lookupValue` function requires one source column and one target column. When you select a source and target column, the values available with the columns are displayed.

Element	Description
Select Source Column	Click the source column header to select from a list of available columns for this lookup table. The data included with the selected column is displayed. Both adapter and domain name columns are displayed.
Select Target Column	Click the target column header to select from a list of available columns for this lookup table. The data included with the selected column is displayed. Both adapter and domain name columns are displayed.

Specify the Default Value

Select the default value to use in the `lookupValue` function.

Enter the default value to use if no match is found. If there is no match that satisfies all the search values, the lookup fails and the default value is returned.

Element	Description
Default Value	Enter a default value to use if no match is found (for example, an actual default value to use or an error message such as `No Value Found`).

Review Your Lookup Table Selections

You can review the lookup table values to use in the `lookupValue` function on the Summary page.

You can review the lookup table values from the Summary page. The Summary page is the final wizard page after you have completed your configuration.

Element	Description
Parameter and Value Table	Displays a summary of the parameters and values you defined on previous pages of the wizard. To return to a previous page to update any values, click the appropriate tab in the left panel or click **Back**.

Element	Description

Resulting Expression	Displays the expression you defined on the previous pages of the wizard. The `lookupValue` function takes the following format:
	
	`lookupValue(dvmLocation, srcColumn, srcValue, targetColumn, defaultValue)`
	
	Where:
	• `dvmLocation`: The lookup table selected on the Select Lookup Table page.
	• `srcColumn`: The source column selected on the Select Columns page.
	• `srcValue`: The source value you enter in the **New Condition** field of the Expression Builder after completing this wizard. Click **Done** to complete this wizard, then define the `srcValue` parameter value.
	• `targetColumn`: The target column selected on the Select Columns page.
	• `defaultValue`: The default value entered on the Default Value page.
	For example, a defined `lookupValue` function after you have completed the wizard and defined the `srcValue` parameter value in the Expression Builder can look as follows:
	
	`dvm:lookupValue('tenant/resources/dvms/Country','rightnow','US','mysoap', 'No data found')`

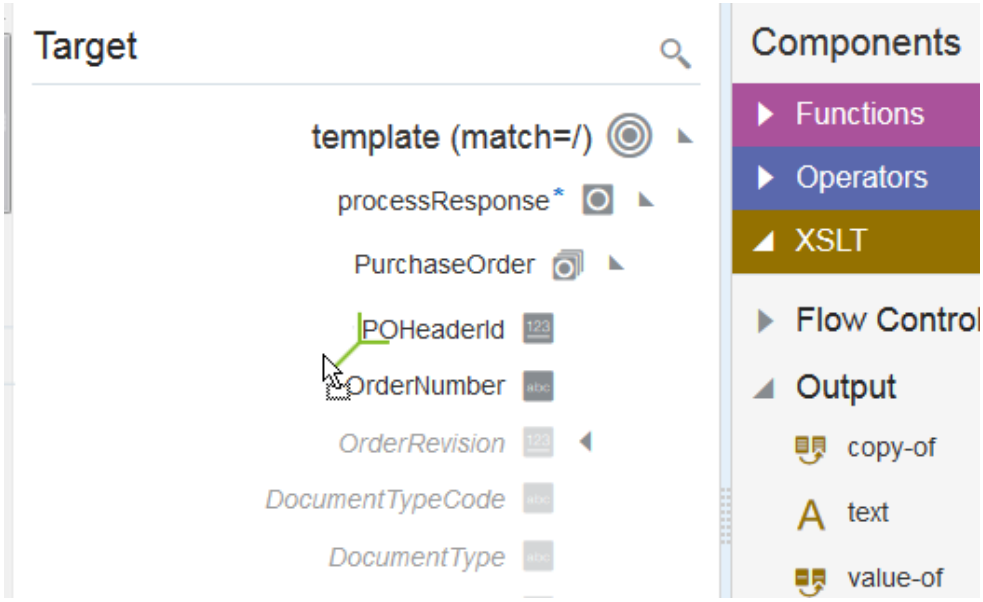
When you click **Done**, the **function** icon is created in the mapper and the function XPath expression is displayed in the Expression Builder.

Work with Multiple Value Statements

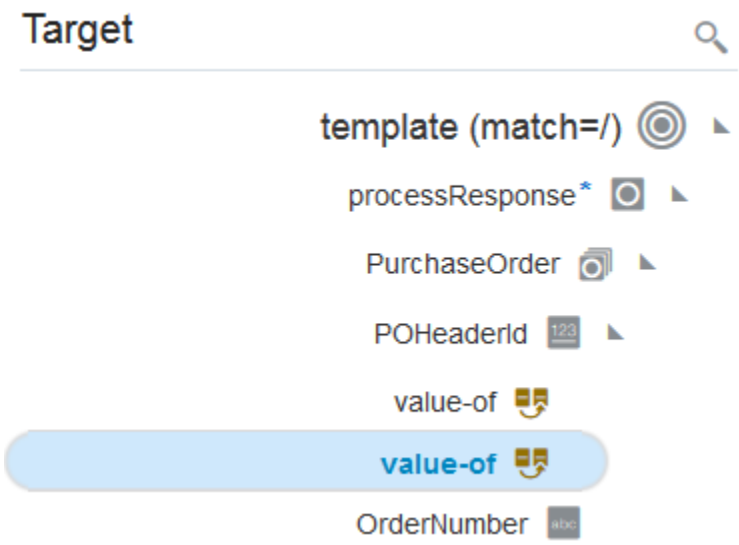
You can add multiple value-of statements or multiple XSLT statements under a leaf node.

To work with multiple value statements:

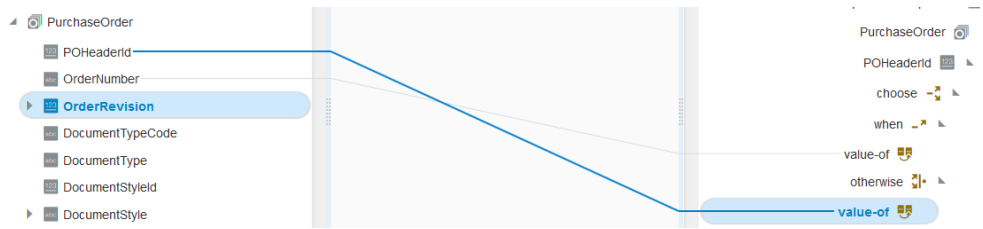
1. Click View and ensure that **Advanced** is selected.
2. Drag a **value-of** statement to a leaf element target in the mapper. For this example, **value-of** is added as a child of **POHeaderId**.



Multiple **value-of** statements are added to the leaf node.



3. Define appropriate mapping logic for each **value-of** statement. For example, add a **choose** statement and a **when** statement with a defined value to the first **value-of** statement and an **otherwise** statement to the second **value-of** statement.



> **Note:**
>
> Multiple **value-of** XSLT statements in a leaf node continue to remain visible in the mapper even if you disable **Advanced**.

4
Mapper Use Cases

Learn about use cases with the mapper.

Topics:

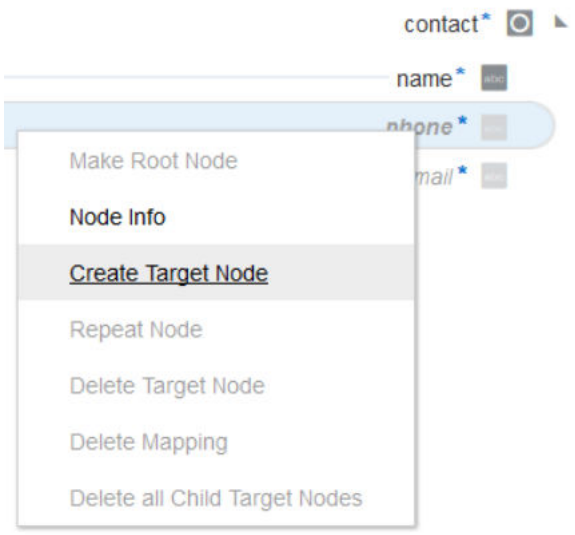
Use Conditional Mappings

You may have a requirement to map data dynamically depending on other data in your integration. This requirement can be achieved with conditional mappings.

Consider the following pseudo code sample of the mapping logic. Three conditions are provided.

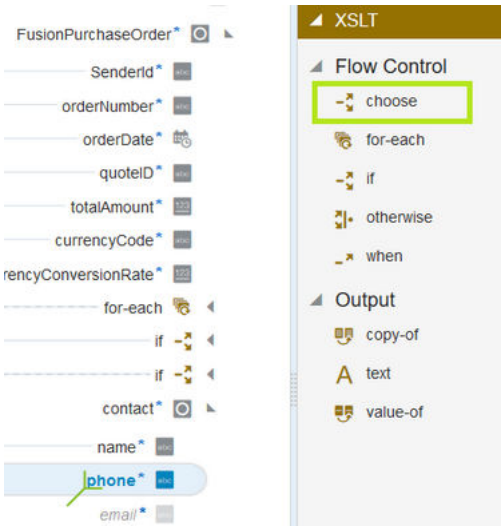
```
if PER03 == 'TE' {
    Contact.Phone = PER04
}
if PER05 == 'TE' {
    Contact.Phone = PER06
}
if PER07 == 'TE' {
    Contact.Phone = PER08
}
```

1. Click XSLT.
2. In the upper right corner, click  .
3. Expand **XSLT** to display the statements required to create conditional mappings.
4. Locate the target element (for this example, named **phone**) in the tree.

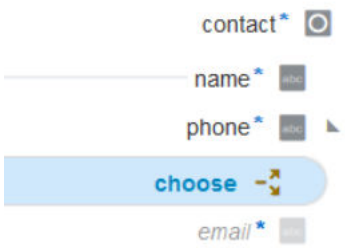


This is the element for which to create conditional mappings.

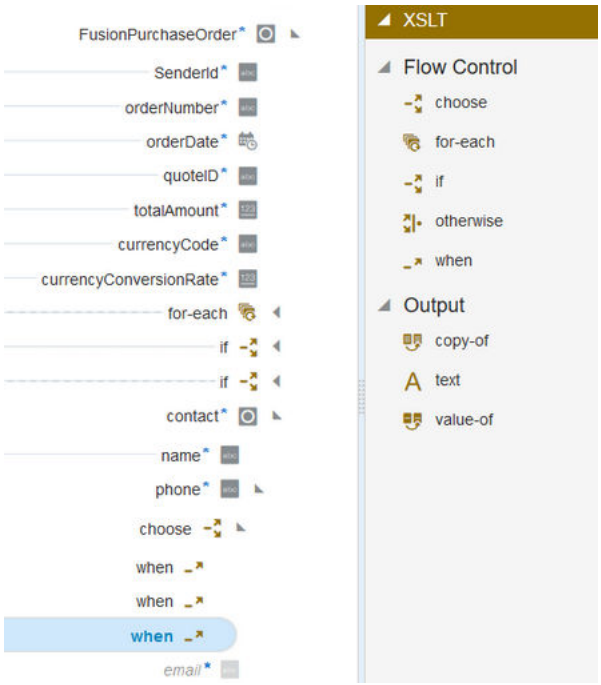
5. If the selected element is a lighter color and italicized, that means the element does not exist in the mapper's output. Right-click and select **Create Target Node**. You cannot insert conditions around **phone** without this step.
6. Drag and drop the **choose** statement as a child of **phone**.



The cursor position surrounding **phone** indicates whether the **choose** statement can be inserted as a child (bottom left) or a parent (upper right). For this case, **choose** is inserted as a child.

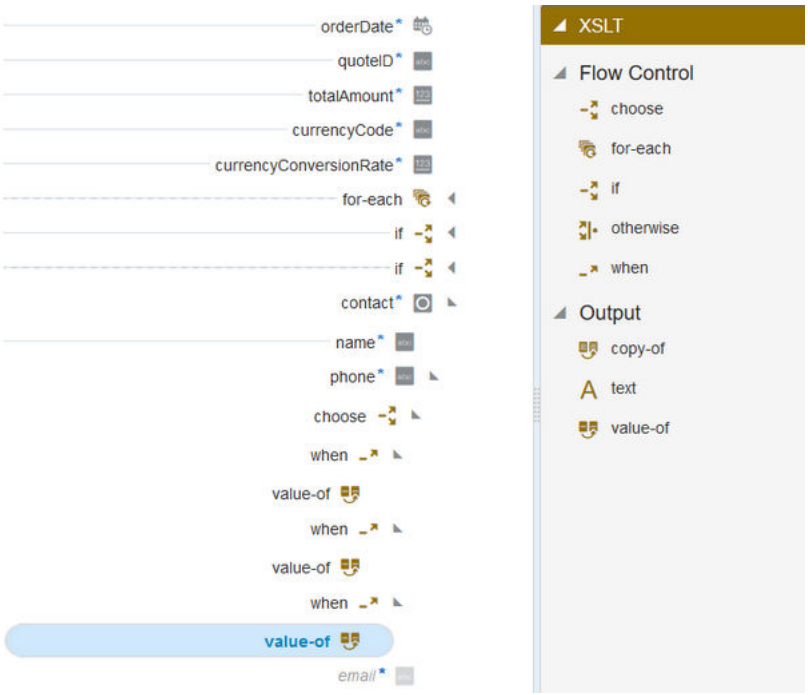


7. Drag and drop a **when** statement as a child of the **choose** statement three times to create placeholders for the three conditions. You can also drop a **when** statement as a sibling before or as a sibling after another **when** statement.



Each condition also needs a corresponding mapping value.

8. Drag and drop a **value-of** statement as a child of each **when** statement. The tree structure needed to create conditional expressions and mapping expressions is now complete.



Expressions for the first condition and mapping can now be created.

```
if PER03 == 'TE' { Contact.Phone = PER04 }
```

9. Select the first **when** statement in the target tree to create the first condition.
10. Drag and drop **PER03** from the source tree into the expression.
11. Enter **= "TE"** to complete the expression.

12. Click the **checkmark** to save the expression.
13. To create the mapping, select the **value-of** statement under the first **when**.
14. Drag and drop **PER04** into the target **value-of** statement.

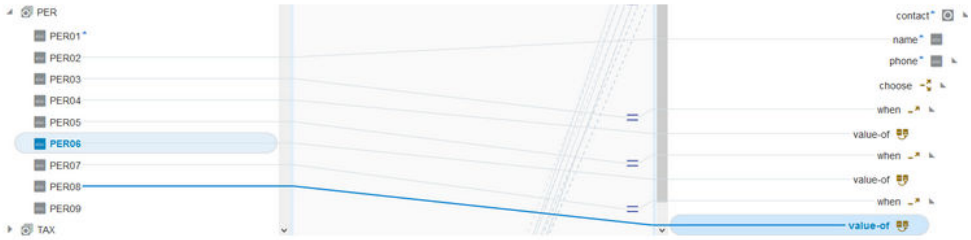
The first conditional mapping is complete.

15. Repeat these steps for the second and third conditional mappings to complete the mapping logic.

```
if PER05 == 'TE' {
    Contact.Phone = PER06
}
if PER07 == 'TE' {
    Contact.Phone = PER08
}
```

16. Save the mapping and integration.

    The completed design looks as follows.



Create an XSLT Map to Read Multiple Correlated Payloads

You can create XSLT maps to loop thought different sources (input payloads) with instances that are correlated by key fields.

Example for 1:0..*n* and 1:1 Relationships Between Sources

The following business units and employees example is provided:

- Each business unit can have 0..*n* employees (1:0..*n* relationship).
- The G/L accounts source with a 1:1 correlation with business units.

You can create an XSLT map that combines them.

The sources (input payloads) for this example are as follows:

- `$BusinessUnits`

```
<company>
  <bu>
    <id>SD</id> <name>Software Development</name>
    <accounbtid>i9</accountid>
  </bu>
  <bu>
    <id>BS</id> <name>Sales</name>
    <accounbtid>i1</accountid>
  </bu>
  <bu>
    <id>MD</id> <name>Marketing</name>
    <accounbtid>i2</accountid>
  </bu>
 </company>
```

- `$Employees`

```
<people>
  <emp> <buid>SD</buid> <name>Joe Smith</name> </emp>
  <emp> <buid>SD</buid> <name>Mike Jones</name> </emp>
  <emp> <buid>BS</buid> <name>Dave Johnson</name> </emp>
 </people>
```

- `$GLAccounts`

```
<gl>
   <account> <id>i1</id> <number>001.345</number> </account>
   <account> <id>i2</id> <number>001.477</number> </account>
   <account> <id>i9</id> <number>001.223</number> </account>
</gl>
```

The link between `$BusinessUnits` and `$Employees` is the business unit ID. The header is `$BusinessUnit` and the detail is `$Employees`. The link for the GL accounts and business units is the account ID.

The following output is needed:

```
<xxx>
  <yyy>
    <BU id='SD'>Software Development</BU>
    <empName>Joe Smith</empName>
    <accNumber>001.223</accNumber>
  </yyy>
  <yyy>
    <BU id='SD'>Software Development</BU>
    <empName>Mike Jones</empName>
    <accNumber>001.223</accNumber>
  </yyy>
  <yyy>
    <BU id='BS'>Sales</BU>
    <empName>Dave Johnson</empName>
    <accNumber>001.345</accNumber>
```

```
  </yyy>
</xxx>
```

Solution

When the instances (records) of the sources have a 1:1 correlation, you can use a predicate.

When the instances have 1:0..*n* correlation, using an `xsl:for-each-group` performs better than using predicates because it avoids overparsing the source.

The XSLT content is as follows:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<xsl:stylesheet version="2.0"  xmlns:xsd="http://www.w3.org/2001/
XMLSchema"
        xmlns:fn="http://www.w3.org/2005/xpath-functions"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:param name="BusinessUnits" />
  <xsl:param name="Employees" />
  <xsl:param name="GLAccounts"/>
  <xsl:template match="/" >
     <xxx>
        <xsl:for-each-group select="$Employees/people/employee" group-
by="buid">
           <!-- this section will be executed only once per 'buid' -->
           <!-- Store the Business Unit Record in a variable -->
           <xsl:variable name="BURecord">
               <xsl:copy-of select="$BusinessUnits/company/bu[id =
fn:current-grouping-key()]"/>
           </xsl:variable>
           <!-- Store the GL Account Record in a variable -->
           <xsl:variable name="GLAccountRecord">
               <xsl:copy-of select="$GLAccounts/gl/account[id =
$BURecord/bu/accountid]" />
           </xsl:variable>
           <!-- end: executed only once per 'buid' -->
           <xsl:for-each select="current-group()">
               <!-- iterates the employees within the current 'buid' -->
               <yyy>
                   <BU id="{./buid}">
                       <xsl:value-of select="$BURecord/bu/name" />
                   </BU>
                   <empName>
                       <xsl:value-of select="./name" />
                   </empName>
                   <accNumber>
                        <xsl:value-of select="$GLAccountRecord/account/
number"/>
                   </accNumber>
               </yyy>
            </xsl:for-each>
        </xsl:for-each-group>
     </xxx>
```

```
  </xsl:template>
</xsl:stylesheet>
```

Summary

- When there is a 1:1 relationship, using predicates instead of `<xsl:for-each-group>` is faster because XSLT does not need to sort the data to create the group.
- When there is a 1:0..*n* relationship, using `<xsl:for-each-group>` performs faster than using predicates. This is because predicates, in the above example, parse the entire business unit source and GL account source per every employee.

See:

- XPath predicates
- xsl:for-each example

5

Troubleshooting the Mapper

Review the following topics to learn about troubleshooting issues with the mapper.

Topics:

Current-dateTime Function Does Not Return the Same Number of Digits for All Timestamp Values

The `Current-dateTime function` in the mapper does not return the same number of digits for all timestamp values.

For example, the three digit microsecond value is not the same format each time.

```
YYYY-MM-DDT24:59:59.123Z
YYYY-MM-DDT24:59:59.12Z
```

If you want the specific format value to be consistent, use the `xp20:format-dateTime` function to format the timestamp. For example:

```
xp20:format-dateTime (fn:current-dateTime(), "[Y0001]-[M01]-[D01]T[H01]:
[m01]:[s01].[f001]" )
```

This function returns the following format:

```
2020-10-30T21:58:15.172Z
```

Import XSLT Code into the Mapper

For some functionality that is not available in the mapper, you can import XSLT code.

- To use a nested for-each loop in the target mapper tree and access values from different sources elements, you can use `xsl:variable` with different sources. However, the mapper is locked from editing. As a workaround, you can use imported maps. See Import a Map File into an Orchestrated Integration.
- The ability to use copy-of functionality is not currently available. As a workaround, import XSLT code into the mapper to achieve copy-of functionality. See Import a Map File into an Orchestrated Integration.