

Data Mapping

Objectives

After completing this lesson, you should be able to:

- Map or transform data by using the OIC Mapper tool
- Create XSL expressions using XSLT functions and operators
- Describe and use the OIC Recommendations Tool
- Edit and import advanced XSL files into OIC
- Register and use custom JavaScript functions
- Create and invoke OIC Lookups



Agenda

- OIC Data Mapper
- OIC Recommendations Engine
- Advanced Transformation Options
- Using OIC Lookups



Integration Development (Review)

**Define
Connections**

**Build Integration
Flow Logic**

Map Data

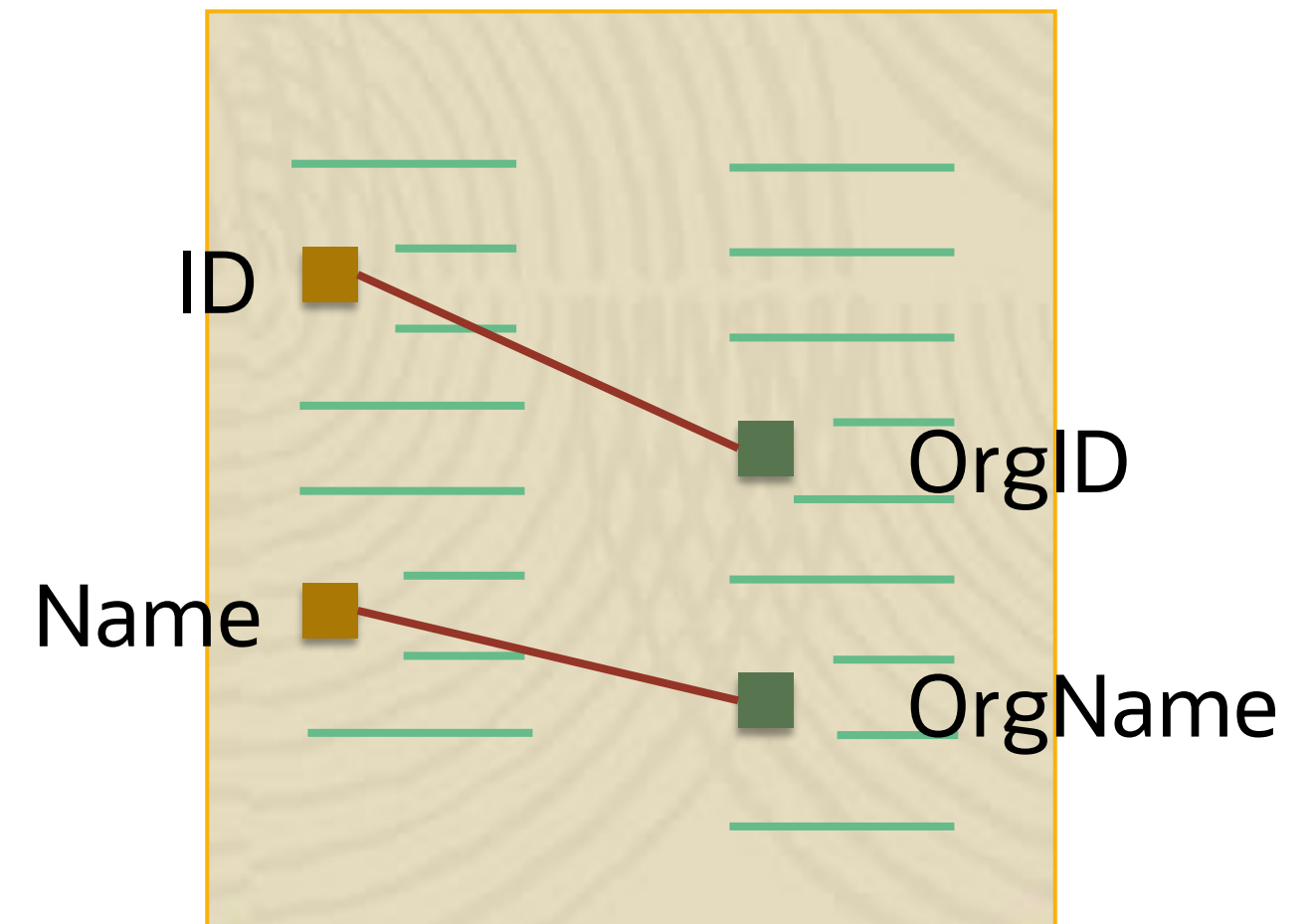
Use the OIC Data Mapper
to define payload for each
invoked connection.

**Activate
Integration**

**Monitor
Integrations**

OIC Data Mapping

- A visual mapper enables you to define data mapping between different data structures by using drag and drop.
- Automatically populate target data structures with information pulled from one or more source data structures.
- Transformation is supported between XML and XML.



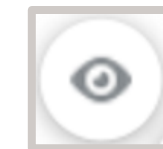
Supported Features

- Transformation:
 - XSLT visual data mapper tool
 - XPath functions (standard & custom): Define how data is modified if needed
 - OIC Lookups (specialized): Using Domain Value Maps
 - Expression Builder: For more complex expressions
- Repeat a Target element to map from different sources
- Create for-each statement automatically
- A recommend option to automate some mappings

Launching the OIC Data Mapper (Review)

Click the **Map** action icon to reveal three smaller icons and then click the pencil edit icon:

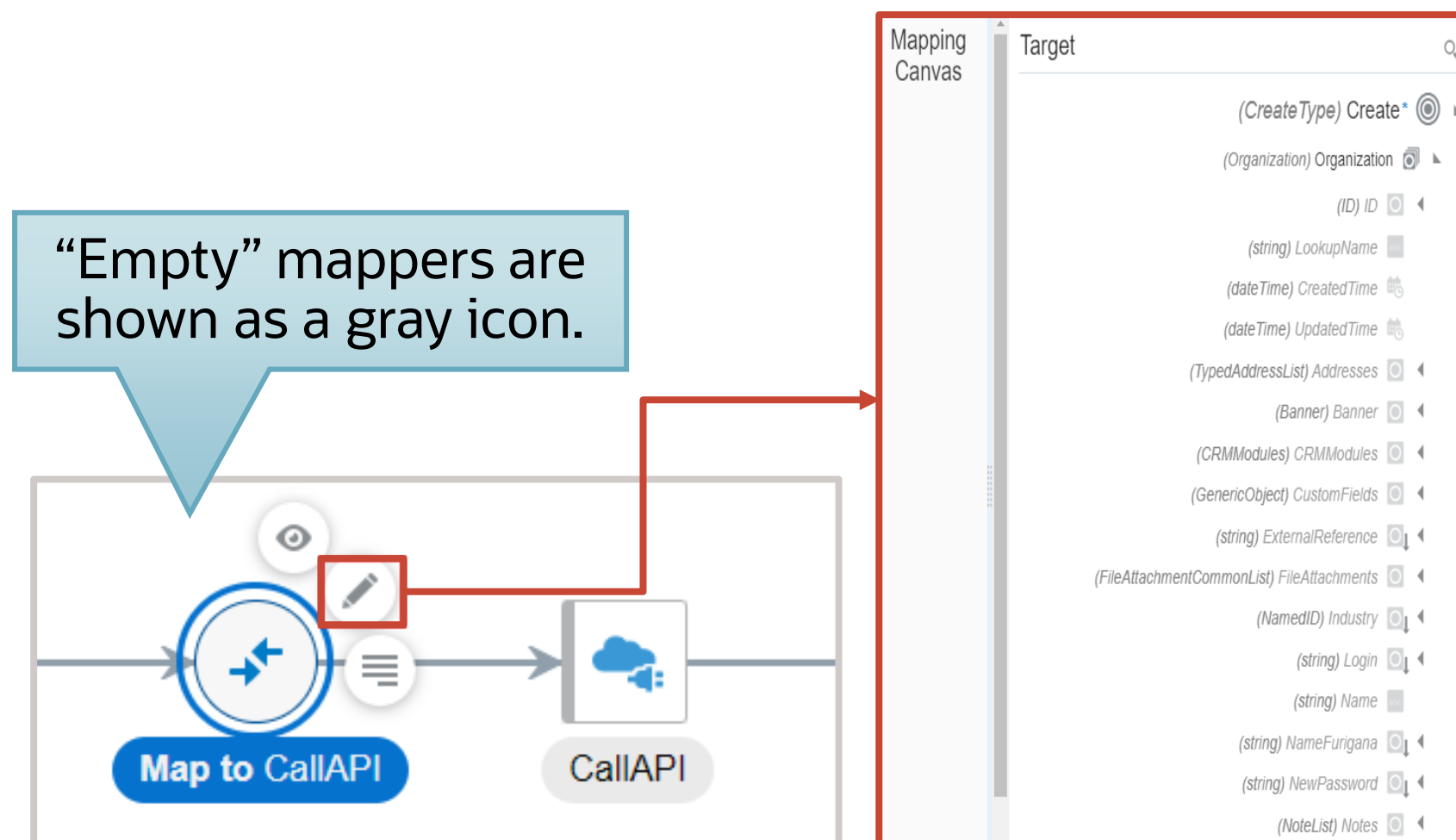
- Preceding an *Invoke* or *Call* action
- Preceding a *Return* or *Callback* action



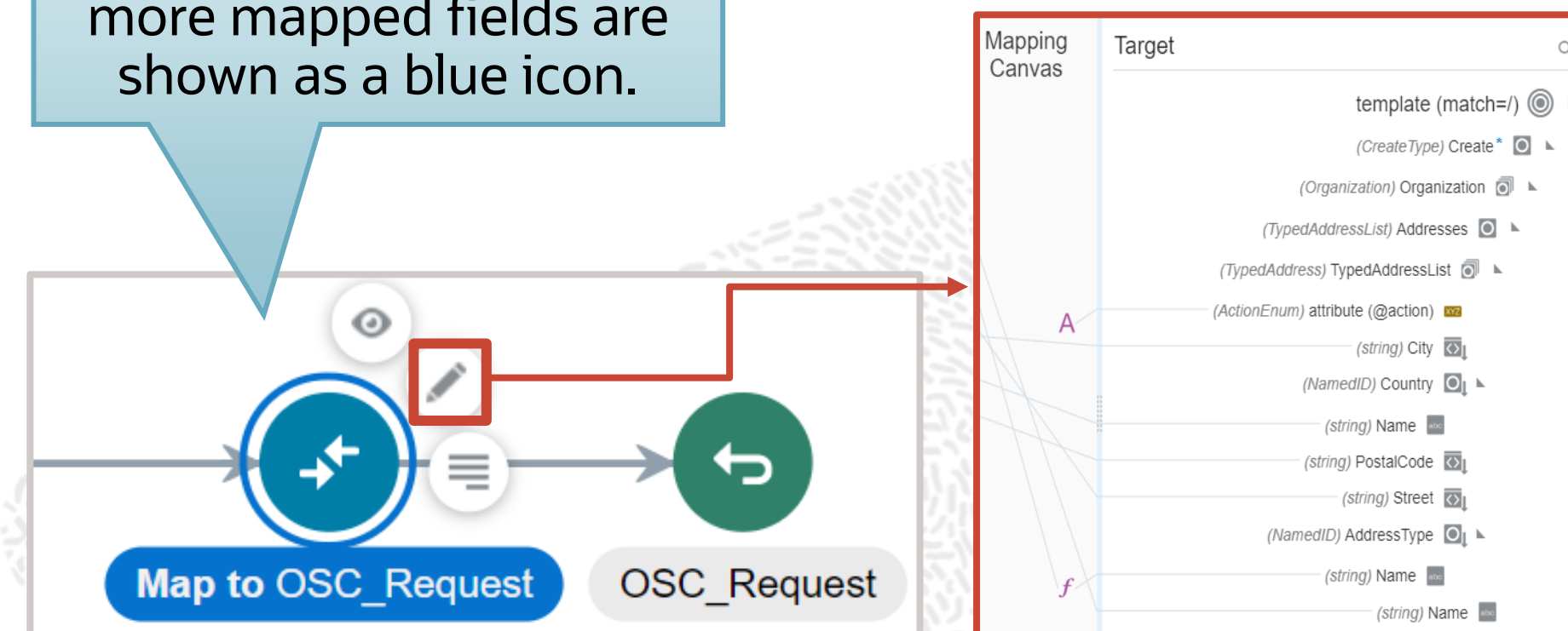
- View



- Edit



Mappers with one or more mapped fields are shown as a blue icon.



OIC Data Mapper: Designer View

Designer Code Test Recommend

Source x Mapped x Target x Mapped x

Sources

- nssrcmpr:process
 - nssrcmpr:Account
 - nsmpr2:OrganizationName
 - nsmpr2:AddressElementAttribute1
 - nsmpr2:PrimaryAddress
 - nsmpr3:AddressLine1
 - nsmpr3:City
 - nsmpr3:Country
 - nsmpr3:PostalCode

Mapping Canvas

A

f

Target

- xsl:template (match=/)
 - nstrgmp:Create
 - nstrgmp:Organization
 - nsmpr1:Addresses
 - nsmpr1:TypedAddressList
 - xsl:attribute (@action)
 - nsmpr1:City
 - nsmpr1:Country
 - rmb_v1_4:Name
 - nsmpr1:PostalCode
 - nsmpr1:Street
 - nsmpr1:AddressType
 - rmb_v1_4:Name
 - nsmpr1:Name

The **Sources** section includes the schemas of all data structures that have been received and are in scope within this integration flow.

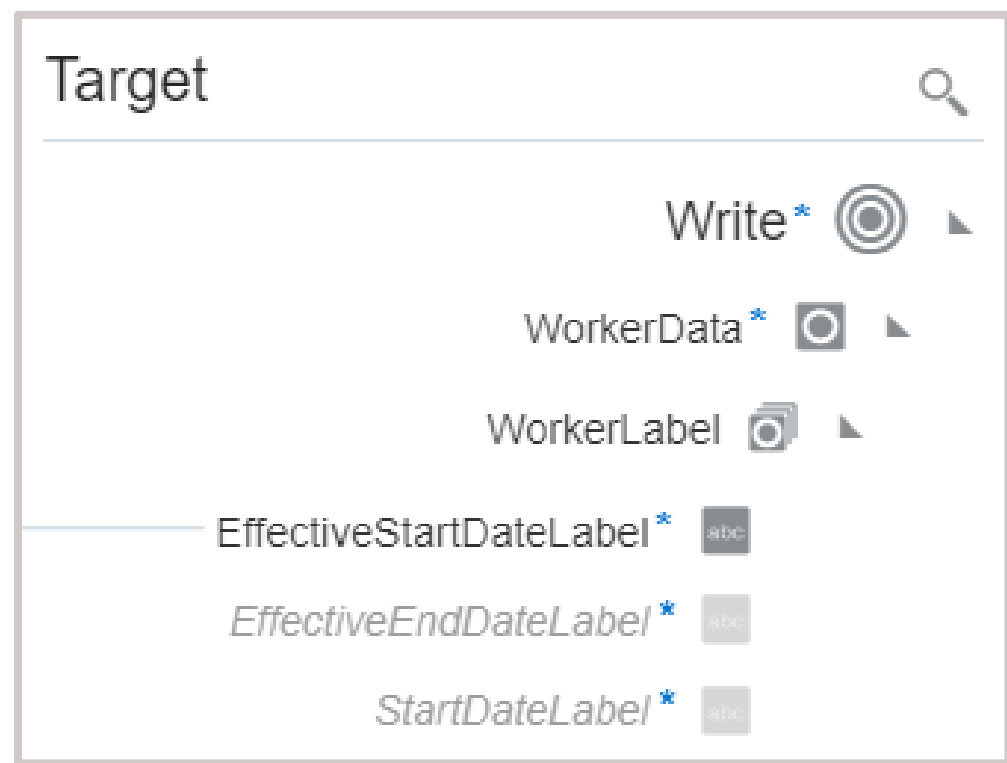
The schema for the **Target** data structure is displayed on the right.



Schema Support

Transformation maps use eXtensible Stylesheet Language (XSL).

- All Source and Target data objects are represented internally as XML structures.
- Mapping supports both qualified and unqualified schemas.
- Required fields are identified by a blue asterisk (*).



Right-click an element or attribute and select **Node Info** for more details.

organization

abc	PartyNum
123	PartyId
abc	PartyType
abc	PartyName

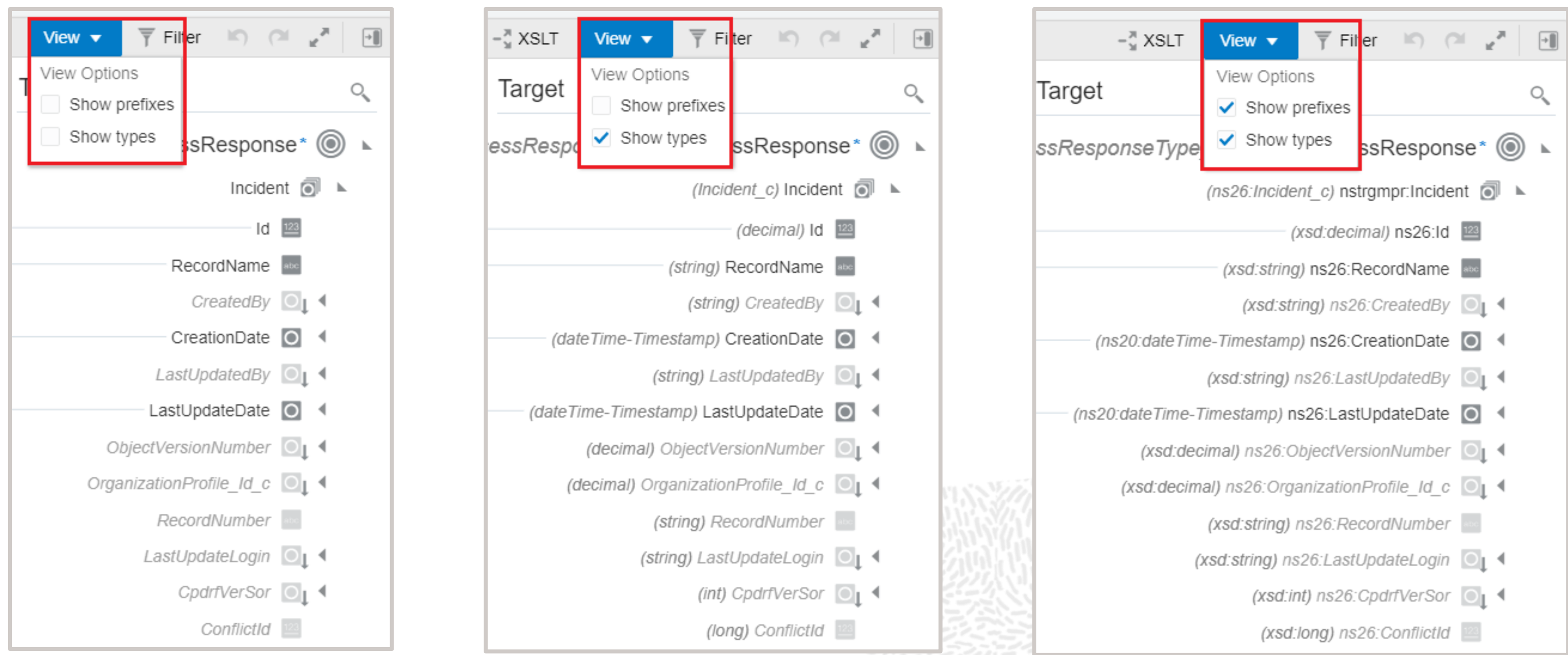
Schema Info

DataType:	xsd:long	ContentType:	Simple	NodeType:	element
Required:	false	Nilable:	false	Abstract:	false
Repeating:	false	minOccurs:	0	maxOccurs:	1
XPath:	/nssrcmpr:createOrganizationAsync/nssrcmpr:organizationParty/nsmpr5:PartyId				



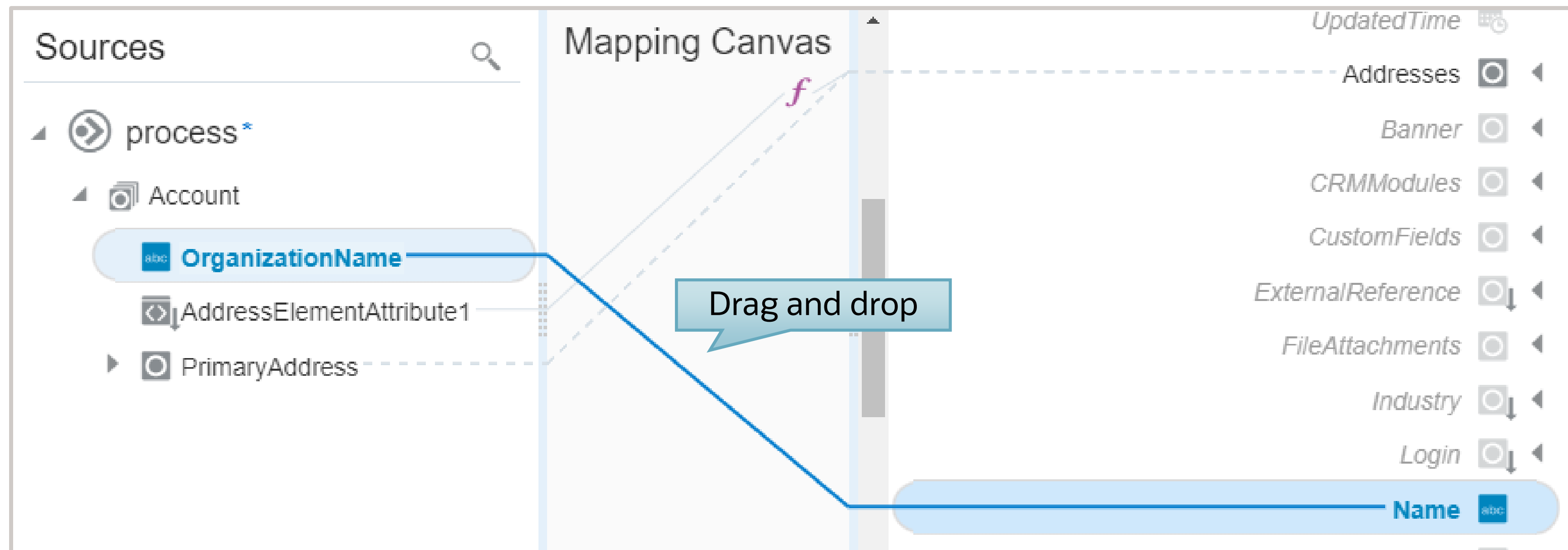
Data Structure View Options

Click the **View** button to select display options for data element nodes and attributes.



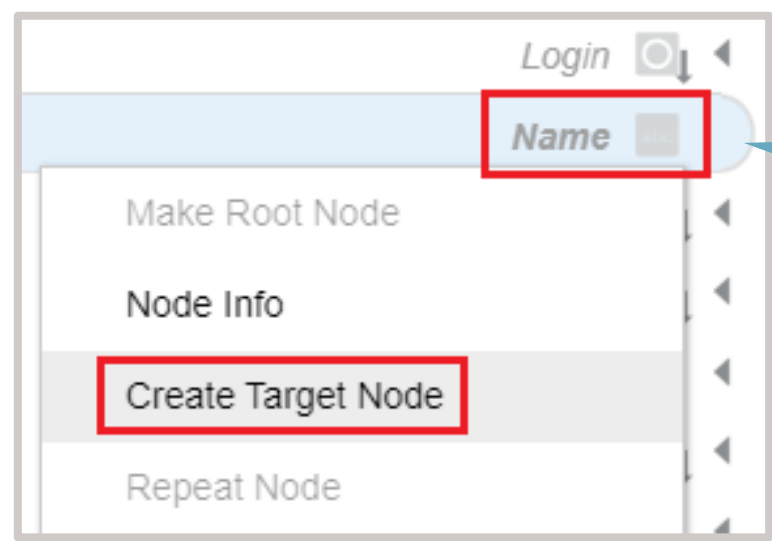
Basic Field Mapping: Drag & Drop

Click a field in a **Source** data field and drag it to the required **Target** field.




Basic Field Mapping: Using the Expression Builder


Right-click an empty **Target** field and select *Create Target Node* which reveals the Expression Builder for that element.



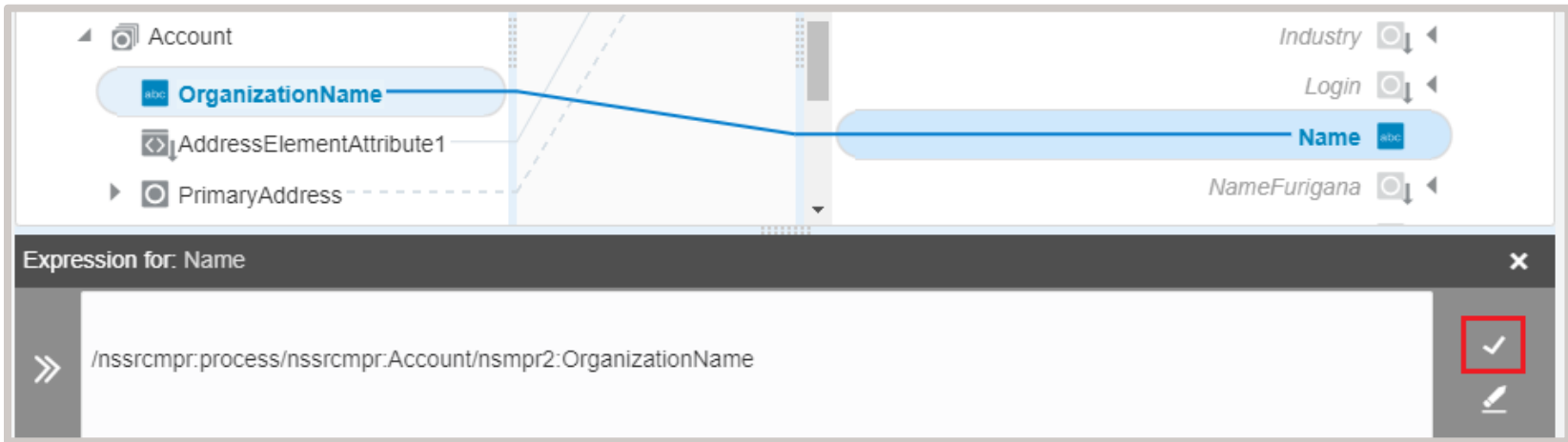
Right-click



Select a **Source** data field



Use the shuttle button to insert expression

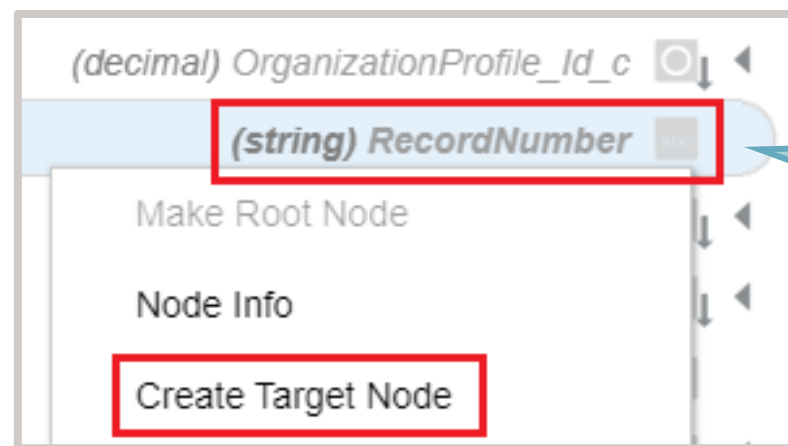


Click the check mark to save the mapping



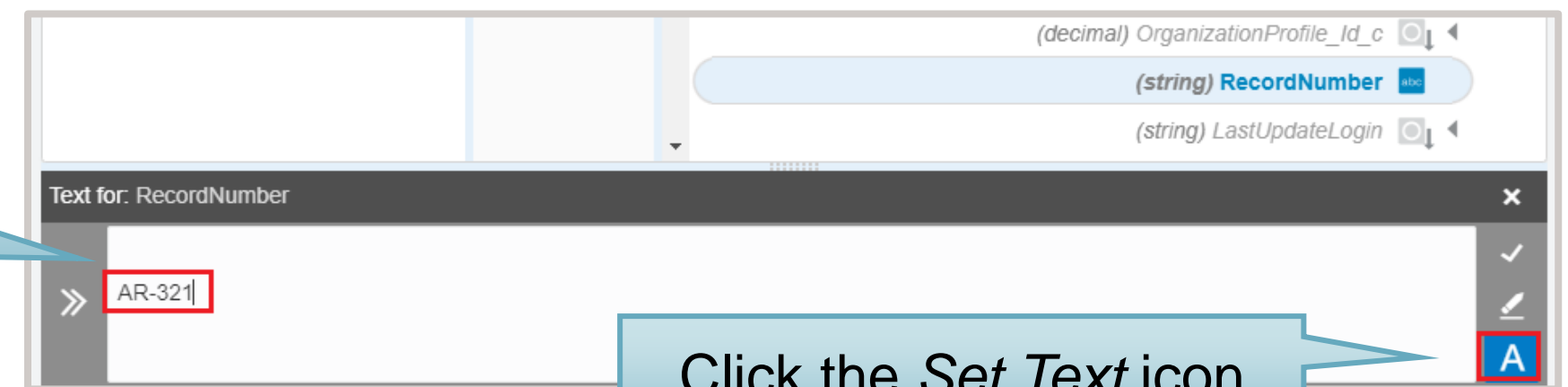
Using the Set Text Mode

Right-click an empty **Target** field and select *Create Target Node* which reveals the Expression Builder for that element.

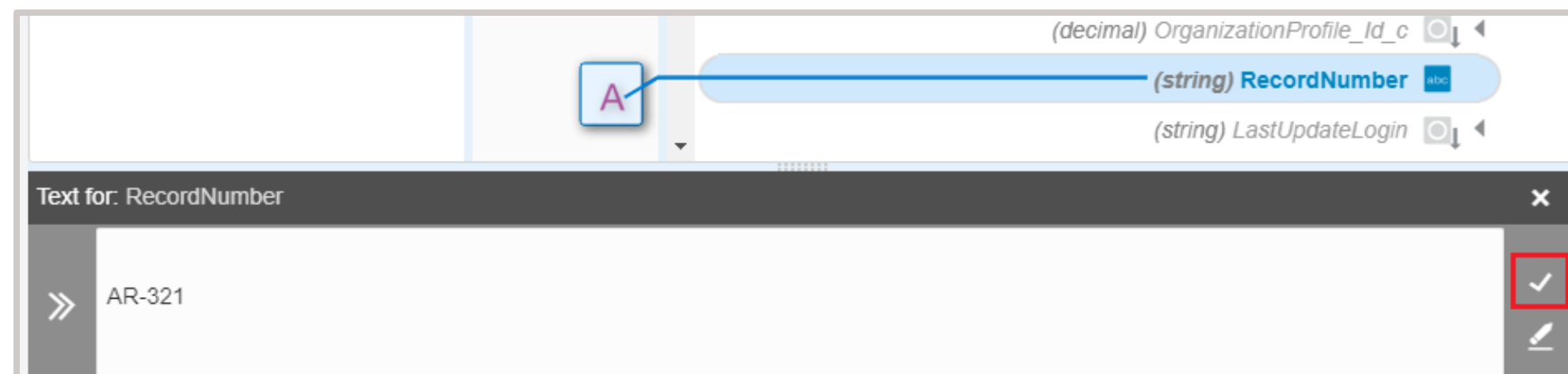


Right-click

Enter the text



Click the *Set Text* icon

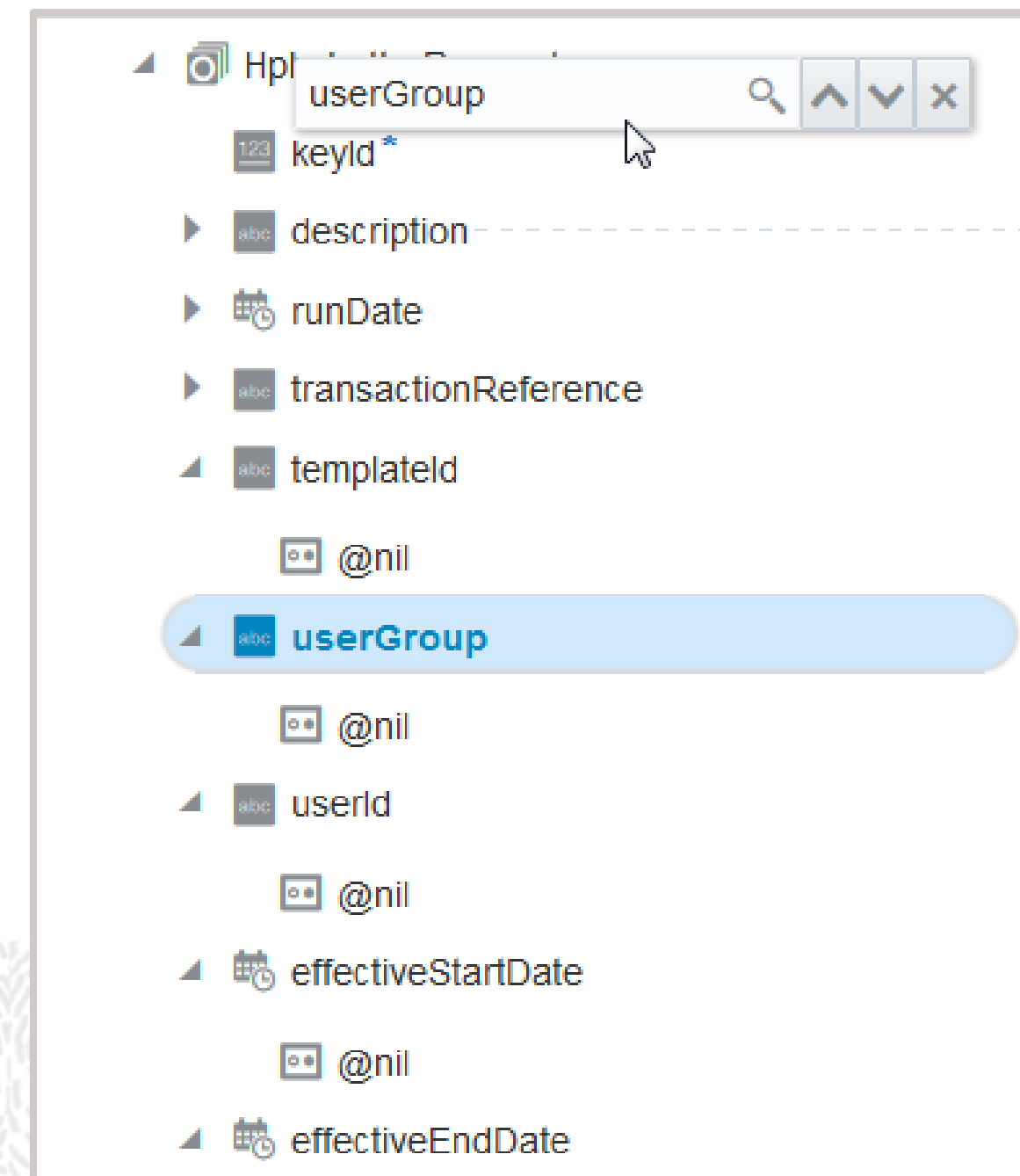
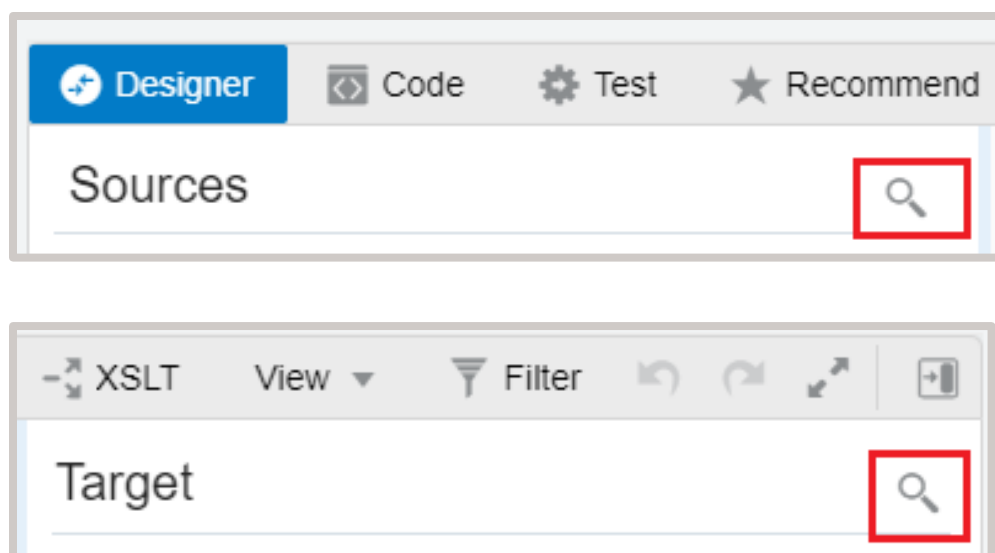


Click the check mark to save the mapping

Search Data Fields

You can search for specific element nodes or attributes in either the Source or Target structure.

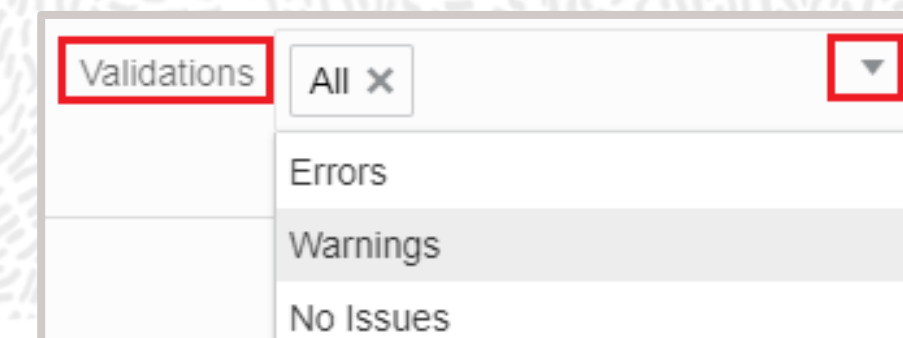
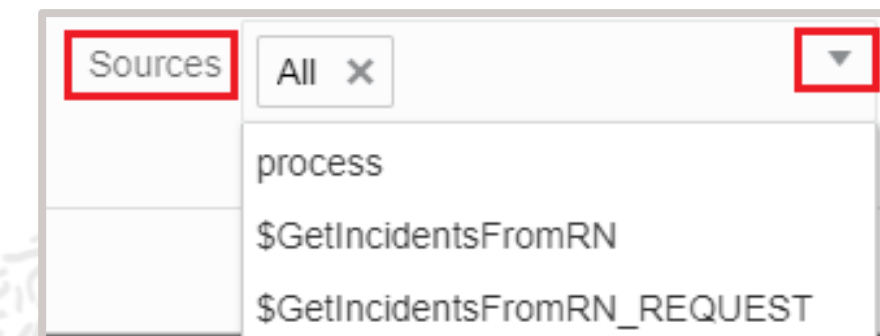
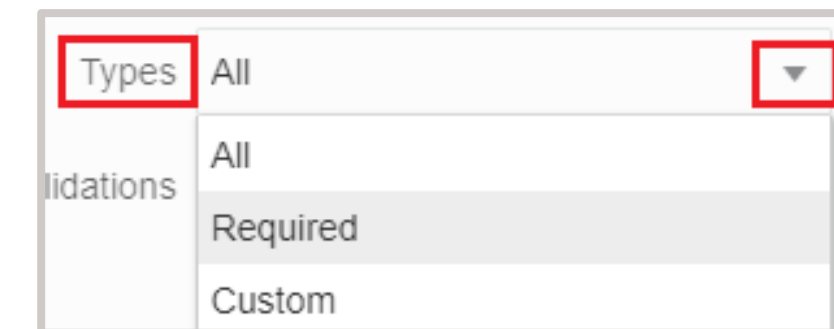
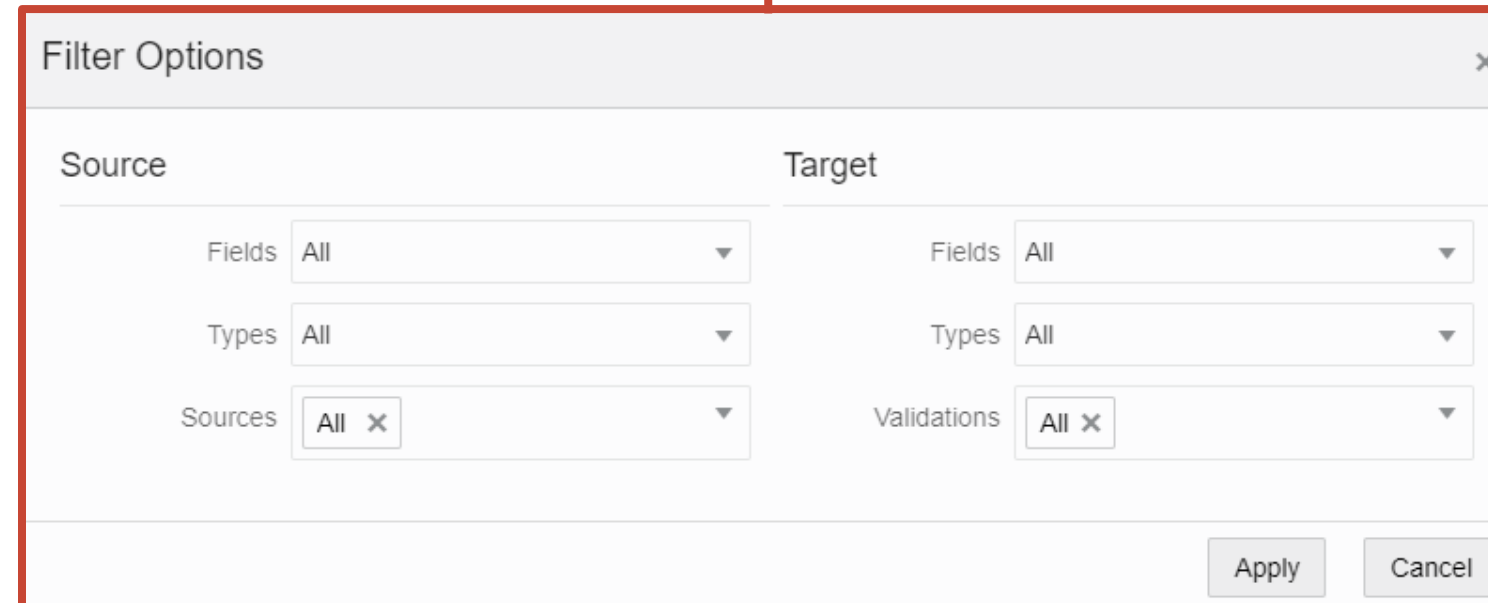
- Click the *Search* icon
- Enter a full or partial name
- Click the **V** icon to advance
- Click the **X** icon to close the Search bar



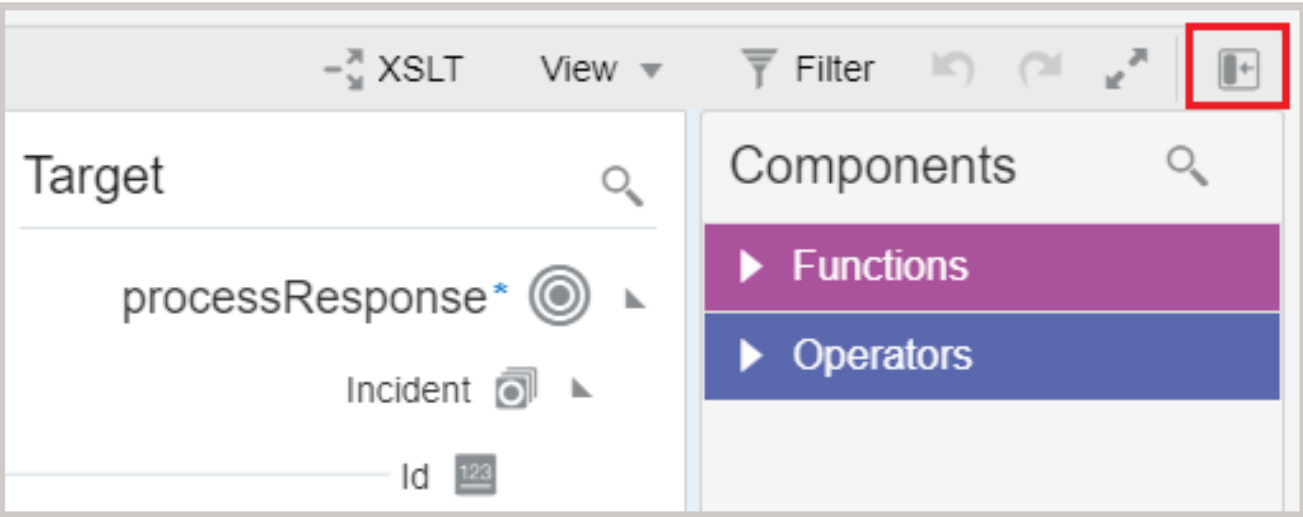
Filter the Source or Target Data Structures

You can filter the display of the source and/or target structures.

- Fields (*mapped, unmapped*)
- Types (*required, custom*)
- Sources
- Target Validations (*errors, warnings, no issues*)



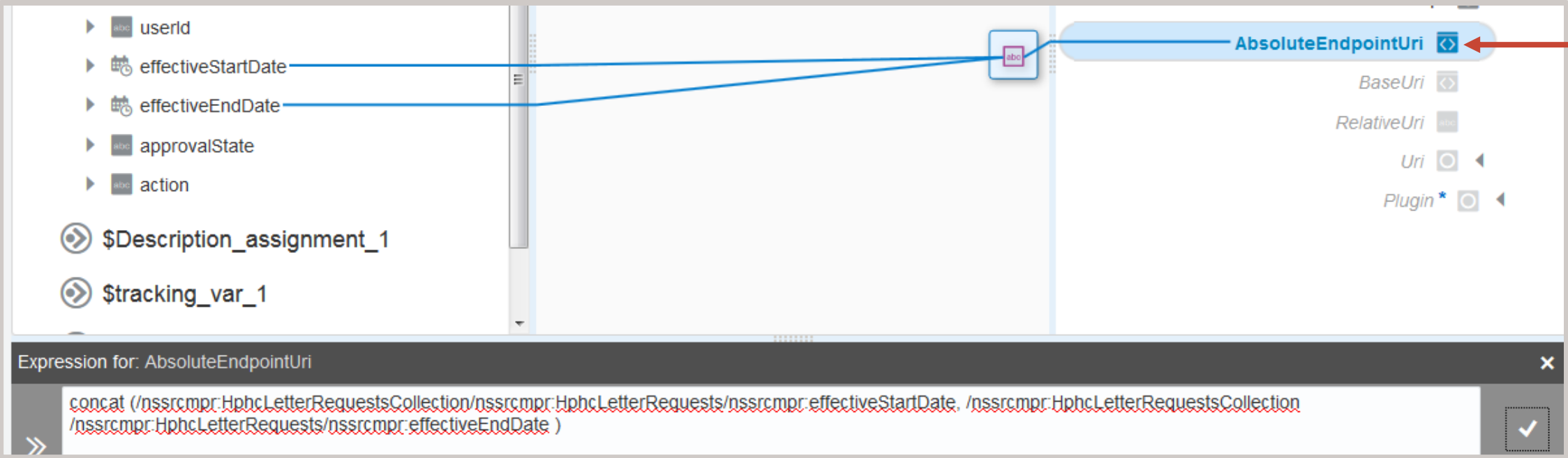
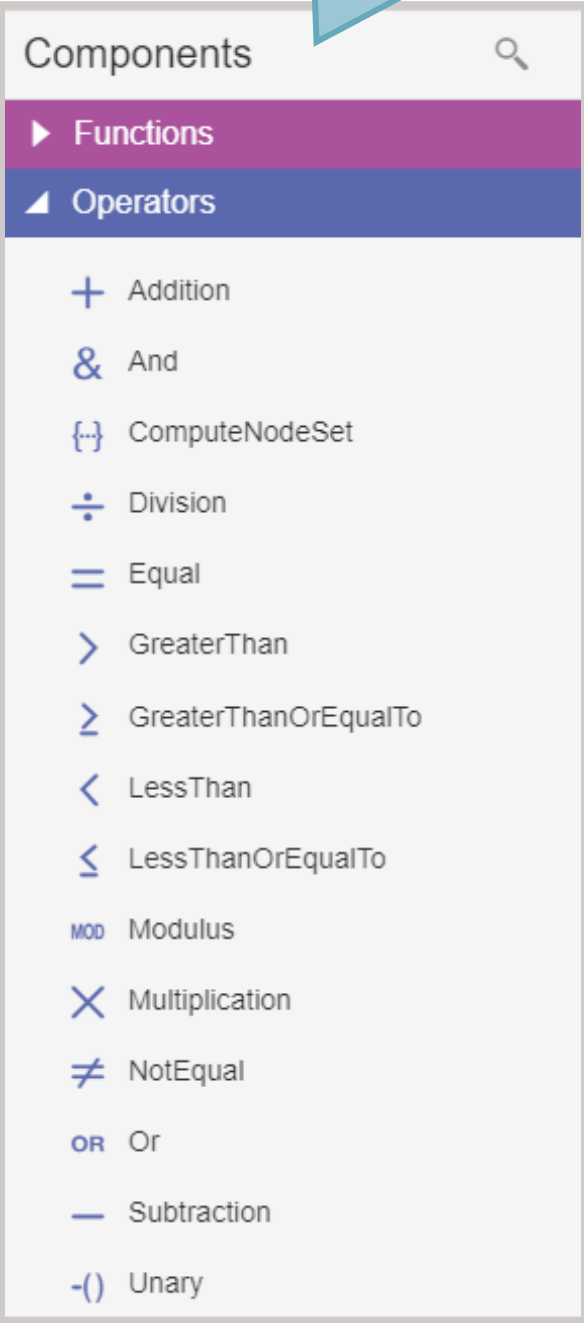
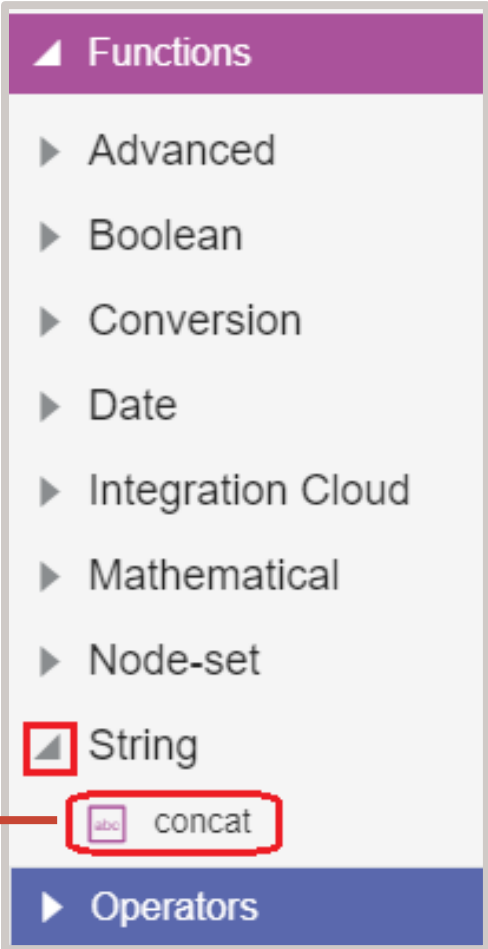
Using XPath Functions and Operators



Click the *Toggle functions* icon to expose the **Components** panel.

Expand a category to locate the function, then drag to the target element or to the Expression box.

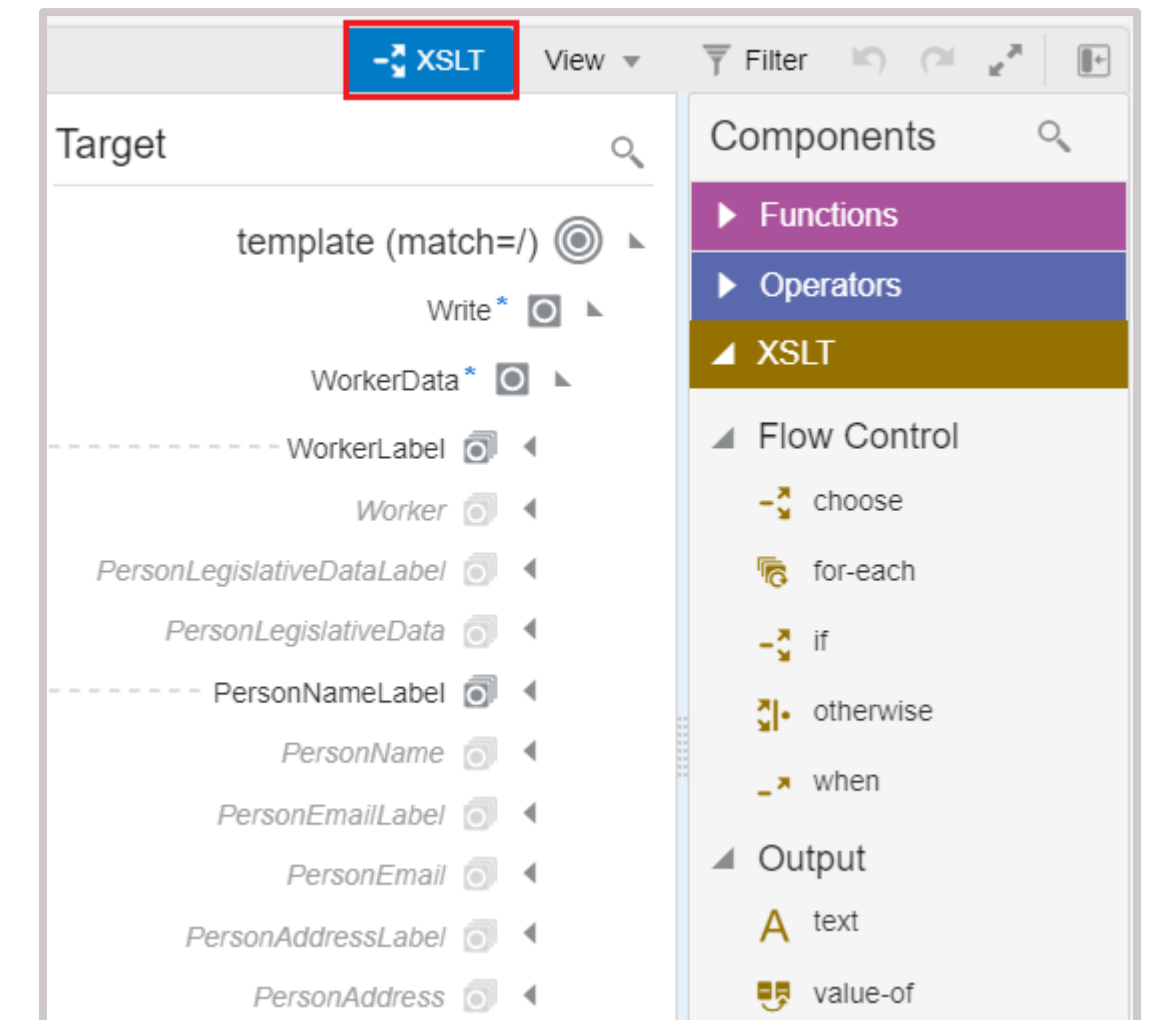
Expand to display XPath operators



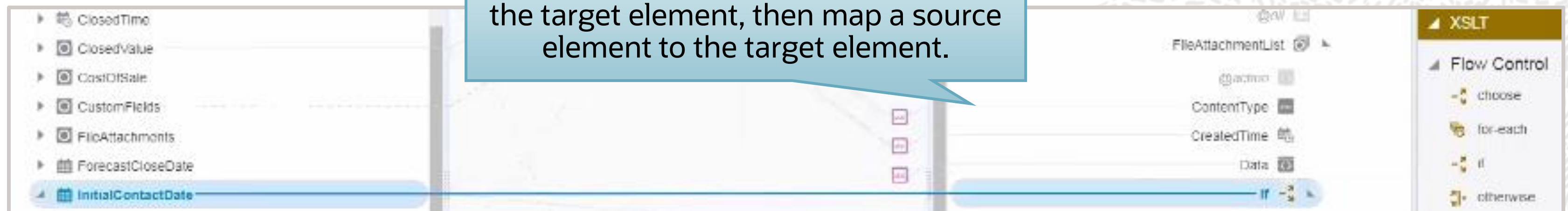
Working With XSLT Statements

Click the **XSLT** (or **Advanced**) button to add the XSLT header to the *Components* panel.

- Expand the XSLT header tab.
- Browse for and drag the appropriate statement onto the target element node.
- If the element is grayed out (is a ghost node), right-click the element and select *Create Target Node*.

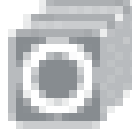


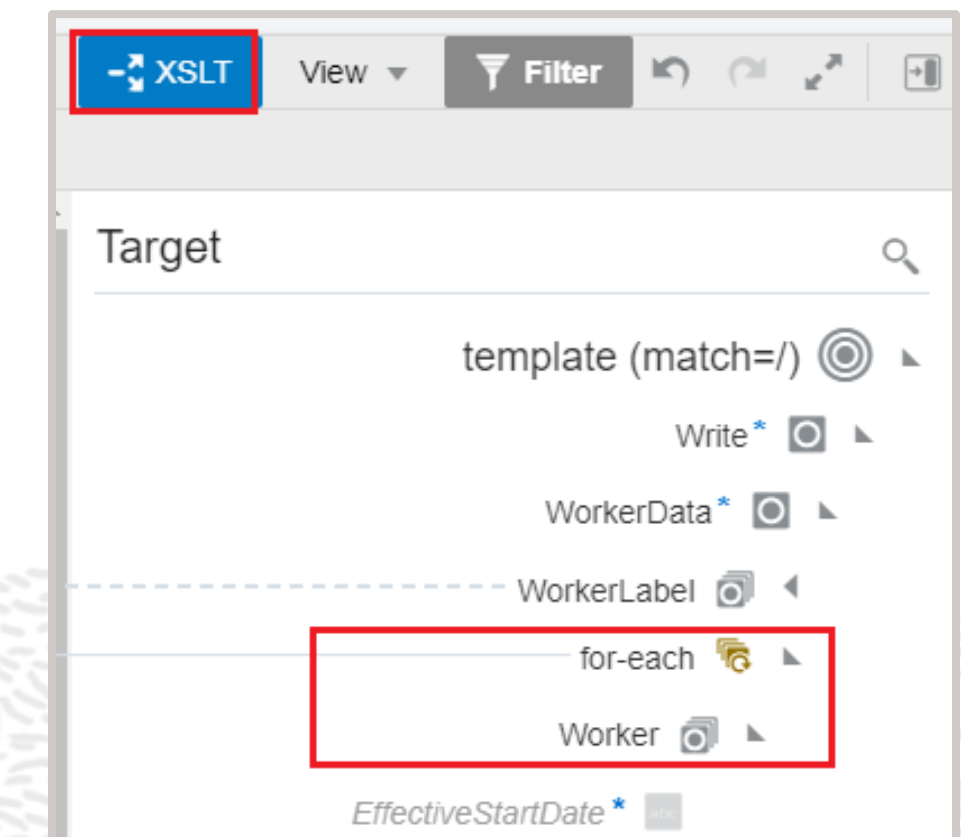
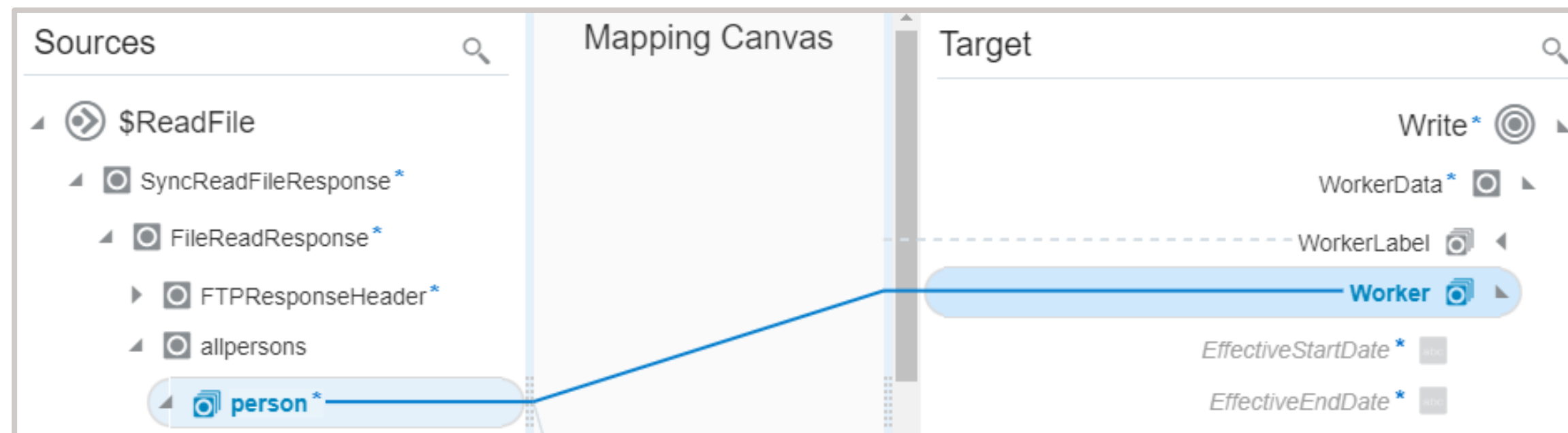
For example, drag an **if** statement to the target element, then map a source element to the target element.



Automatically Create for-each Statements

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

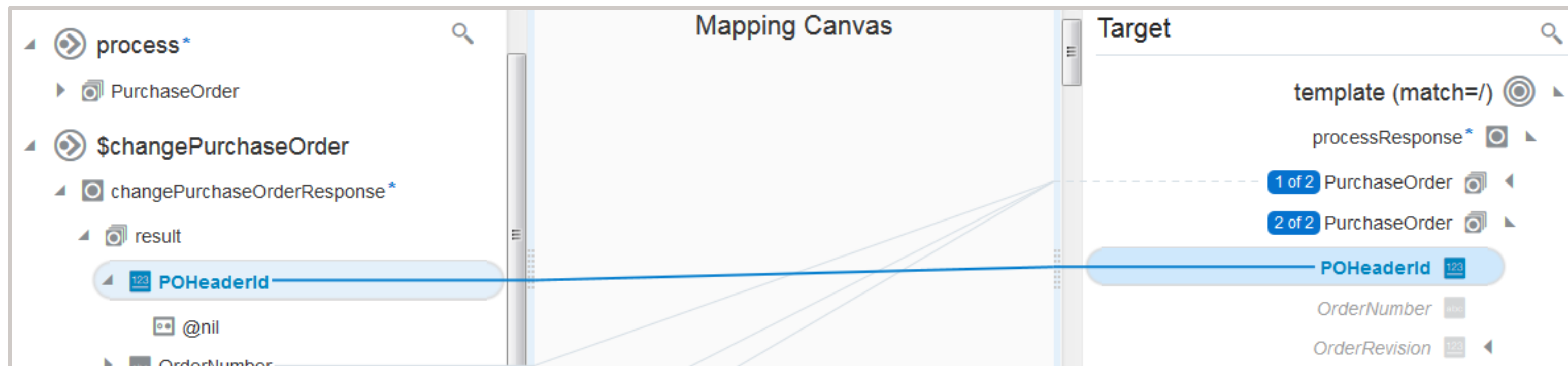
- Repeatable elements are identified by this icon: 
- Toggle the XSLT button to view created for-each statement



Repeat a Target Element to Map to Different Sources

You can repeat a target element in the mapper, enabling 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.

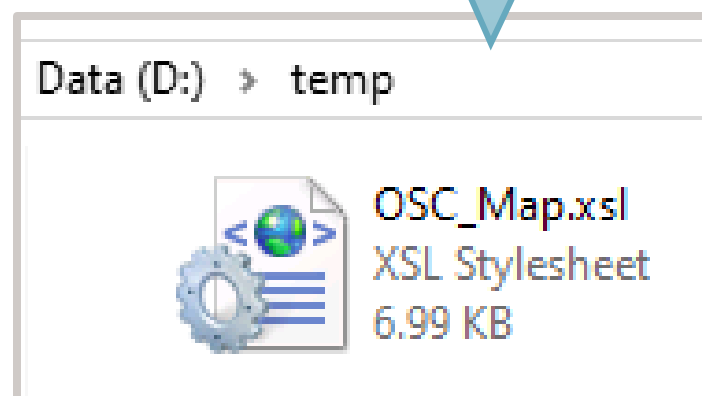


Accessing the XSL Stylesheet

Click the **Code** tab to display the source stylesheet.

If desired, copy contents to be viewed in a separate application.

Copy and paste all contents and save as an .xsl file



```

Designer  Code  Test  Recommend
17      <oracle-xsl-mapper:param name="GetIncidentsFromRN_REQUEST" xml:id="id_43"/>
18      </oracle-xsl-mapper:source>
19      <oracle-xsl-mapper:source type="XSD" xml:id="id_44">
20      <oracle-xsl-mapper:schema
location="../../processor_63/resourcegroup_64/ICSIntegrationMetadata.xsd" xml:id="id_45"/>
21      <oracle-xsl-mapper:rootElement name="metadata"
namespace="http://www.oracle.com/2014/03/ic/integration/metadata" xml:id="id_46"/>
22      <oracle-xsl-mapper:param name="self" xml:id="id_47"/>
23      </oracle-xsl-mapper:source>
24      </oracle-xsl-mapper:mapSources>
25      <oracle-xsl-mapper:mapTargets xml:id="id_7">
26      <oracle-xsl-mapper:target type="WSDL" xml:id="id_8">
27      <oracle-xsl-mapper:schema
location="../../application_18/outbound_19/resourcegroup_20/OSC_Request_REQUEST.wsdl" xml:id="id_9"/>
28      <oracle-xsl-mapper:rootElement name="processResponse"
namespace="http://xmlns.oracle.com/cloud/adaptor/osc/OSC_Request_REQUEST/types" xml:id="id_10"/>
29      </oracle-xsl-mapper:target>
30      </oracle-xsl-mapper:mapTargets>
31      <!--GENERATED BY ORACLE XSL MAPPER 12.1.2.0.0-->
32      </oracle-xsl-mapper:schema>
33      <!--User Editing allowed BELOW this line - DO NOT DELETE THIS LINE-->
34      <xsl:param name="GetIncidentsFromRN" xml:id="id_25"/>
35      <xsl:param name="GetIncidentsFromRN_REQUEST" xml:id="id_48"/>
36      <xsl:param name="self" xml:id="id_49"/>
37      <xsl:param name="tracking_var_1" xml:id="id_50"/>
38      <xsl:param name="tracking_var_2" xml:id="id_51"/>
39      <xsl:param name="tracking_var_3" xml:id="id_52"/>
40      <xsl:template match="/" xml:id="id_11">
41      <nstrgmpr:processResponse xml:id="id_12">
42      <nstrgmpr:Incident>
43      <ns26:Id>
44      <xsl:value-of select="/nstrgmpr:process/nstrgmpr:Account/ns25:PartyId"/>
45      </ns26:Id>
46      <ns26:CreationDate>
47      <xsl:value-of
select="$GetIncidentsFromRN/nsmpr0:QueryObjectsResponse/nsmpr0:Incident/rnb_v1_4:CreatedTime"/>
48      </ns26:CreationDate>
49      </nstrgmpr:Incident>
50      </nstrgmpr:processResponse>
51      </xsl:template>
52      </xsl:stylesheet>
    
```


Testing Your Mappings

Click the **Test** button to open testing tool

Click to generate random inputs

The screenshot shows the Oracle Mapping Designer interface with the **Test** tab selected. The **Source** pane displays the XML payload for the `createOrganizationAsync` process. The **Target** pane displays the resulting XML structure for the `nstrgmpr:Create` process. The interface includes a **Test** button in the top toolbar, a **Generate Inputs** button, and an **Execute** button. Callouts indicate the following actions:

- Click the **Test** button to open testing tool
- Click to generate random inputs
- Click to execute then view the resulting target data structure
- Replace random generated inputs or copy/paste other test payloads



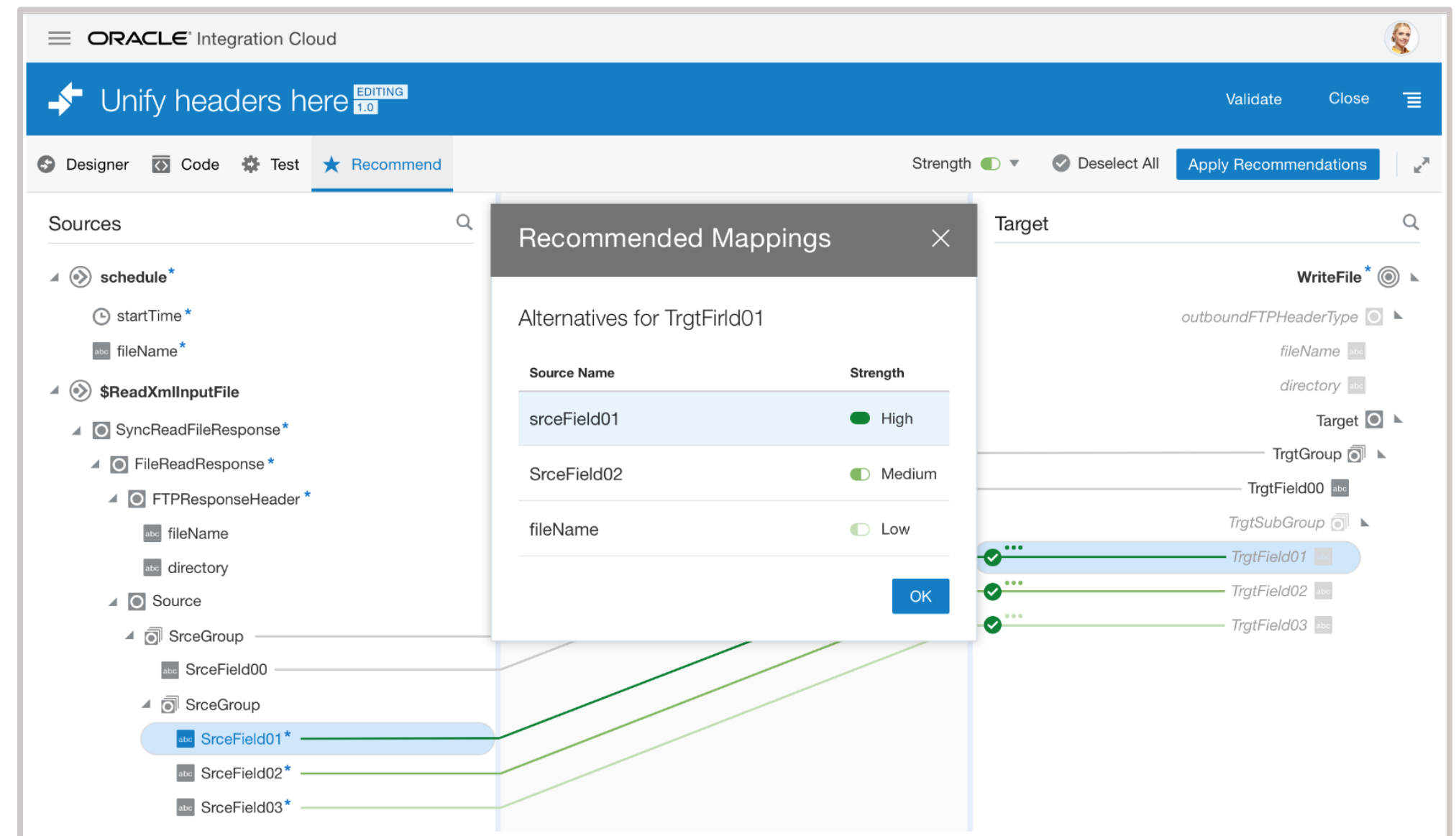
Agenda

- OIC Data Mapper
- OIC Recommendations Engine
- Advanced Transformation Options
- Using OIC Lookups



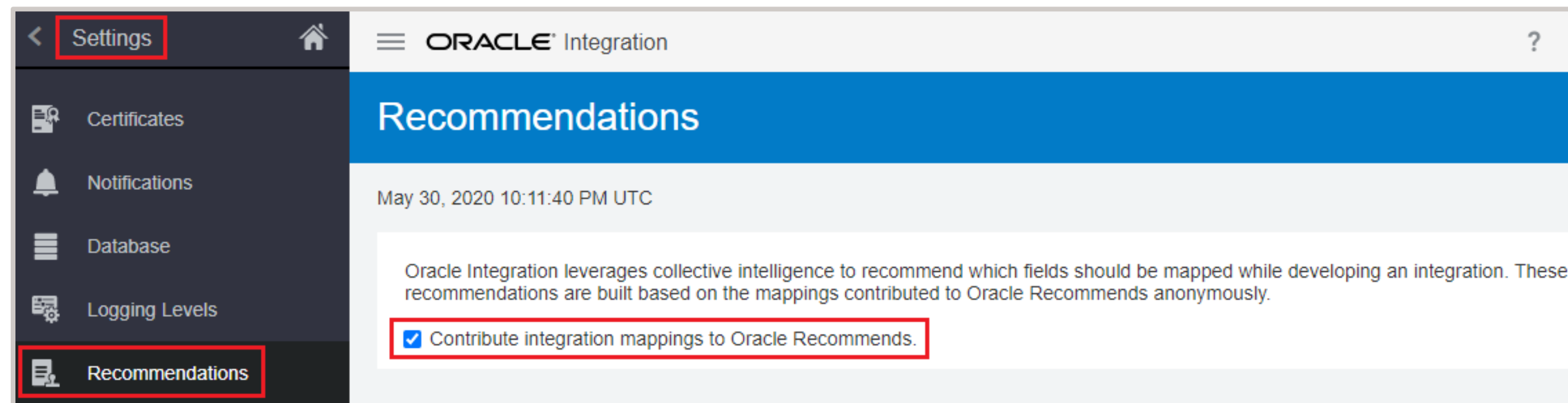
Data Mapping Recommendations

- AI and ML powered to save up to 60 percent of development effort
- Auto mapping and transformation to connect applications faster
- Machine learning guidance in OIC based on community usage
- Recommended Mappings highlight proven and popular best practices



Enabling the Oracle Recommendations Engine

By default, the recommendations engine is enabled. When enabled, all integrations on that instance are published to the recommendations engine.



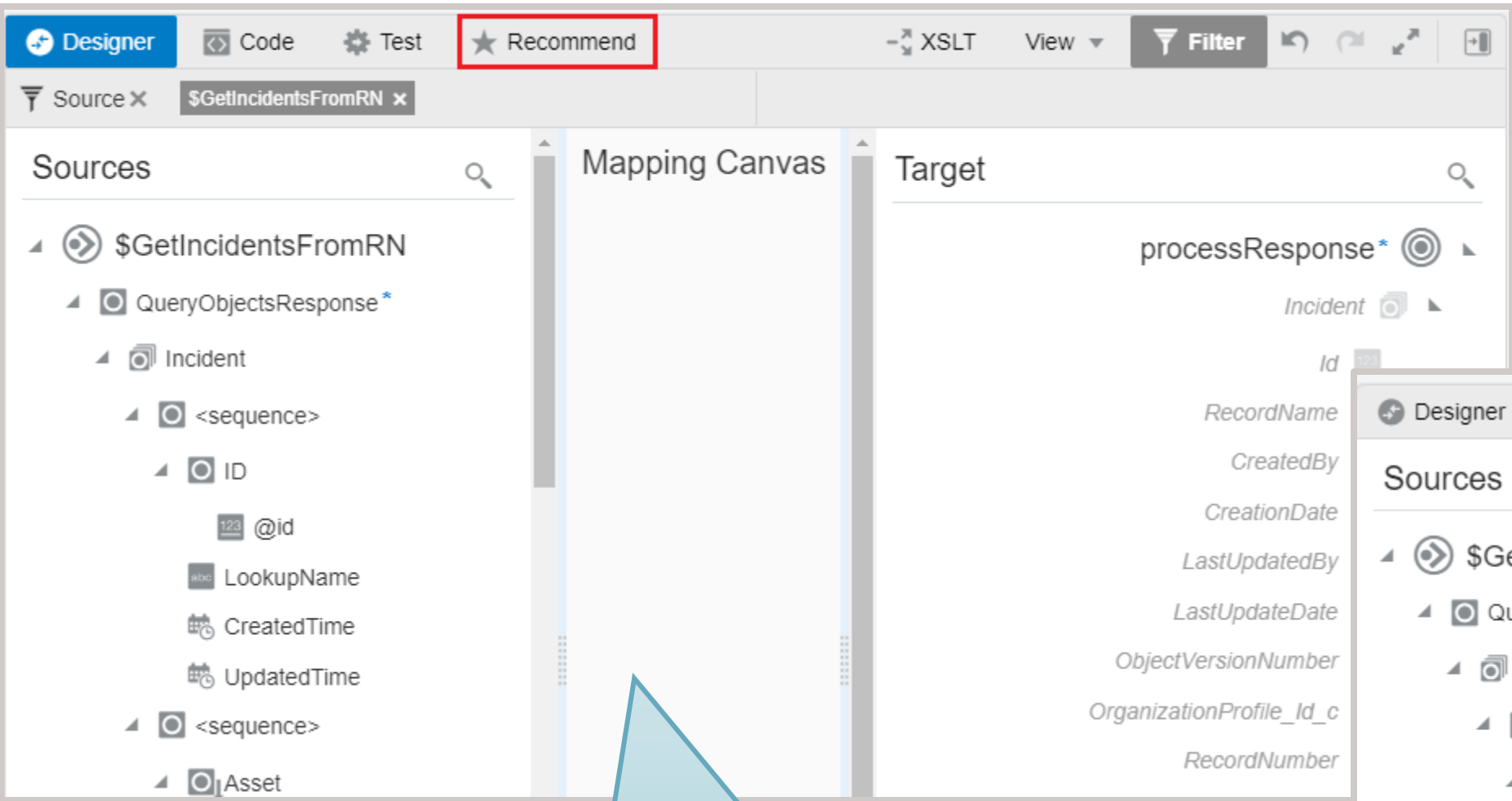
However, you can deselect this option for individual integrations upon activation.

Oracle Recommends

☒ Contribute integration mappings to Oracle Recommends.

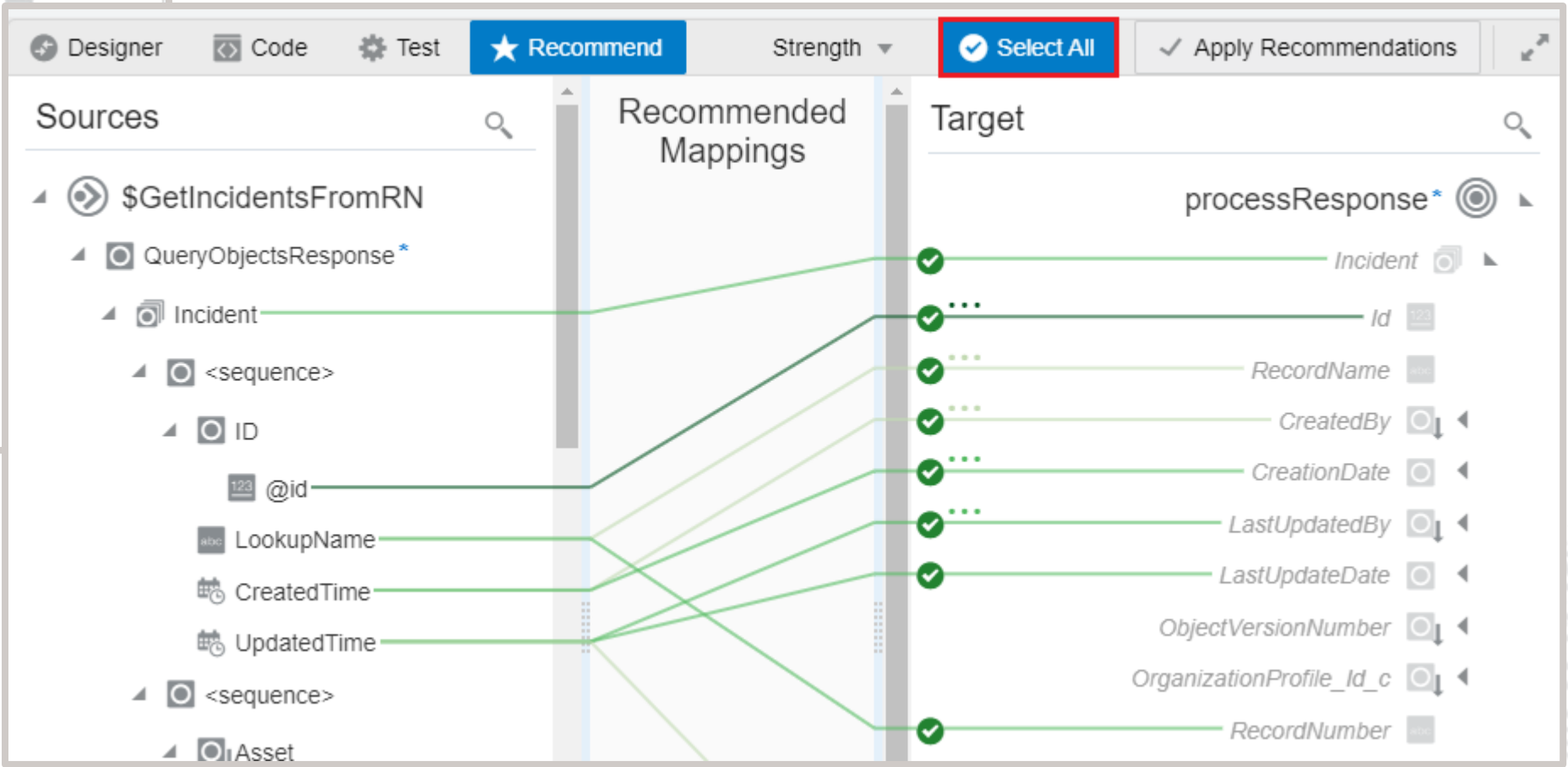
i Oracle Integration leverages the collective intelligence to recommend which fields should be mapped while developing an integration. These recommendations are built based on the mappings contributed to Oracle Recommends anonymously. Unselect the checkbox if you do not wish to contribute the mappings. You may change this in recommendations page from settings menu.

Launching the Mapper Recommendations



By default, all recommendations are selected.

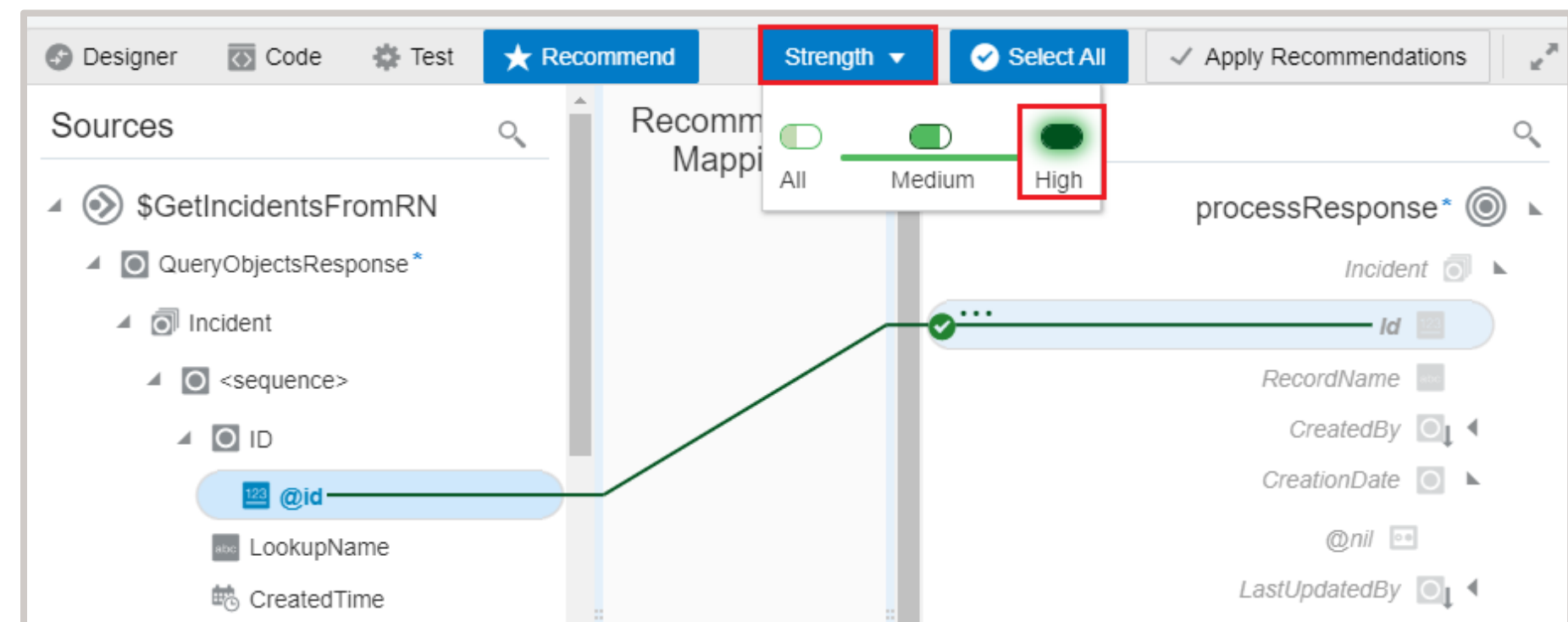
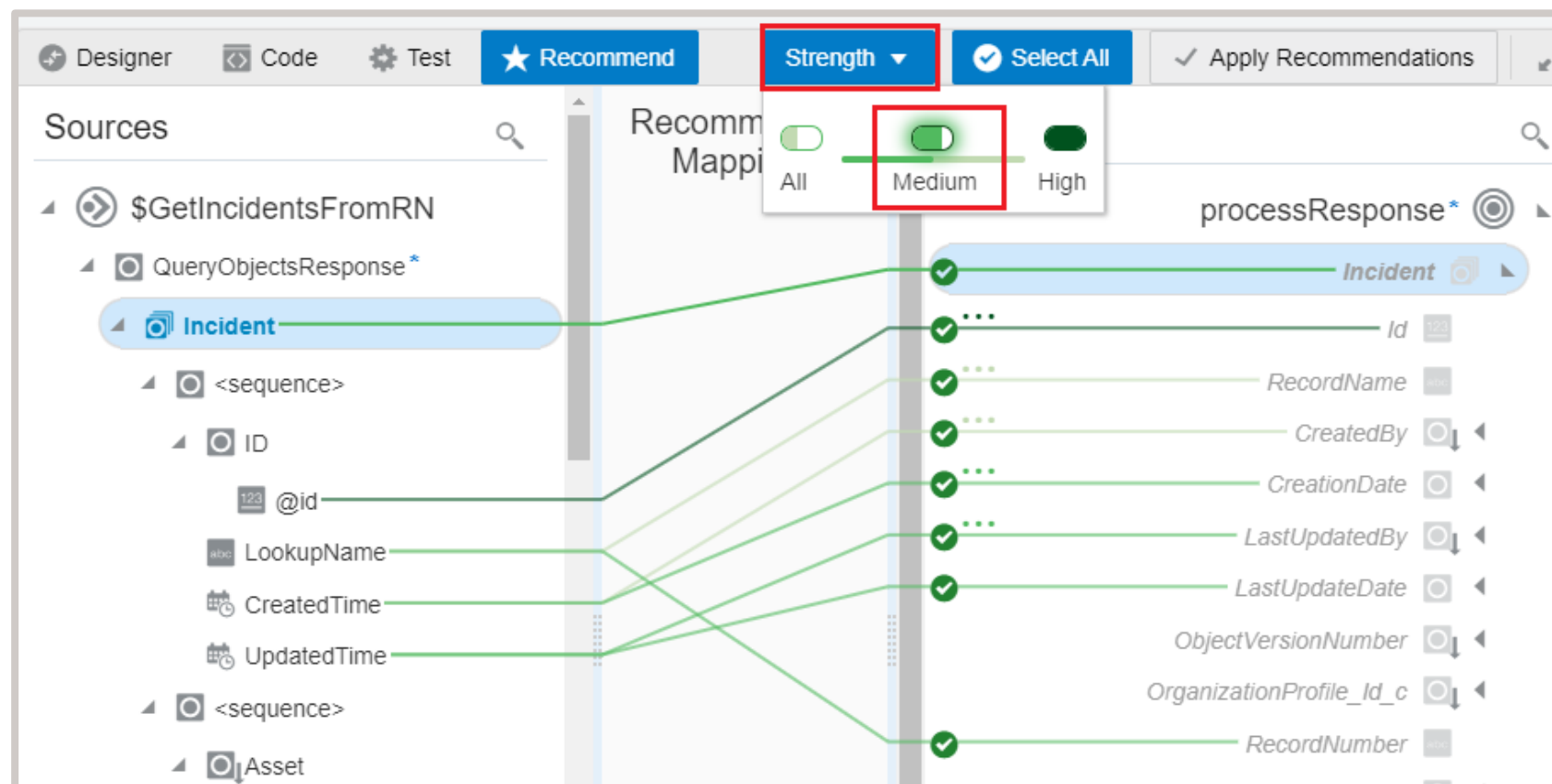
Open the mapper, and then click the **Recommend** tab



Displaying By Strength

Recommendations are categorized by three strength levels: Low, Medium, High.

- Select Medium or High to remove the display of lower strength recommendations.



Selecting and Accepting Recommendations

Click a target icon to deselect or reselect a recommendation...

The screenshot shows a list of recommended mappings. The first row, 'Id', is highlighted with a red box around its target icon. Below it, 'RecordName' and 'CreatedBy' have their target icons circled in red. 'CreationDate' has a green checkmark next to it.

...or open the *Recommended Mappings* box to select recommended alternatives.

The dialog box titled 'Recommended Mappings' shows a 'Select' button at the top. Below it, under 'Alternates for Id', there is a table with 'Source Name' and 'Strength' columns. The first row, 'ID/@id', is highlighted with a red box and has its 'High' strength toggle turned on. Other rows are 'ID' (Medium) and 'PartyId' (Low). An 'OK' button is at the bottom right.

Source Name	Strength
ID/@id	High
ID	Medium
PartyId	Low

The main application window has three panes: 'Sources' on the left, 'Recommended Mappings' in the center, and 'Target' on the right. The 'Sources' pane shows a tree structure with '\$GetIncidentsFromRN' selected. The 'Recommended Mappings' pane shows a list of mappings with green checkmarks. The 'Target' pane shows a 'processResponse*' target with a list of fields. A red box highlights the 'Apply Recommendations' button in the top right corner.

When finished, click **Apply Recommendations** to accept and apply selected mappings.



Complete the Data Mapping

Selected recommendations are applied and the mapper returns to **Designer** view

Continue to edit additional mappings (as required), then click **Validate** to save.

The screenshot displays the Oracle Data Mapper interface for a mapping titled "Map to OSC_Request" under the project "Mills Test (1.0)". The interface is in the "Designer" view, with tabs for "Designer", "Code", "Test", and "Recommend". A "Filter" button and navigation icons are also visible.

The mapping configuration is shown in three panels:

- Sources:** A tree view showing the source data structure. The selected source is "\$GetIncidentsFromRN". Below it, the structure includes:
 - QueryObjectsResponse*
 - Incident
 - <sequence>
 - ID
 - @id (value: 123)
 - LookupName (value: abc)
 - CreateTime
 - UpdateTime
 - <sequence>
 - Asset

- Mapping Canvas:** A central workspace where lines connect source elements to target elements. In this view, lines connect the "Incident" source to the "processResponse" target, and the "ID" source to the "Id" target.
- Target:** A tree view showing the target data structure. The selected target is "processResponse*". Below it, the structure includes:
- Incident
 - Id (value: 123)
 - RecordName (value: abc)
 - CreatedBy
 - CreationDate
 - LastUpdatedBy
 - LastUpdateDate
 - ObjectVersionNumber
 - OrganizationProfile_Id_c
 - RecordNumber (value: abc)

Agenda

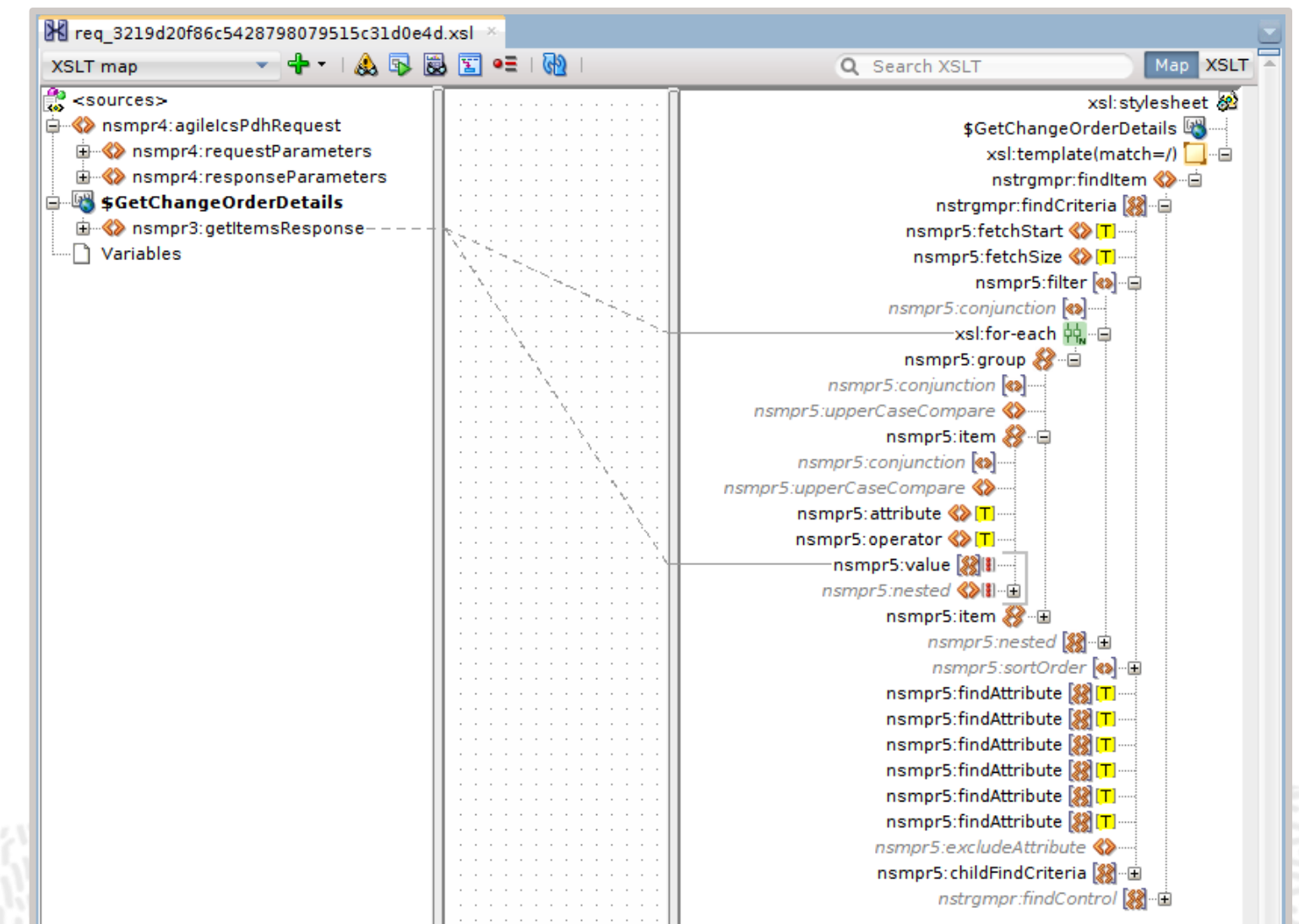
- OIC Data Mapper
- OIC Recommendations Engine
- Advanced Transformation Options
- Using OIC Lookups



Creating Transformations Outside of OIC

Why might you choose to edit a transformation outside of the OIC Mapper tool?

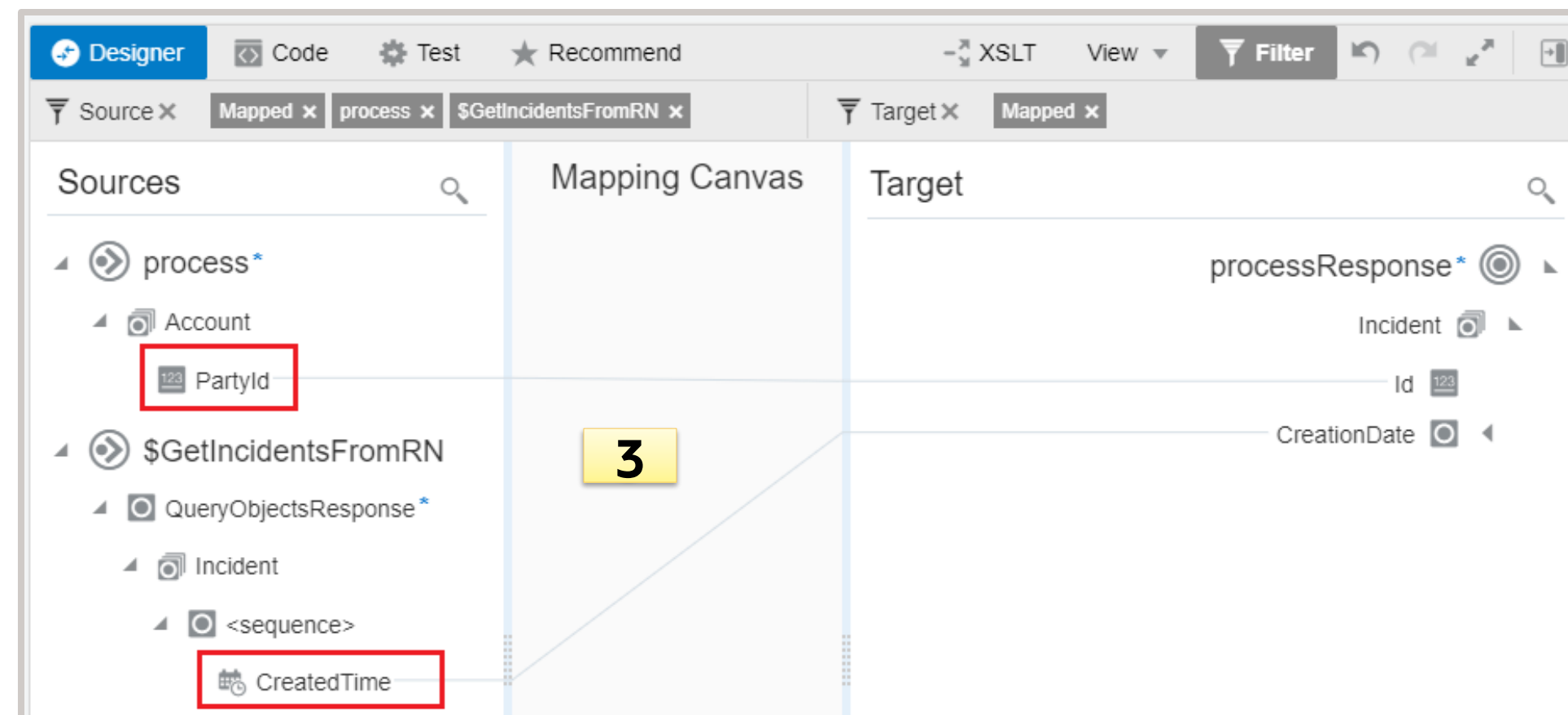
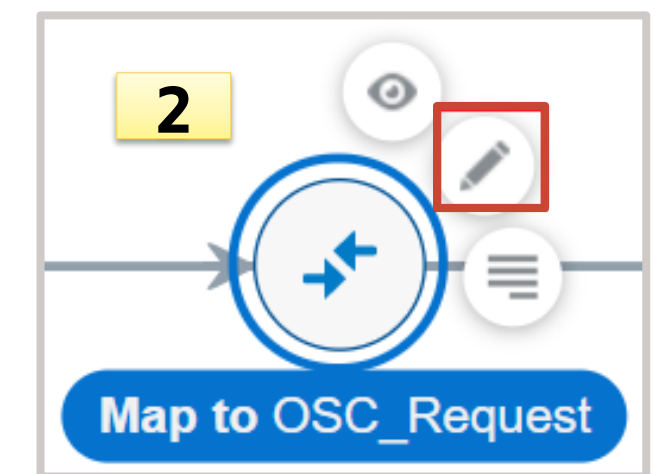
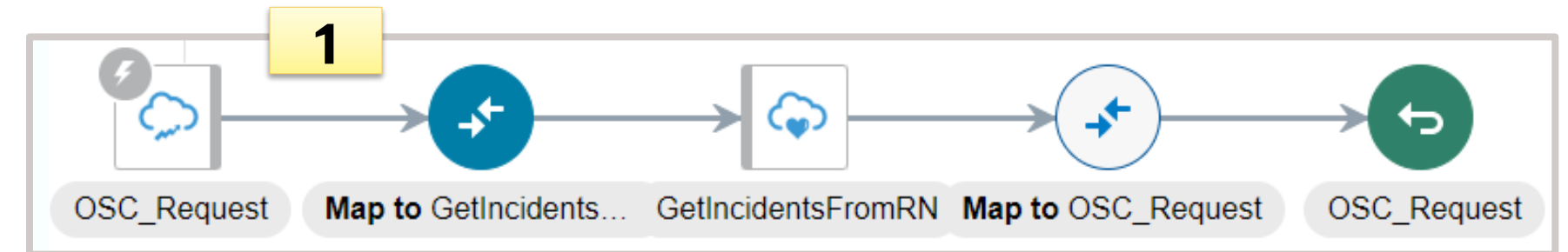
- Some mappings can be done faster using an external tool.
 - “If” and “choose-when-otherwise” conditionals
 - Large sections of similar/repeating elements
- More advanced functions and capabilities (*not supported in the OIC Mapper*) can be implemented.
 - Variables
 - Advanced selectors
 - Templates



Preparing for XSL Export

Prior to export, complete the following:

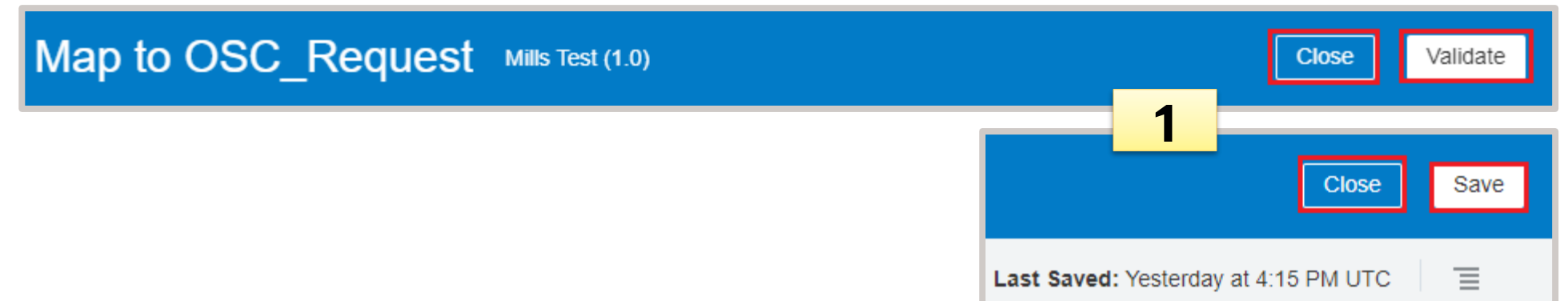
1. Design the integration flow logic.
2. Open the “empty” Data Mapper.
3. Map one or more data values from each source schema to the target schema. *This provides visibility for all required schemas in the exported XSL.*



Exporting the XSL File

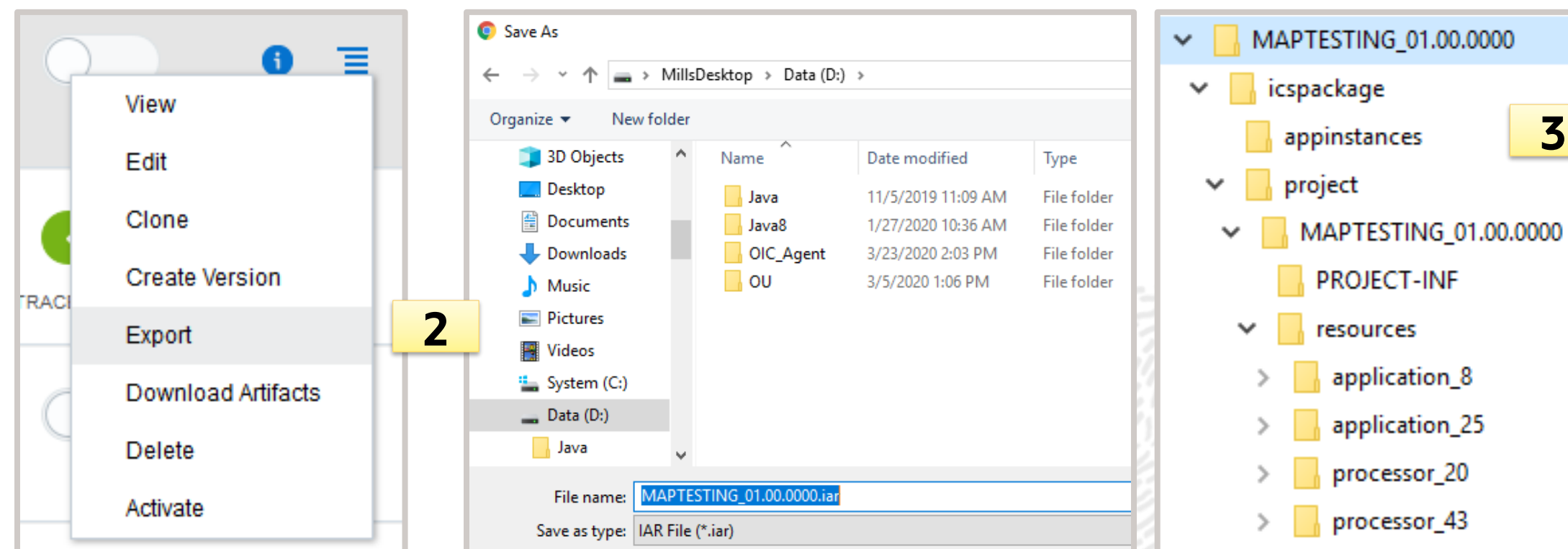
1. Prior to export, complete the following:

- Validate and then close the mapper.
- Save and then close the integration.



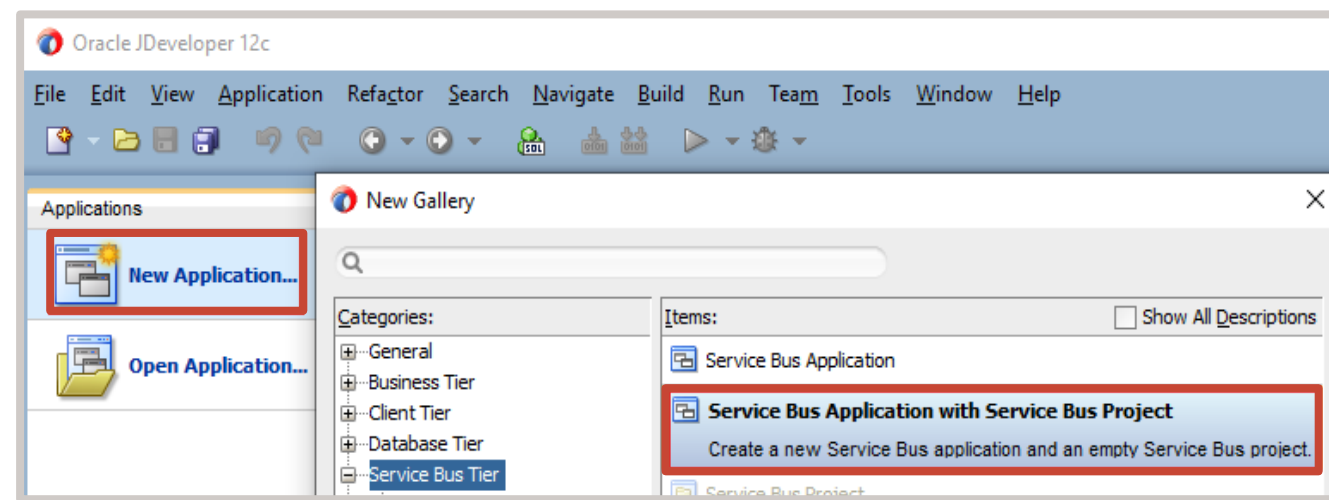
2. Export the entire integration.

3. Extract the .iar archive (if you wish to manually view the file).

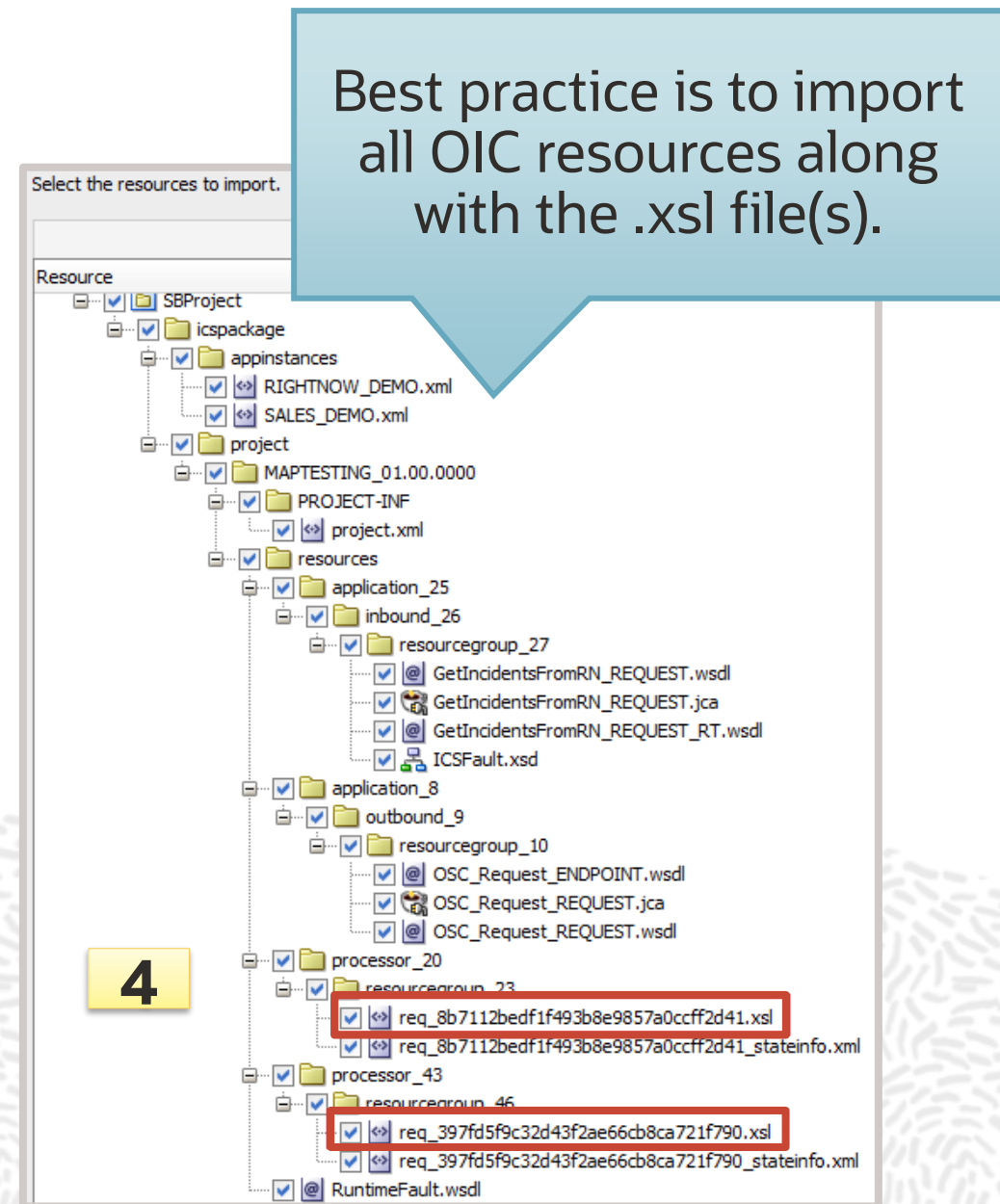
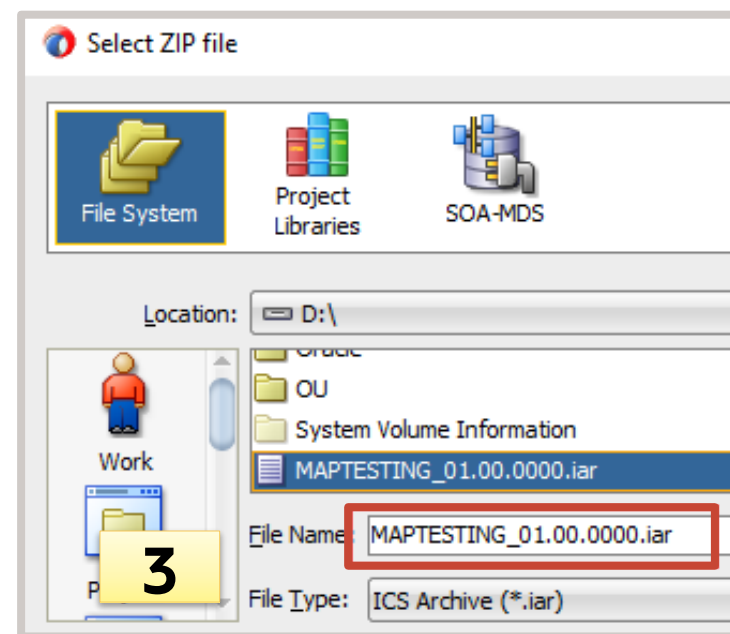
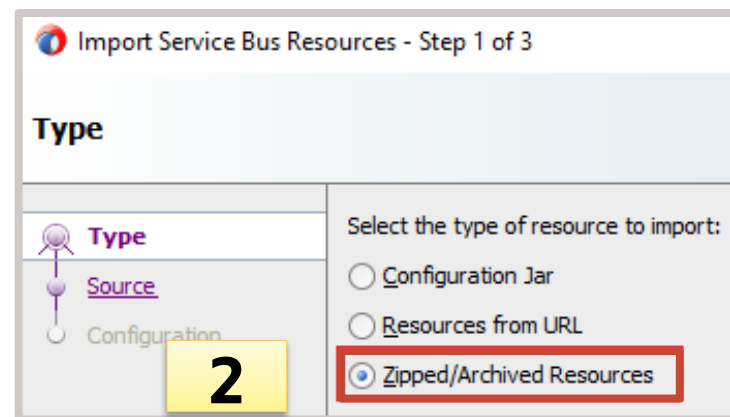
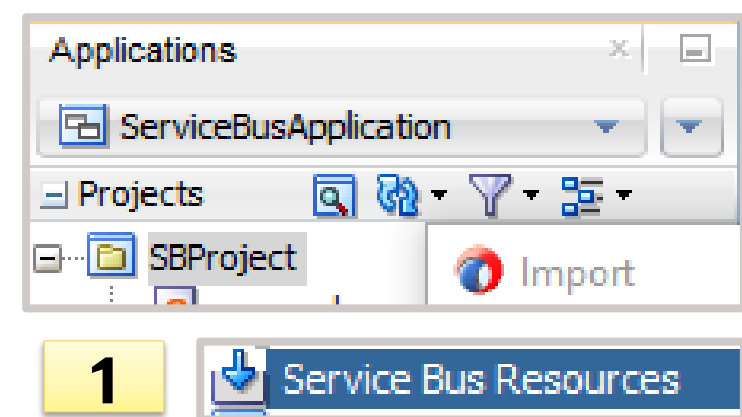


Setting Up JDeveloper to Edit the XSL File

- Create an Oracle Service Bus application and project in Oracle JDeveloper.



- Import the integration archive to the OSB project.



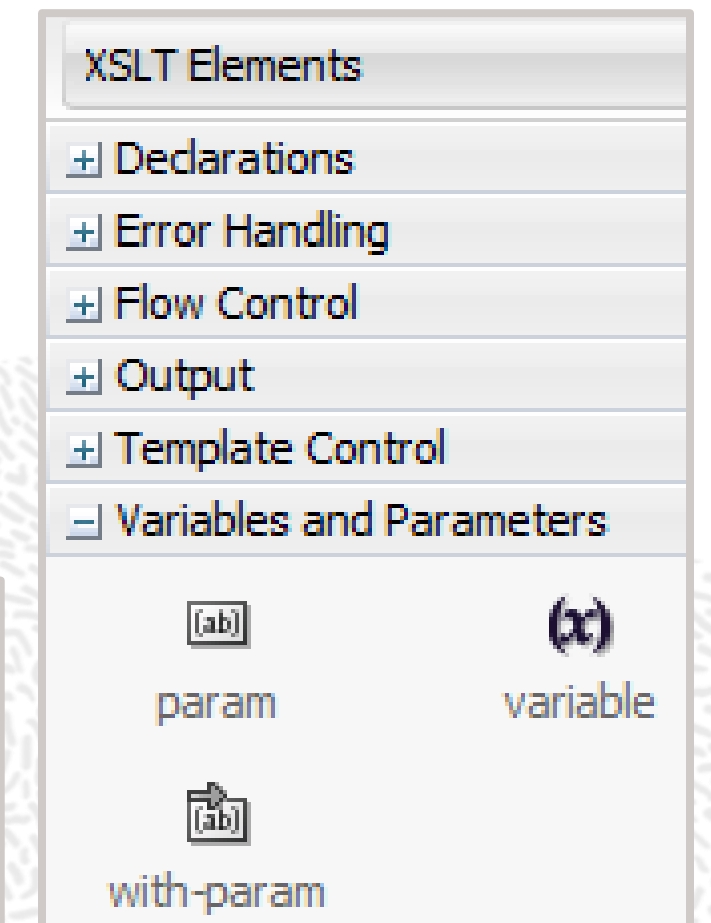
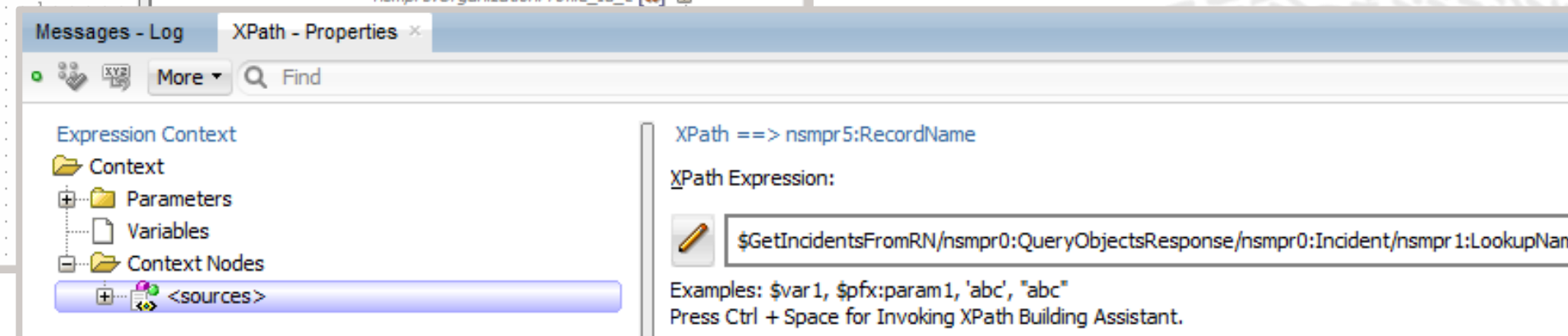
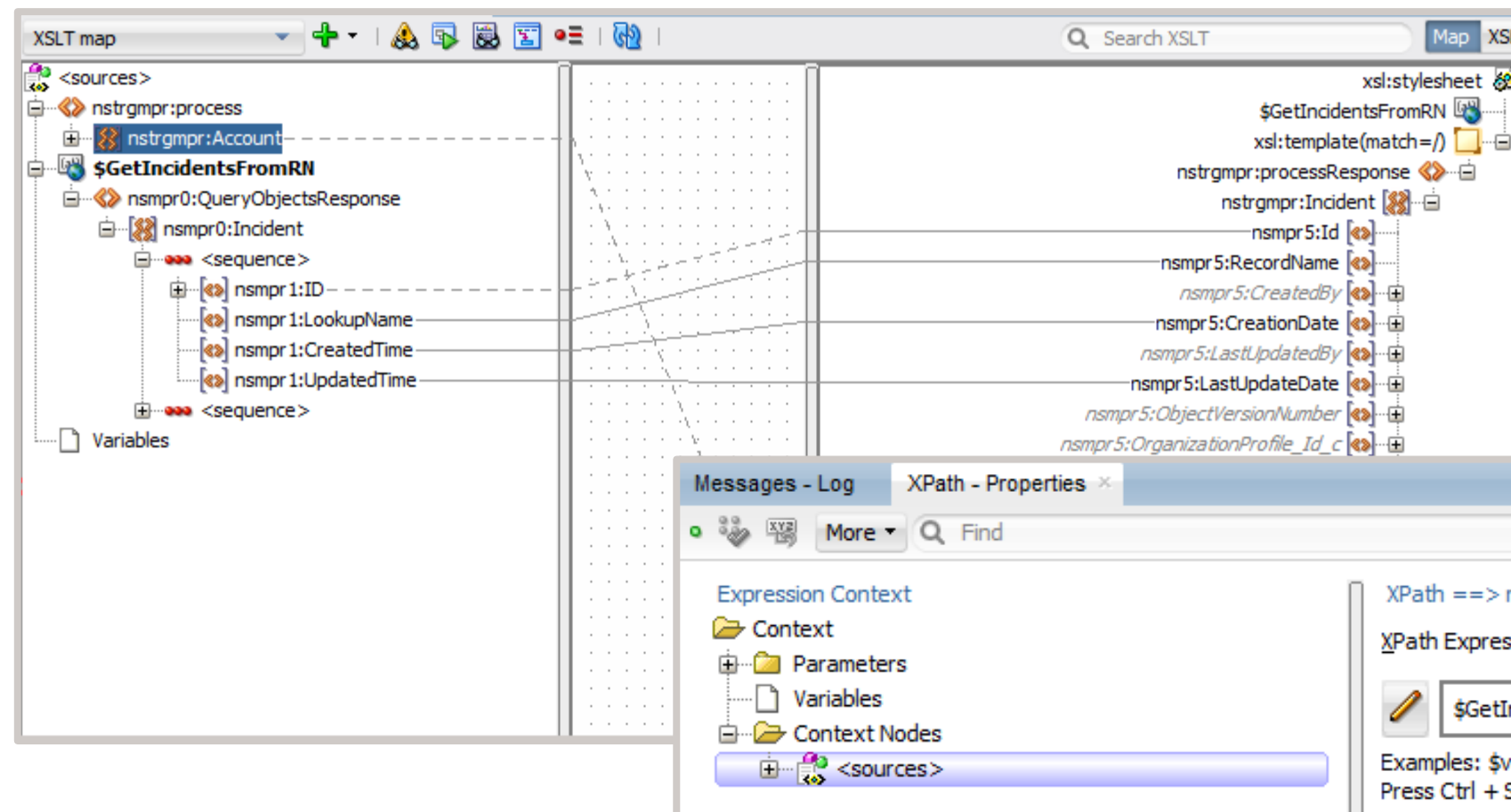
Using the XSLT Editor

Locate and open to edit the .xsl file:

- Use the **Design** editor or edit the **Source** directly.
- Switch between the **Map** view and the **XSLT** view.
- Leverage any of the XSLT elements.

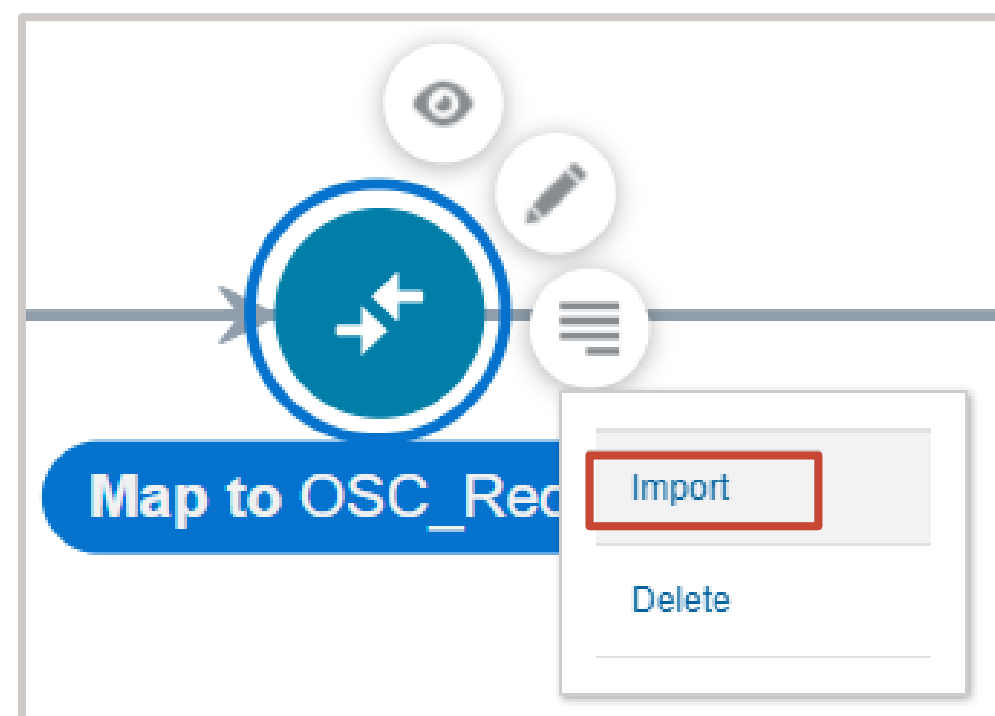
```

<nsmpr5:CreationDate xml:id="id_29">
  <xsl:value-of select="$GetIncidentsFromRN/nsmpr0:QueryObjectsResponse/nsmpr0:1
</nsmpr5:CreationDate>
<nsmpr5:LastUpdateDate xml:id="id_27">
  <xsl:value-of select="$GetIncidentsFromRN/nsmpr0:QueryObjectsResponse/nsmpr0:1
</nsmpr5:LastUpdateDate>
<nsmpr5:RequestId xml:id="id_48">
  <xsl:value-of select="/nstrgmpr:process/nstrgmpr:Account/nsmpr7:OwnerPartyId"
</nsmpr5:RequestId>
</nstrgmpr:Incident>
</nstrgmpr:processResponse>
</xsl:template>
</xsl:stylesheet>
  
```

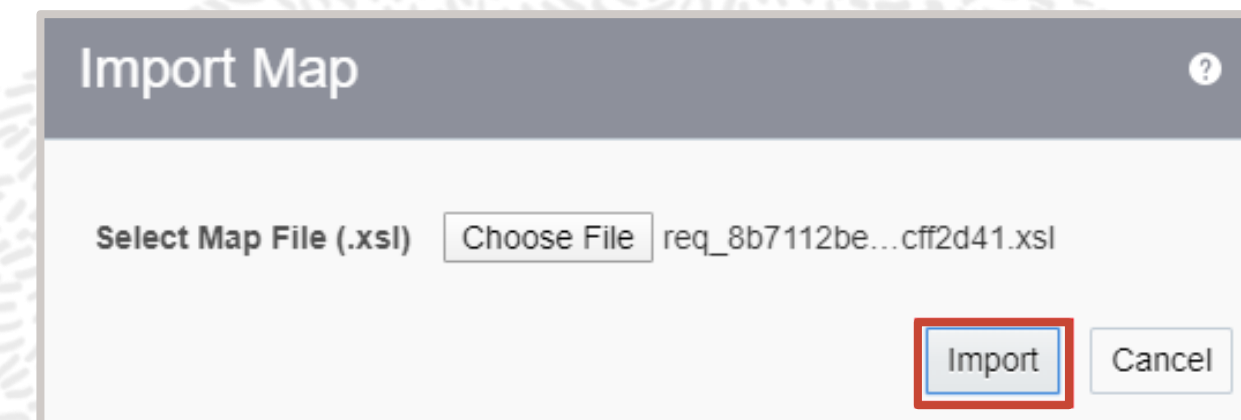
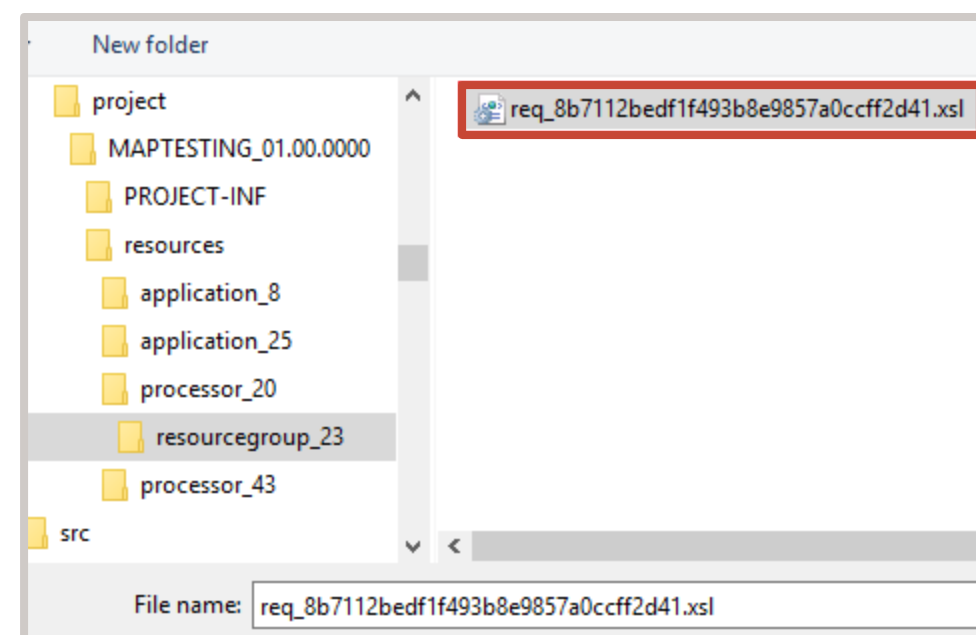
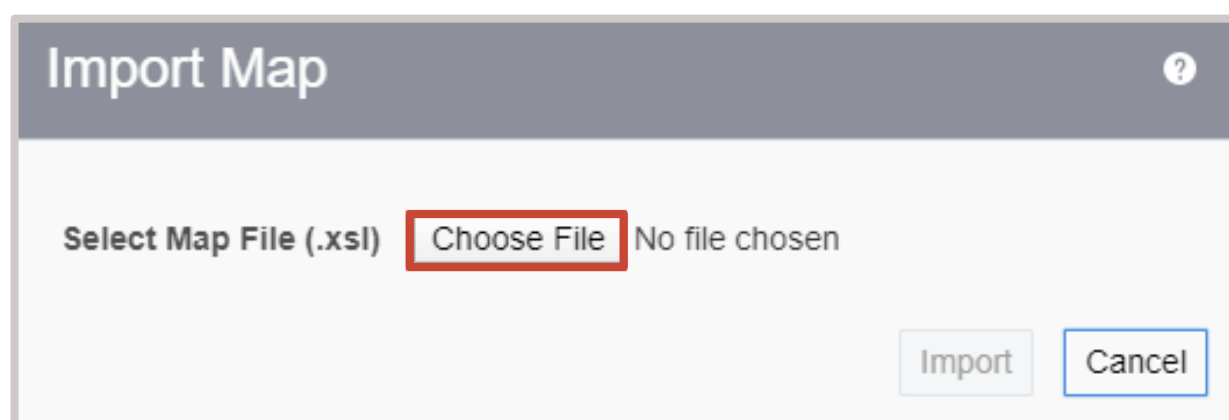


Import the Edited XSL File into OIC

- Save the edited transformation.
- Return to the OIC Map action and click **Import**.



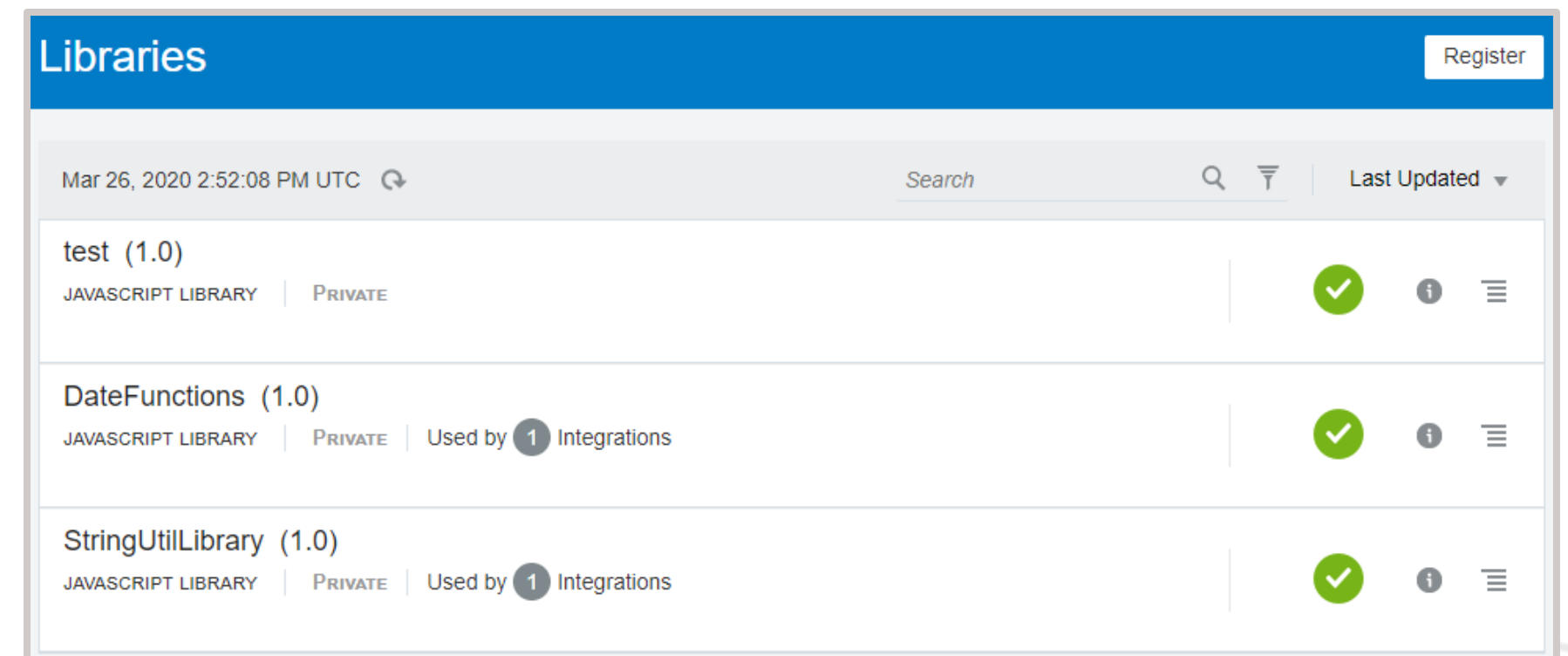
The mapper is now locked
(see notes section).



Using Custom JavaScript Functions

More advanced transformation requirements can be facilitated by using custom functions written in JavaScript. Two options are offered in OIC:

- Register as a custom XPath function
 - Added to the list of XPath functions in a **User Defined** folder
 - Available within any XSLT statement
- Register as a JavaScript function call
 - Available within any Orchestration integration flow



Adding JavaScript Libraries

1. Create new custom JavaScript functions
 - Save in one or more separate .js files
2. Register a new library in OIC
 - Upload a **.js** file or a **.jar** of multiple files
3. Define each function within the library
 - Classify as **Orchestration** or **XPath**
 - Define data types for **Input** and **Output** parameters
 - Optionally define parameter descriptions

```
function parseDate(myDate){
    var parts, date, time, dt, ms;

    parts = myDate.split(/[T ]/); // Split on `T` to get date and time
    date = parts[0];
    time = parts[1];

    dt = new Date();

    parts = date.split(/[-\/]/); // Split date on - or /
    dt.setFullYear(parseInt(parts[0], 10));
    dt.setMonth(parseInt(parts[1], 10) - 1); // Months start at 0 in JS
    dt.setDate(parseInt(parts[2], 10));

    parts = time.split(/[:]/); // Split time on :
    dt.setHours(parseInt(parts[0], 10));
    dt.setMinutes(parseInt(parts[1], 10));
    dt.setSeconds(parseInt(parts[2], 10));

    ms = dt.getTime();
    return ms;
}
```

ORACLE Integration

Libraries

Mar 25, 2020 10:34:17 PM UTC

StringUtilLibrary (1.0)

JAVASCRIPT LIBRARY | PRIVATE | Used by 2 Integrations

Register Library

* Select Library File (.js/.jar) **Choose File** No file chosen

* Name Enter Library name

* Identifier ENTER LIBRARY IDENTIFIER

* Version 01.00.0000

Description Enter a brief description...

Javascript Library

Functions

Search

DateFunctions.js 2

getDurationBetweenDates ✓

parseDate ✓

parseDate

File: DateFunctions.js

Classification Type

☒ Orchestration ☐ XPath

Input

myDate String abc

Boolean

Number

String

Output

ms String abc

Using Custom XPath Functions in the OIC Mapper

- Locate the custom function and drag and drop onto the target element.
- Provide source arguments as required.

The screenshot displays the OIC Mapper interface with four main panes: Sources, Mapping Canvas, Target, and Components.

- Sources:** A tree view showing a 'process' source with an 'Account' component. Under 'Account', there are several fields: 'PartyId', 'PartyNumber', 'SourceSystem', 'SourceSystemReferenceValue', 'OrganizationName', 'UniqueNameSuffix', and 'PartyUniqueName'.
- Mapping Canvas:** A central workspace where a mapping is being created. A blue line connects the 'OrganizationName' field from the Sources pane to a function icon (a blue square with a white 'f') on the canvas.
- Target:** A tree view showing a 'QueryObjects' target with an 'Incident' component. Under 'Incident', there are several fields: 'ID', 'CreatedTime', 'UpdatedTime', 'Asset', 'AssignedTo', 'Banner', and 'BilledMinutes'.
- Components:** A sidebar on the right containing a list of functions. The 'Functions' category is expanded, showing a list of functions including 'Boolean', 'Conversion', 'Date', 'Integration Cloud', 'Mathematical', 'Node-set', 'String', and 'User Defined'. The 'User Defined' category is further expanded, showing a list of functions including 'getElementByXPath' and 'parseDate'.

Red boxes highlight the 'LookupName' field in the Target pane and the 'getElementByXPath' function in the Components pane. A red arrow points from the 'getElementByXPath' function to the 'LookupName' field. A blue line also points from the 'OrganizationName' field in the Sources pane to the function icon on the Mapping Canvas.

At the bottom of the interface, there is a text area labeled 'Expression for: LookupName' containing the following XPath expression:

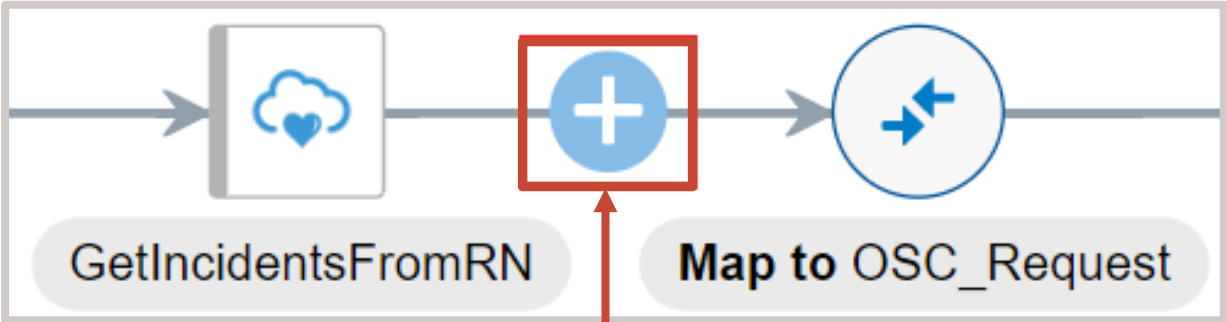
```
orajs0:getElementByXPath ( /nssrcmpr:process/nssrcmpr:Account/ns33:OrganizationName)
```

All library functions that have been classified as *XPath* will appear in the **User Defined** category.

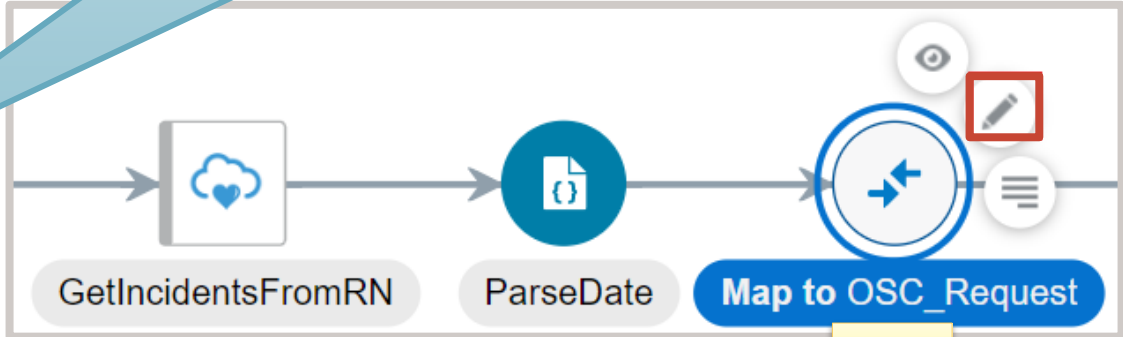
Executing Orchestration JavaScript Functions

- 1. Drag the **Javascript** Call action to the integration flow.
- 2. Select the custom function.
- 3. Use the Expression Builder to define parameter values.
- 4. The return value is now available as a Source data element.

All Library functions that have been classified as **Orchestration** will appear in the list of callable functions.



The 'Select a Function' dialog shows a search bar and a list of functions. The function 'getDurationBetweenDates' is selected, and a red box highlights the 'Select' button. A yellow box with the number '2' is below the dialog.



The 'Call' dialog shows three options: 'Integration', 'Javascript', and 'Process'. The 'Javascript' option is selected, and a red box highlights it. A yellow box with the number '3' is below the dialog.

The configuration for the 'getDurationBetweenDates' function is shown. It includes the 'Javascript Library' (DateFunctions - 1.0), 'Output Parameter' (duration (String)), and 'Output Description' (No description). The 'Input Parameters' section shows a table with two parameters: 'dateStr1' and 'dateStr2'. The 'dateStr1' parameter is set to 'ExistingCustomerFlagLastUpdateDate', and the 'dateStr2' parameter is set to 'Add an expression'. A red box highlights the 'dateStr1' value, and a yellow box with the number '3' is below the dialog.

The 'Sources' list shows the available data elements. The element '\$ParseDate' is highlighted with a red box, and a yellow box with the number '4' is below it.



Agenda

- OIC Data Mapper
- OIC Recommendations Engine
- Advanced Transformation Options
- Using OIC Lookups



The Challenge



Information	System A	System B
Organization identifier	ID	OrgID
Organization name	Name	OrgName
...



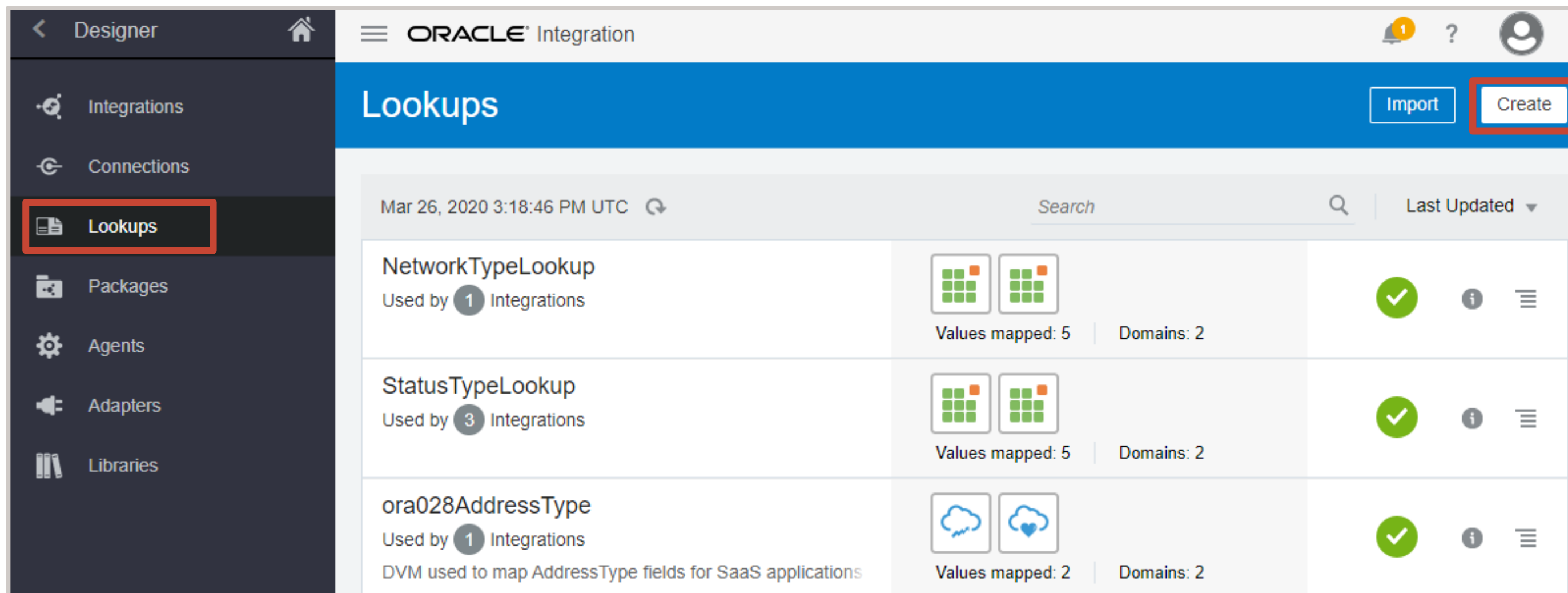
OIC Lookups

- Use lookups to create in-memory “tables” that map the different terms used to describe the same item across your applications.
- Lookups are:
 - Reusable
 - Based on static definitions
 - Values are specified at design time.
 - Value map tables are looked up for values at run time.
- Example: Country Codes



Getting Started

- On the toolbar, click **Designer**.
- On the Designer Portal, click **Lookups**.
- Click **Create**.



Creating Lookups

- Select the adapter type of the on-premises or SaaS application.
 - Alternatively, provide a unique domain name for the application or data format type.
- Add values to each field.

Example

Lookup

Create a lookup to associate the different values used by your application.

	Domain Name 1	Domain Name 2	
	Replace with Adapter		
	Edit Domain Name	(Add a Value)	

Create a lookup to associate the different values used by your applications. When you create an integration, you can use the lookup to map values from the application to the target application.

	CustomApp_1	Domain Name 2	
	(Add a Value)		

Edit Domain Name

Edit Domain Name for the application you are mapping

* Domain Name

CustomApp_2

OK

Select Adapter

All

Search

Click "Select" for the application you are mapping.

	Oracle E-Business Suite	Select
	Oracle Engagement Cloud	Select
	Oracle Enterprise Performance Management Cloud	Select
	Oracle ERP Cloud	Select
	Oracle Field Service Cloud	Select
	Oracle HCM Cloud	Select

Oracle Engagement Cloud

Oracle Service Cloud (RightNow)

	1	Billing	
	2	Shipping	

CountryCodes

Lookup

Create a lookup to associate the different values used by your applications. When you create an integration, you can use the lookup to map values from the application to the target application.

	Name	Alpha-2	Alpha-3	
	Australia	AU	AUS	
	Belize	BZ	BLZ	
	Canada	CA	CAN	

Click to add another row



Editing Lookups

- Lookups can be cloned or exported to a CSV file for easy reuse.
- CSV files can be edited and then imported back to OIC.

NEW CountryCodes

Common mapping for country names to 2-digit or 3-digit country codes

Values mapped: 3 Domains: 3

✓

i

☰

Export To CSV

	A	B	C
1	DVM	CountryCodes	Common mapping for country names to 2-digit or 3-digit country codes
2	Name	Alpha-2	Alpha-3
3	Australia	AU	AUS
4	Belize	BZ	BLZ
5	Canada	CA	CAN

3 rows

Edit CSV

	A	B	C
1	DVM	CountryCodes	Common mapping for country names to 2-digit or 3-digit country codes
2	Name	Alpha-2	Alpha-3
3	Algeria	DZ	DZA
4	American Samoa	AS	ASM
5	Andorra	AD	AND
6	Angola	AO	AGO
7	Anguilla	AI	AIA
8	Antarctica	AQ	ATA
9	Antigua and Barbuda	AG	ATG
10	Argentina	AR	ARG
11	Armenia	AM	ARM
12	Aruba	AW	ABW
13	Australia	AU	AUS
14	Austria	AT	AUT
15	Azerbaijan	AZ	AZE
16	Bahamas	BS	BHS

246 rows

Lookups

Mar 26, 2020 4:04:08 PM UTC CountryCodes

Lookup Name contains CountryCodes

NEW CountryCodes

Common mapping for country names to 2-digit or 3-digit country codes

Values mapped: 3 Domains: 3

✓

i

☰

Import

Create

Replace Existing Lookup?

CountryCodes

A lookup with the same name already exists. It will be replaced when the new lookup CountryCodes is imported.

Import and Replace

Cancel

NEW CountryCodes

Common mapping for country names to 2-digit or 3-digit country codes

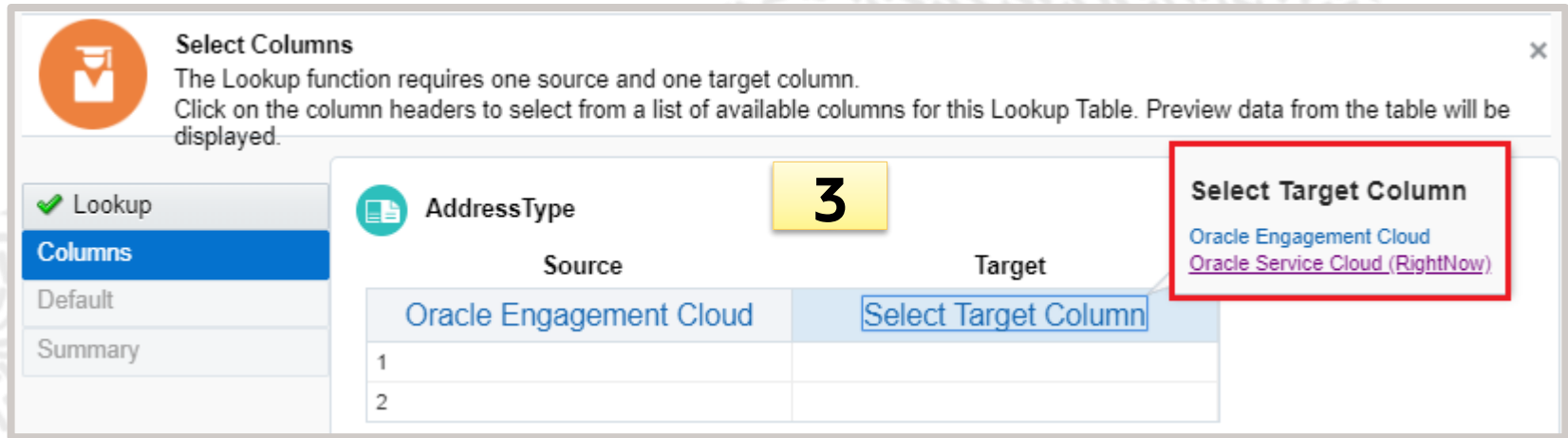
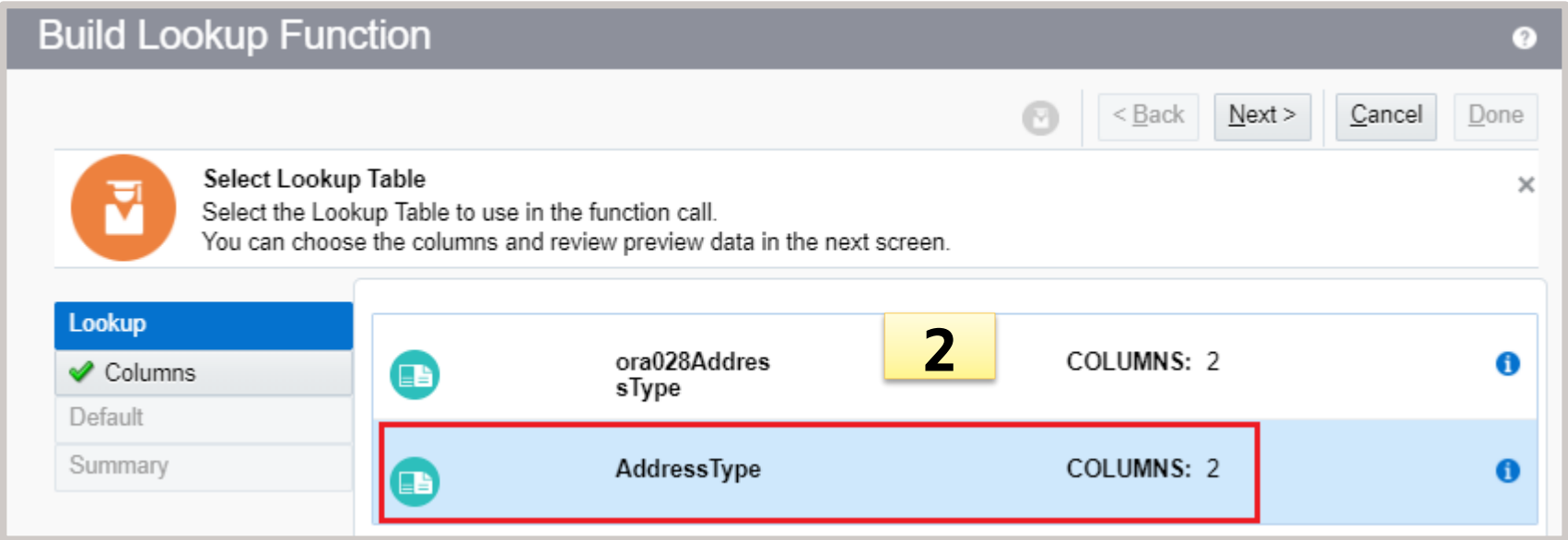
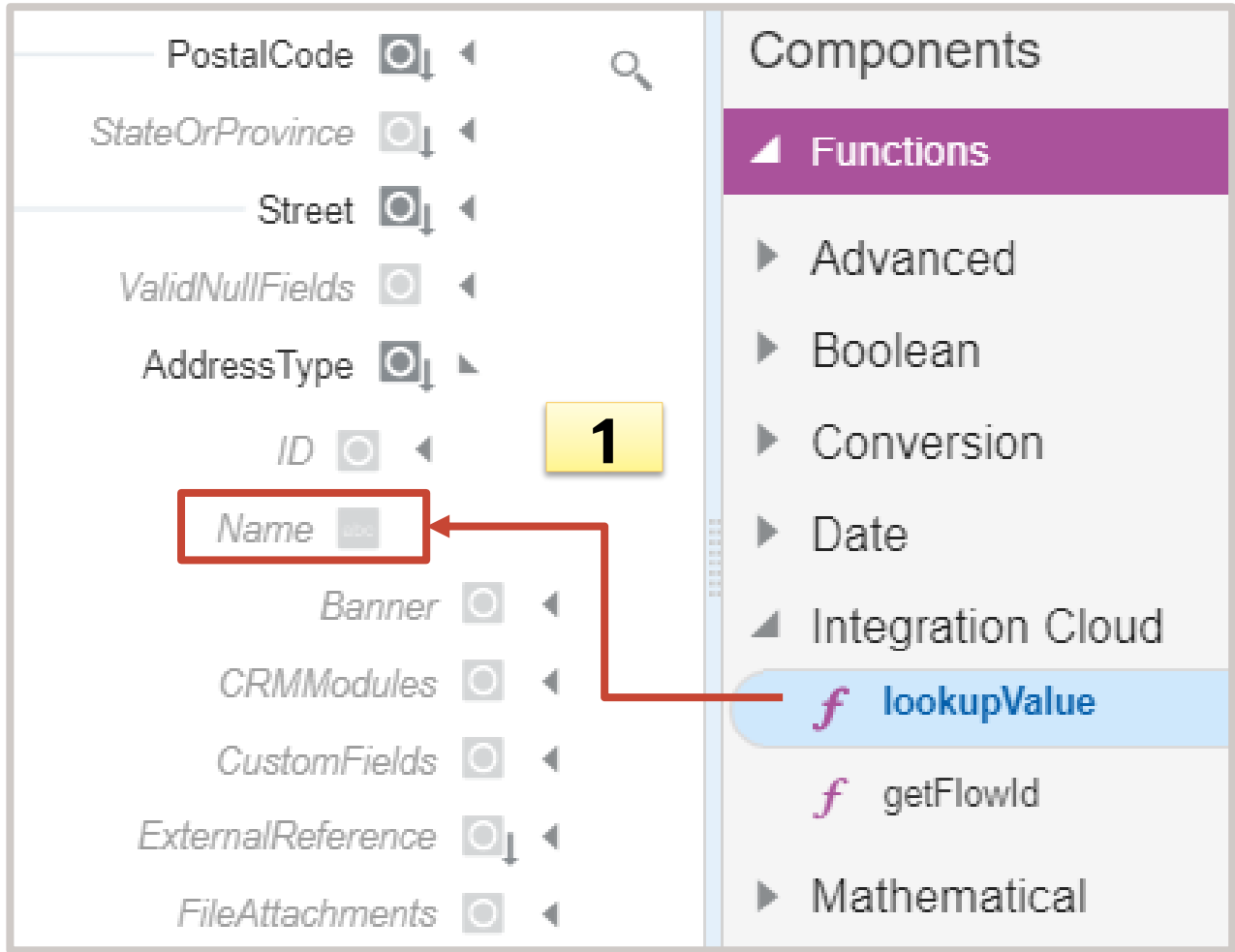
Values mapped: 246 Domains: 3

Changes made to lookups already used in active integrations take effect immediately (*within 2 minutes*). There is **no** need to re-activate integrations using a changed lookup.



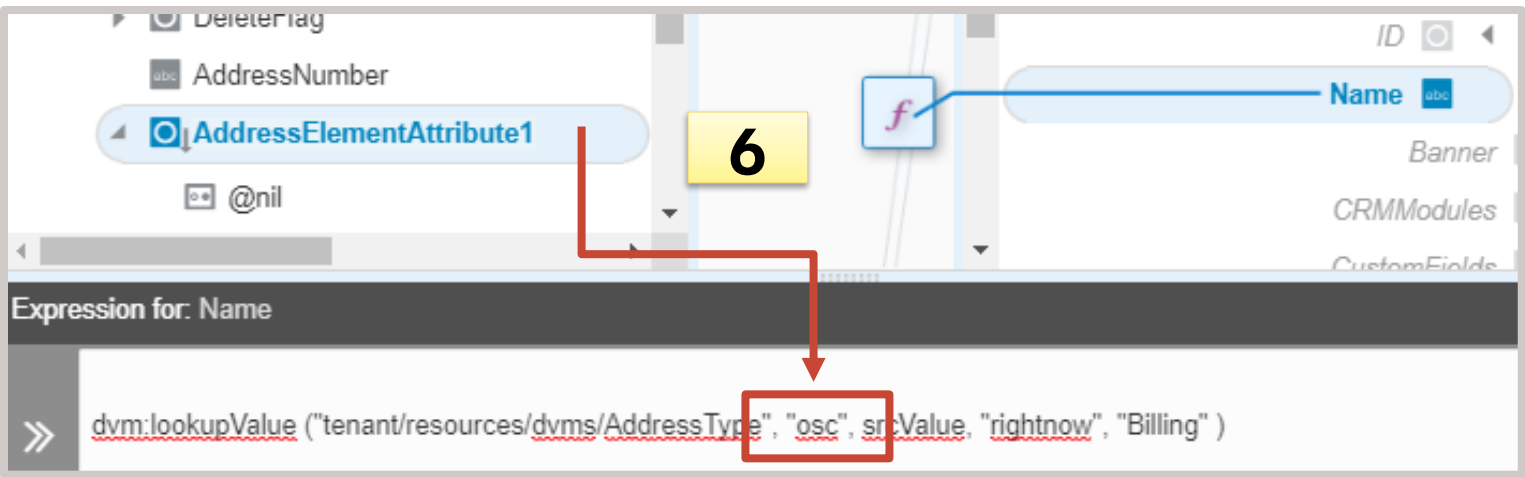
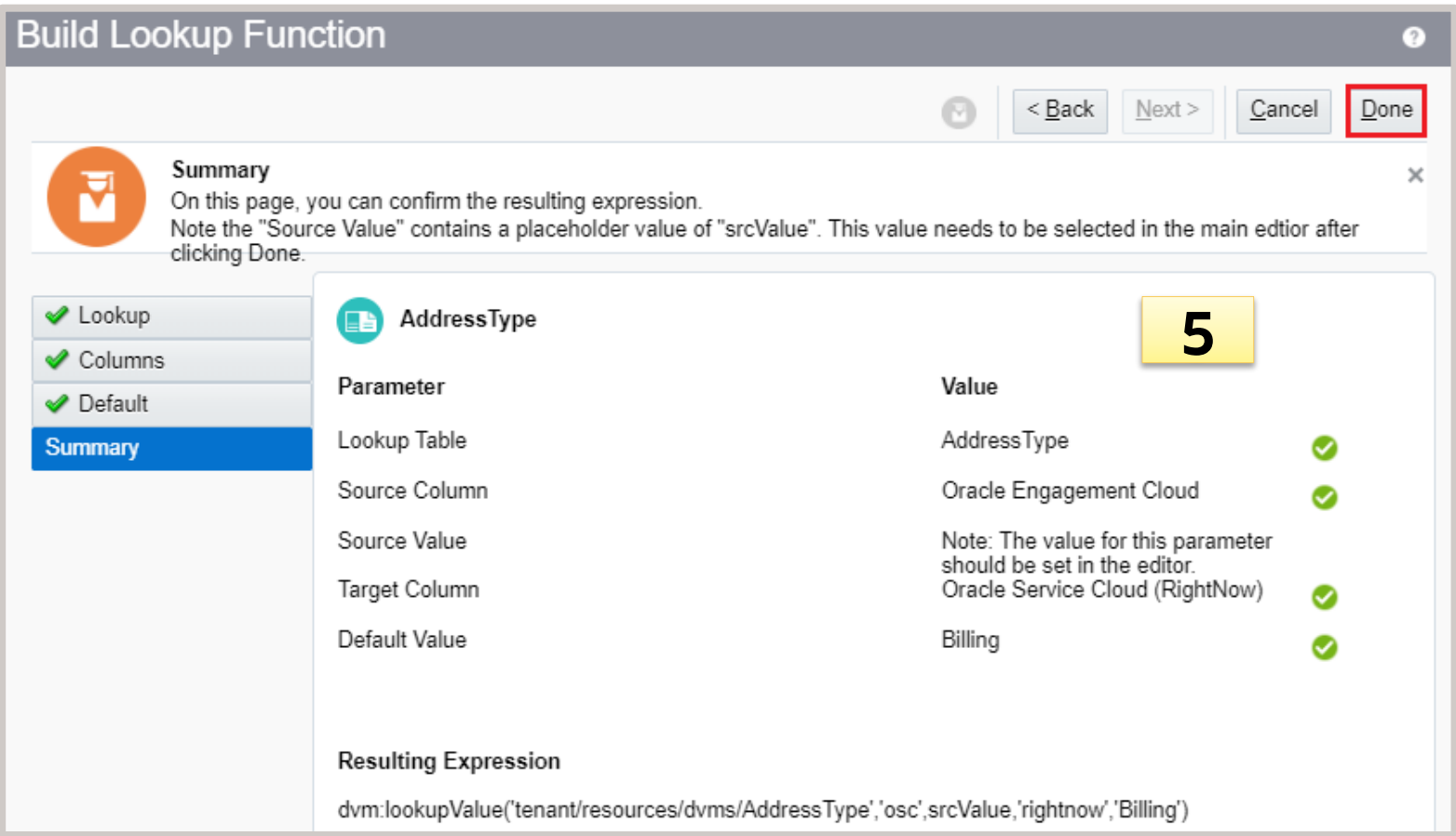
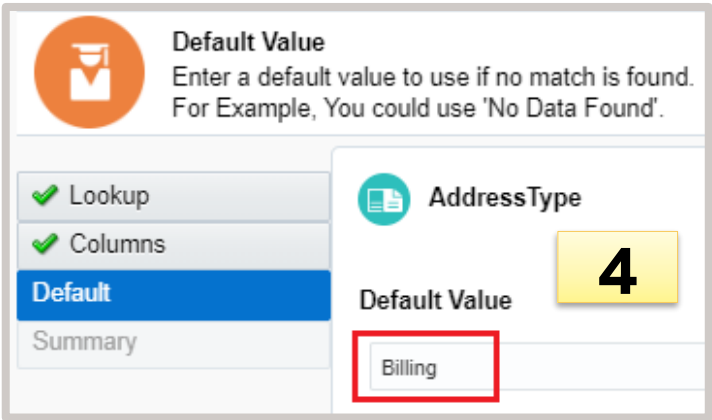
Referencing Lookups in the OIC Data Mapper

1. Locate the **lookupValue** function and drag and drop onto the target element.
 - This will launch the **Build Lookup Function** wizard.
2. Select the lookup and click *Next*.
3. Choose the **Source** and **Target** Columns.

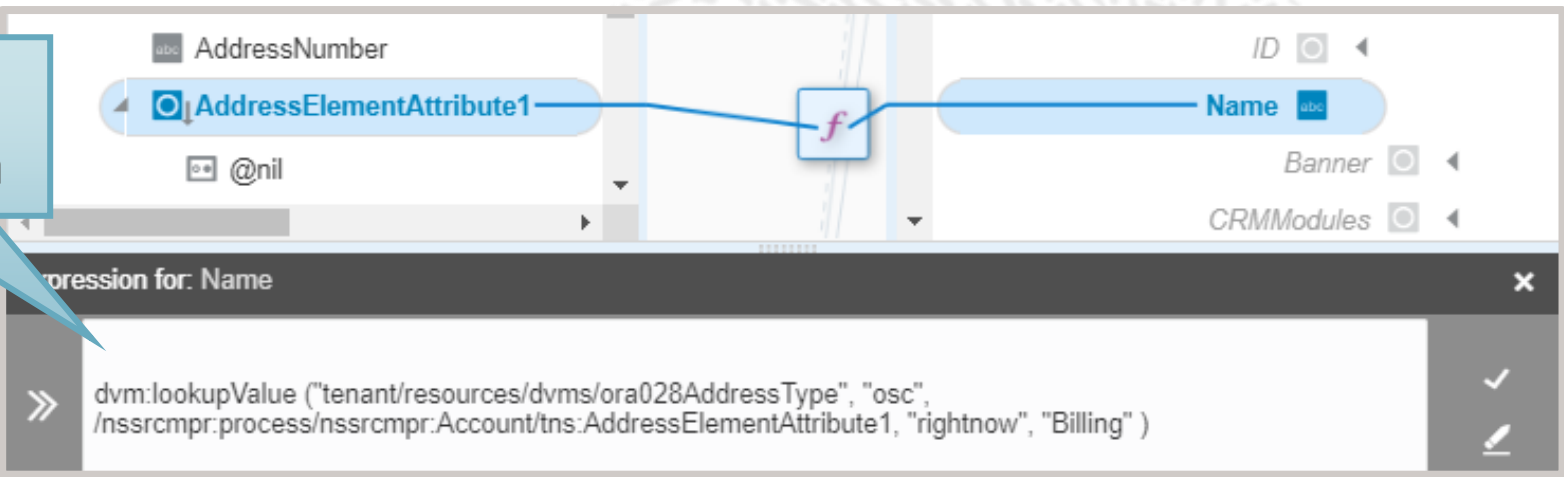


Referencing Lookups in the OIC Data Mapper

- 4. Provide a default value.
- 5. Review the Summary page and click *Done*.
- 6. Locate and map the Source value to the third parameter and click the *Save* check mark.



Final expression



Summary

In this lesson, you should have learned how to:

- Map or transform data by using the OIC Mapper tool
- Create XSL expressions using XSLT functions and operators
- Describe and use the OIC Recommendations Tool
- Edit and import advanced XSL files into OIC
- Register and use custom JavaScript functions
- Create and invoke OIC Lookups



Practice 8-1: Creating Pub/Sub Integrations with Lookups

This practice includes:

- PART 1 – Creating a Lookup Component
- PART 2 – Creating the Publish Integration Flow
- PART 3 – Creating the Subscribe Integration Flow
 - Configuring Data Mapping using a Lookup function
- PART 4 – Testing the Publish and Subscribe Integrations



Creating a Subscriber Integration (Review)

Once the Publisher integration has been configured, you can create the Subscriber integration to process the message.

- Click the Map icon and then click the **+** Create icon to launch the OIC Data Mapper.

