talend | Connecting the Data-Driven Enterprise

Lab Guide

TDM Essentials

Version 6.1

Talend Inc.
800 Bridge Parkway, Suite 200
Redwood City, CA 94065
United States
+1 (650) 539 3200

Welcome to Talend Training

Congratulations on choosing a Talend training module. Take a minute to review the following points to help you get the most from your experience.

Technical Difficulty

Instructor-Led

If you are following an instructor-led training (ILT) module, there will be periods for questions at regular intervals. However, if you need an answer in order to proceed with a particular lab, or if you encounter a situation with the software that prevents you from proceeding, don't hesitate to ask the instructor for assistance so it can be resolved quickly.

Self-Paced

If you are following a self-paced, on-demand training (ODT) module, and you need an answer in order to proceed with a particular lab, or you encounter a situation with the software that prevents you from proceeding with the training module, a Talend Support Engineer can provide assistance. Double-click the **Live Expert** icon on your desktop and follow the instructions to be placed in a queue. After a few minutes, a Support Engineer will contact you to determine your issue and help you on your way. Please be considerate of other students and only use this assistance if you are having difficulty with the training experience, not for general questions.

Exploring

Remember that you are interacting with an actual copy of the Talend software, not a simulation. Because of this, you may be tempted to perform tasks beyond the scope of the training module. Be aware that doing so can quickly derail your learning experience, leaving your project in a state that is not readily usable within the tutorial, or consuming your limited lab time before you have a chance to finish. For the best experience, stick to the tutorial steps! If you want to explore, feel free to do so with any time remaining after you've finished the tutorial (but note that you cannot receive assistance from Tech Support during such exploration).

Additional Resources

After completing this module, you may want to refer to the following additional resources to further clarify your understanding and refine and build upon the skills you have acquired:

- Talend product documentation (help.talend.com)
- Talend Forum (talendforge.org/)
- Documentation for the underlying technologies that Talend uses (such as Apache) and third-party applications that complement Talend products (such as MySQL Workbench)

This page intentionally left blank to ensure new chapters start on right (odd number) pages.

CONTENTS

LESSON 6 Using the tHMap Component

LESSON 7 Using Context Variables

LESSON 8 Using the cMap Component

LESSON 1

Creating a Simple XML to XML Map

This chapter discusses the following.

Overview

Lesson Overview

This lab is a simple exercise to learn the basics of Structure management and how to map elements between a source and a destination.

Imagine you are requested to transform an XML file into another XML file with similar structure, removing some nodes to reduce the output file size.

Objectives

After completing this lesson, you will be able to:

- Create a new Structure from an existing local XML file
- Explore a Structure and interactively display a sample of it
- Create a new Map
- Use existing Structures to specify the Input and Output areas of a Map
- Automap the elements whose names match in both the Input and Output areas of a Map
- Remove some elements from the output
- Test Run the Map and save the results to an external file, or display them in the internal viewer

Next Step

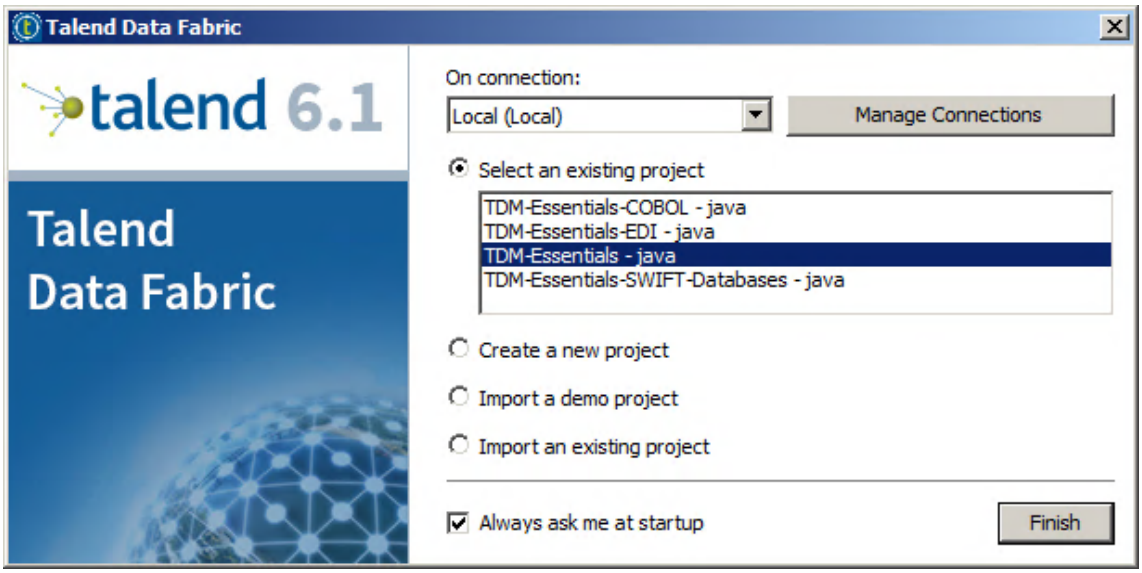
The first step is to import an existing XML structure.

Creating a Structure

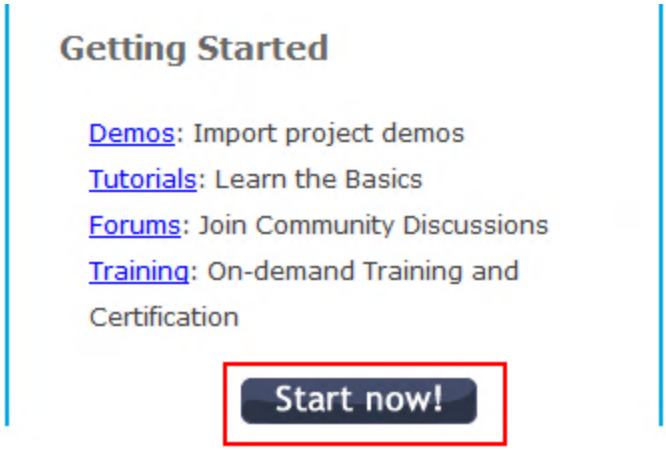
Starting Talend Studio

1. Click the **Talend Studio** link on the desktop.

2. Select the existing **TDM-Essentials** project and click **Finish**.

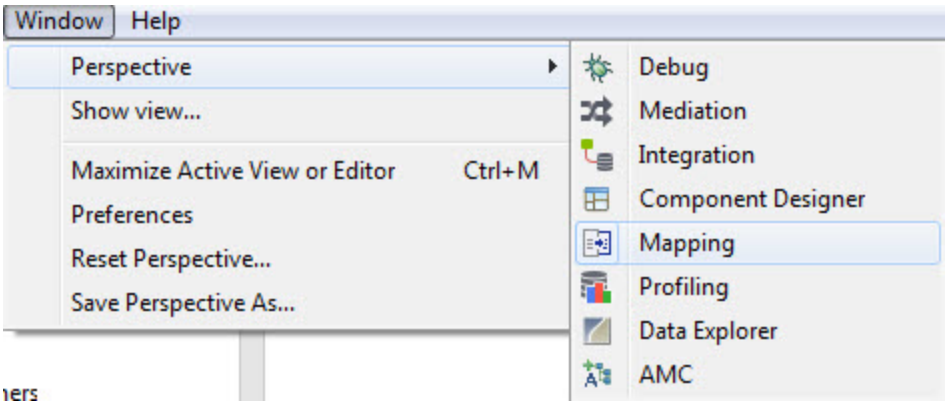


3. Click the **Start Now** button if the Welcome screen shows up. Otherwise, go to the next step.

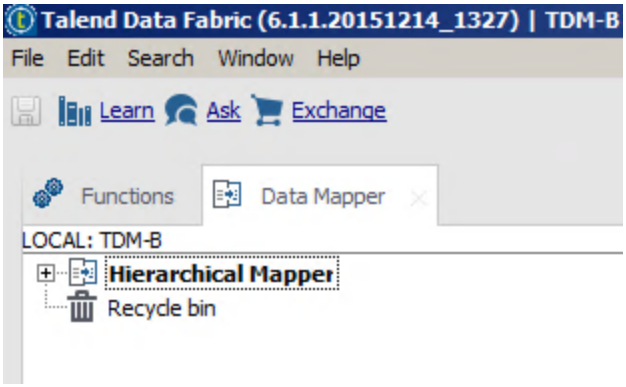


Creating a New Structure

1. Go to **Window > Perspective > Mapping** to switch to the Mapping Perspective and harness all the features of *Talend Data Mapper*.

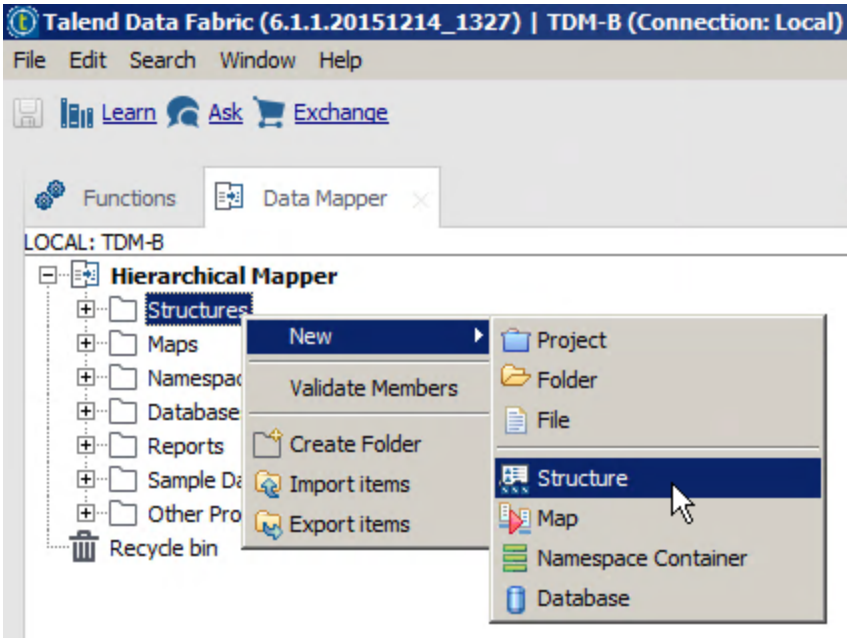


2. The **Mapping** perspective has two different views by default: **Functions** and **Data Mapper**. Select the **Data Mapper** tab.



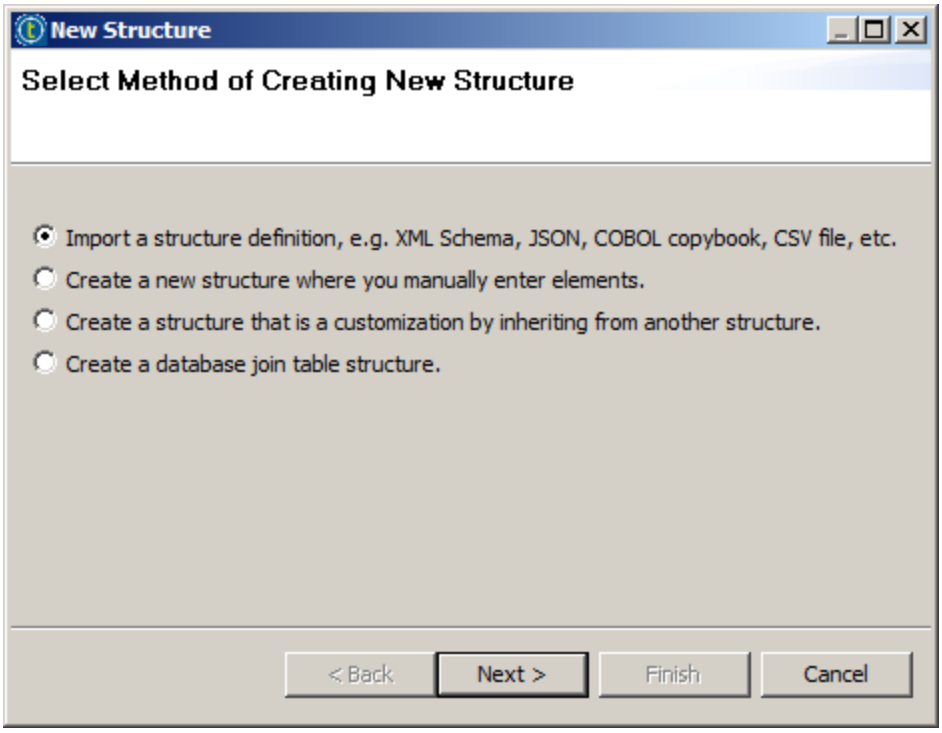
3. Now let's create a Structure that will be used later on

   In the **Data Mapper** tab, expand the **Hierarchical Mapper** node, right-click **Structures** and select **New > Structure**.



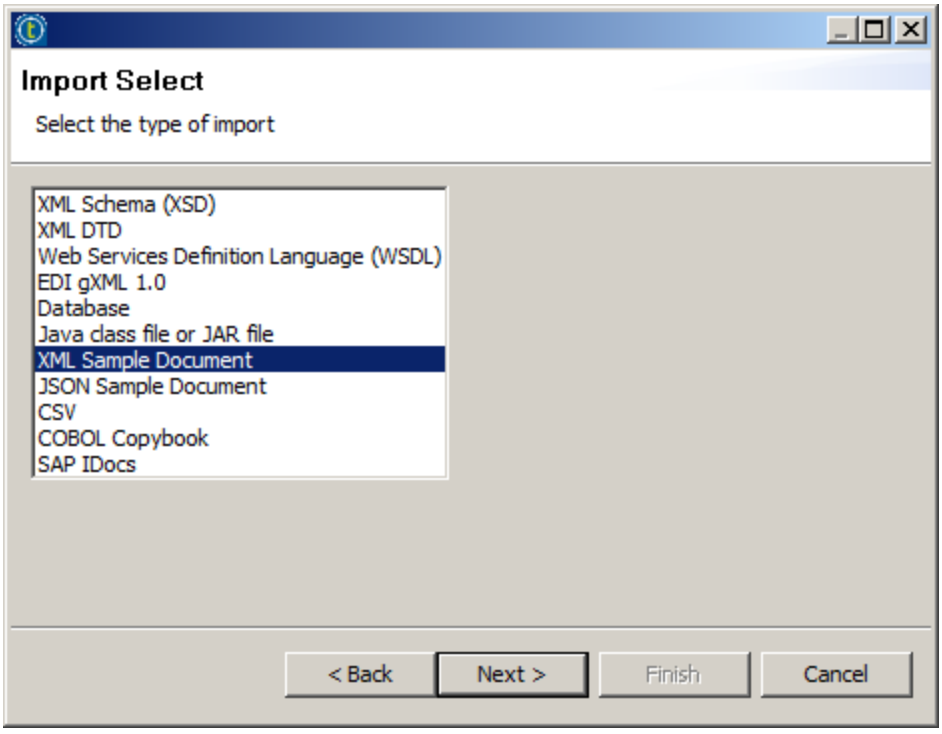
Importing a Structure Definition

1. Select **Import a structure definition, e.g. XML Schema, JSON, COBOL copybook, CSV file, etc.** to create a Structure from an existing file and click **Next >**.



Selecting the Type of Import

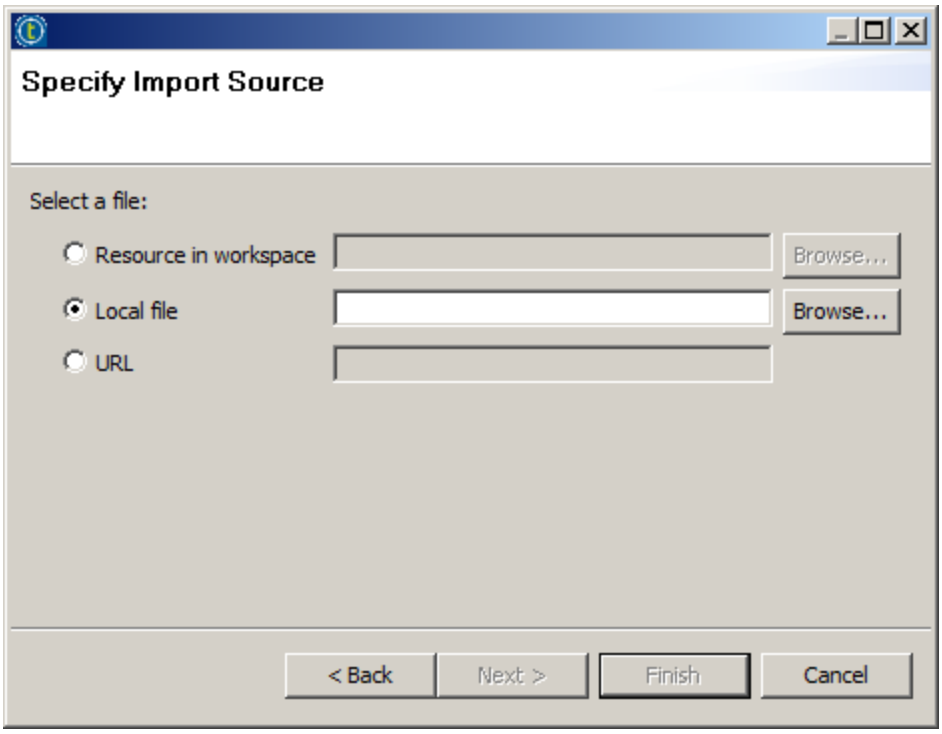
1. Click **XML Sample Document** and click **Next >**.



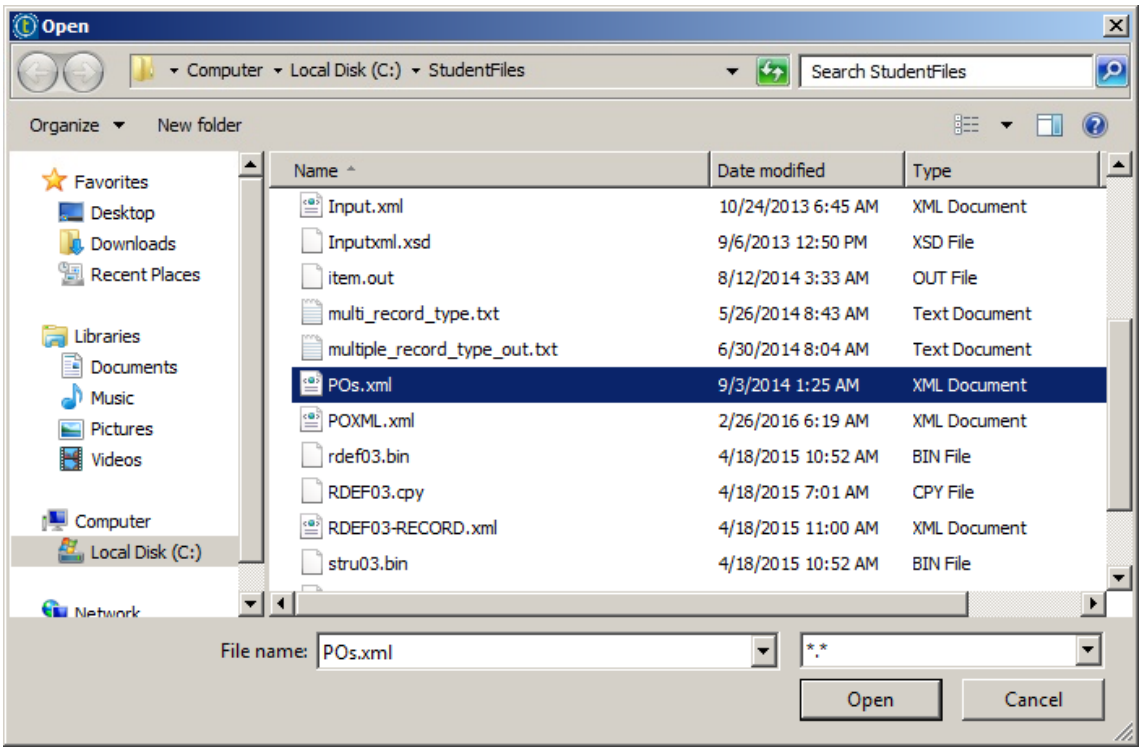
Note: When an XSD schema is available, it should be used instead of the actual XML document. An XSD schema contains the full definition of the XML document, while the XML document could implement only a part of the original schema.

Specifying the Import Source

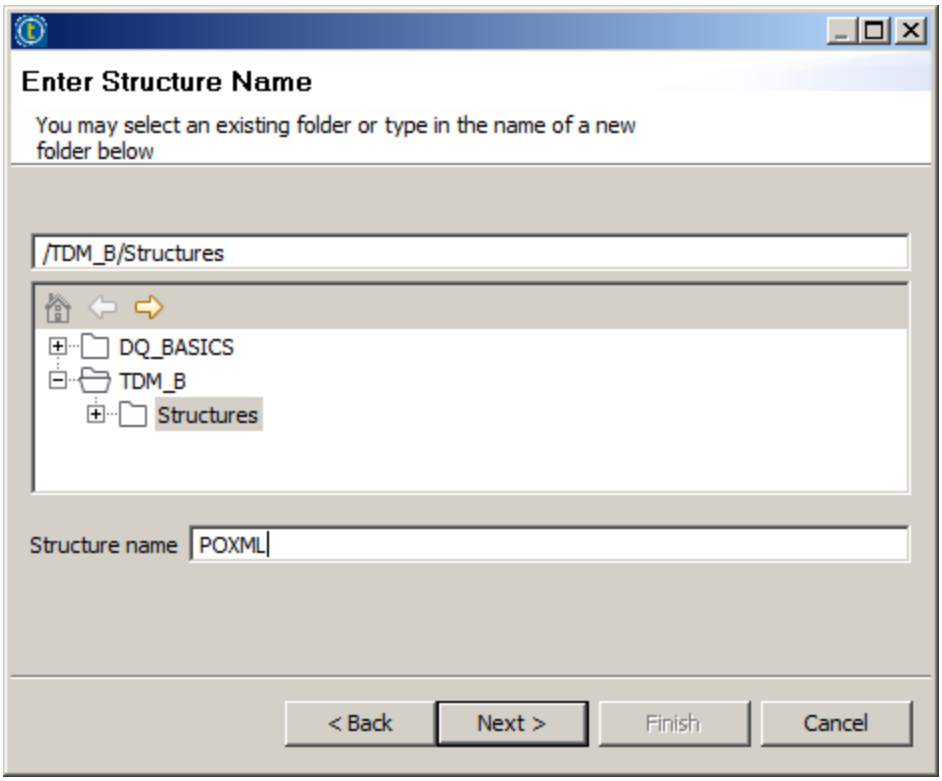
1. Select the **Local file** option and then click **Browse...**to browse to the local *C:\StudentFiles* folder where the training files are stored.



2. Select *POs.xml* then click **Open**.

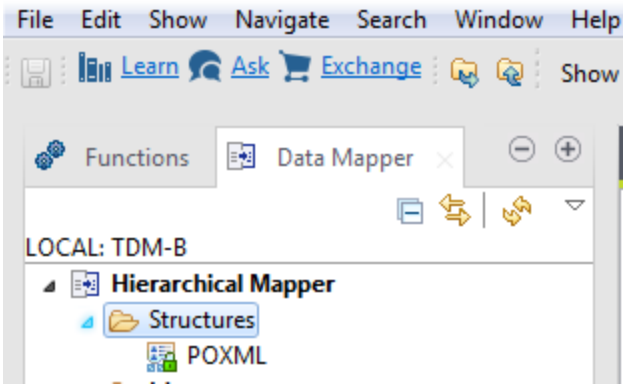


3. Click **Next >**.
4. Keep the default saving location and enter *POXML* in the **Structure name** field. Click **Next >** and then **Finish**.



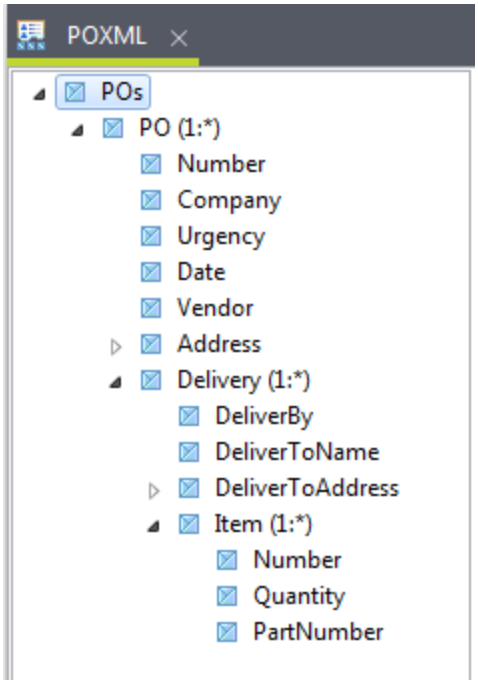
Displaying the Structure

1. Expand the **Hierarchical Mapper > Structures** node and double-click **POXML** to display the contents of the Structure:



Exploring the Structure

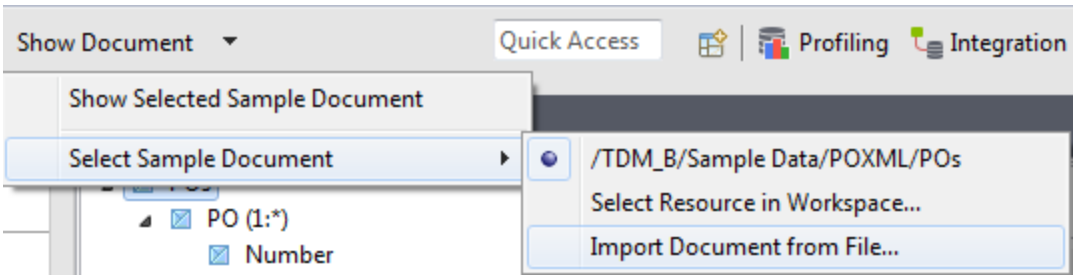
1. Explore the XML Structure by expanding and collapsing the various nodes.



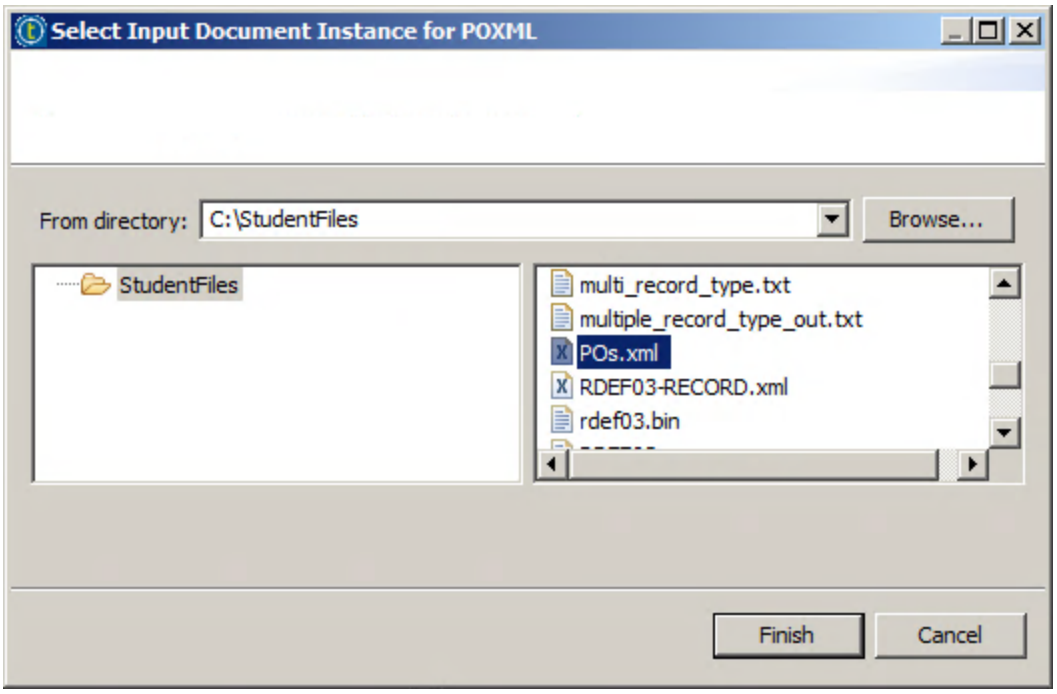
Displaying a Sample Document

Since you created the Structure from an existing XML document, the POXML Structure will use the original file as the Sample Document when displaying a preview of the data. If you had used an XSD to create the structure, or if you wanted to use a different Sample Document, you would click the downward-pointing arrow next to the **Show Document** button in the application toolbar. Then you would select an existing resource in the workspace or, as shown in step 2, import the document from a file.

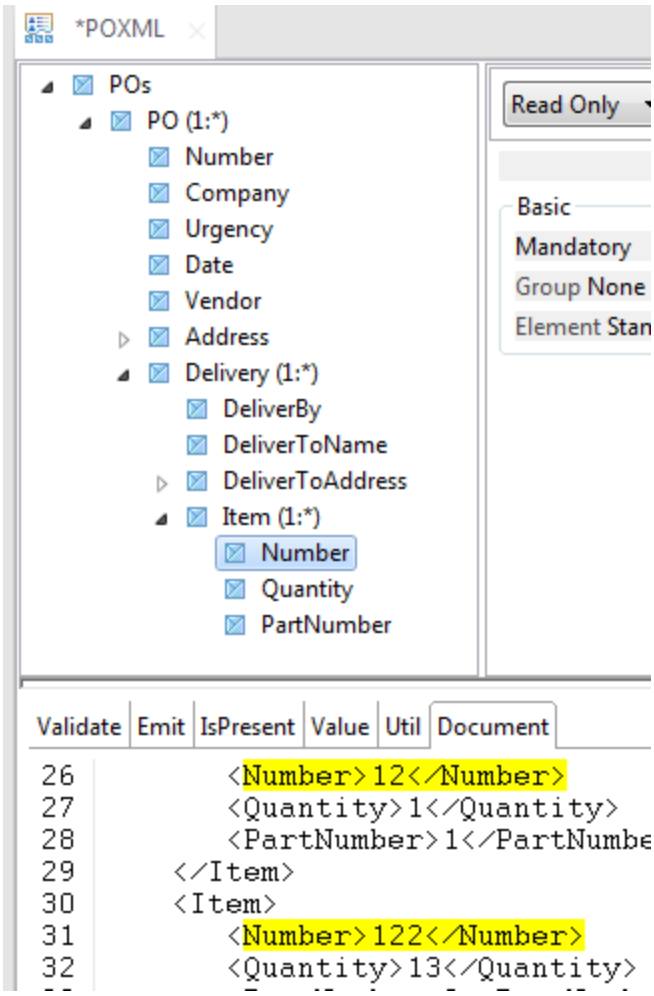
1. Expand **Show Document** from the toolbar by clicking the down arrow next to it, then select **Select Sample Document > Import Document from File...**



2. Click the **Browse...** button and select the *C:\StudentFiles* folder to display its contents.
3. Select the *POs.xml* file and click **Finish**.



4. Once the Sample Document has been selected, the actual data will appear in the **Document** tab. In addition, selecting a specific node in the Structure will highlight the matching data in the preview.

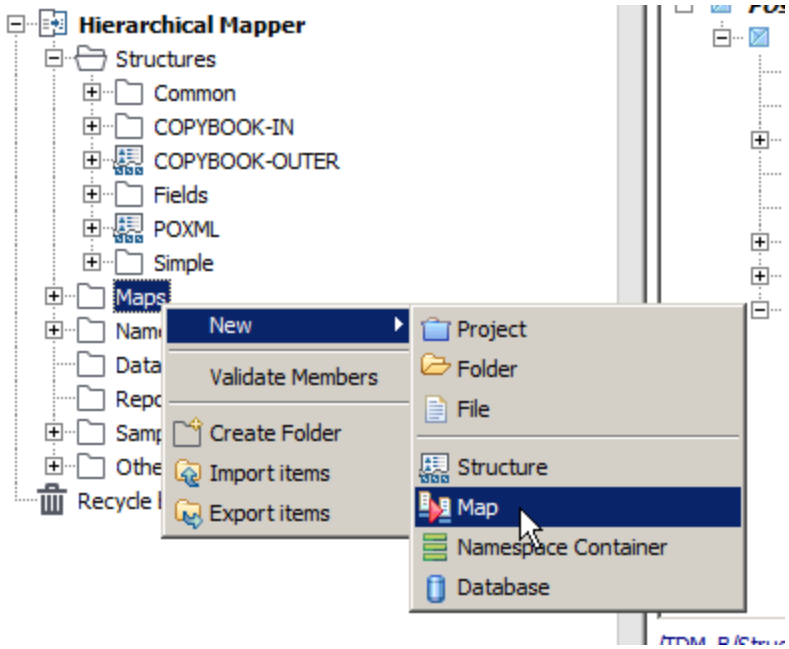


Next Step

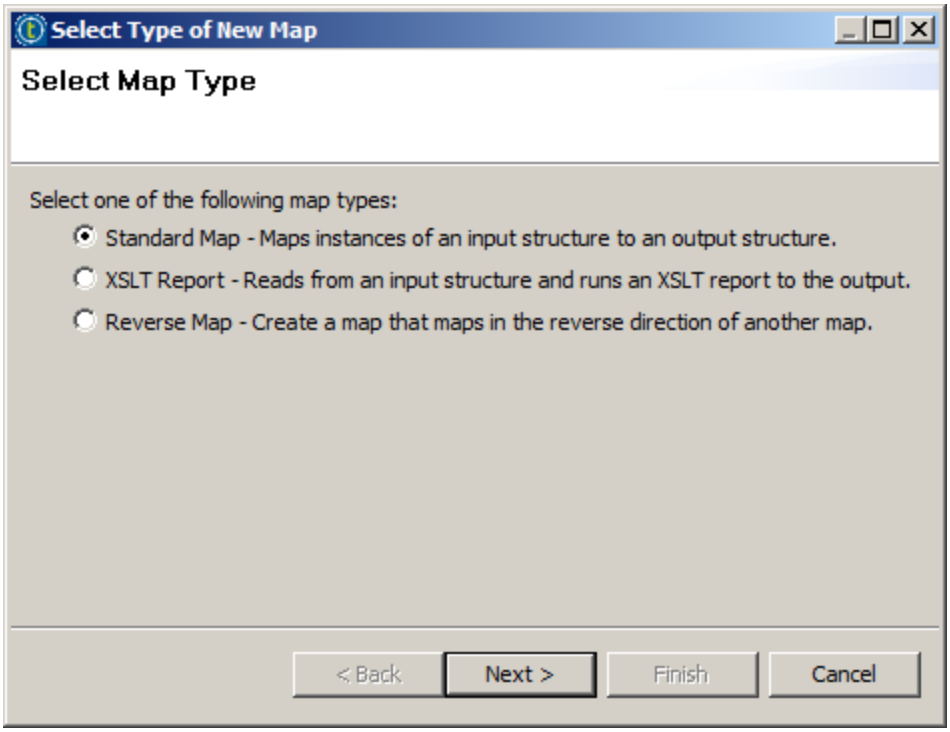
With the new Structure in place, you can now create the Map between the input and output.

Creating a Map

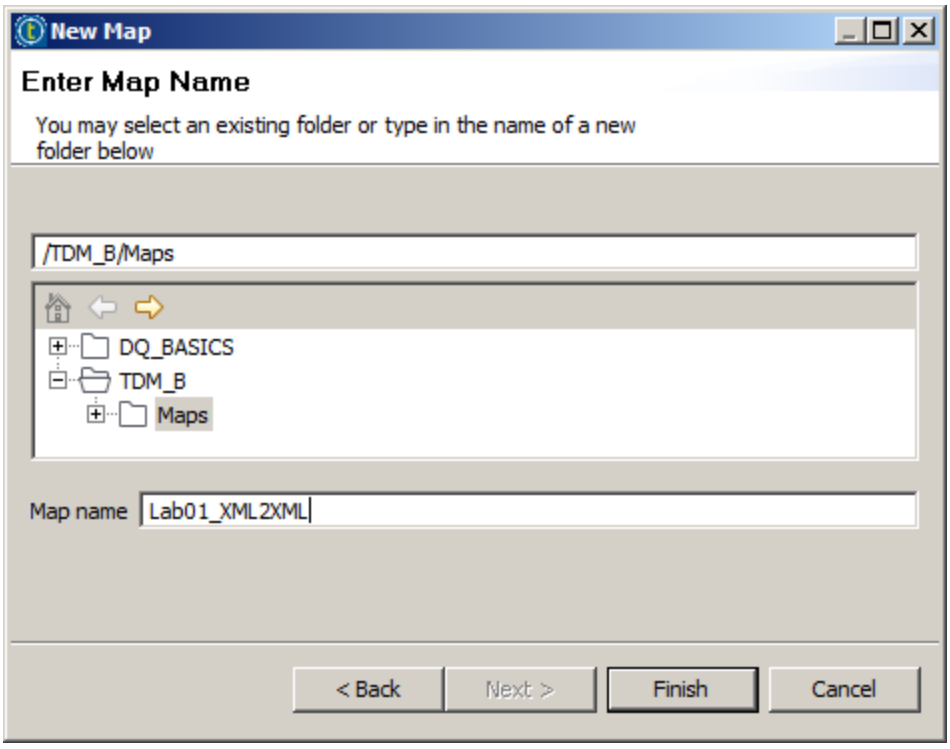
1. Right-click **Hierarchical Mapper > Maps** and select **New > Map**.



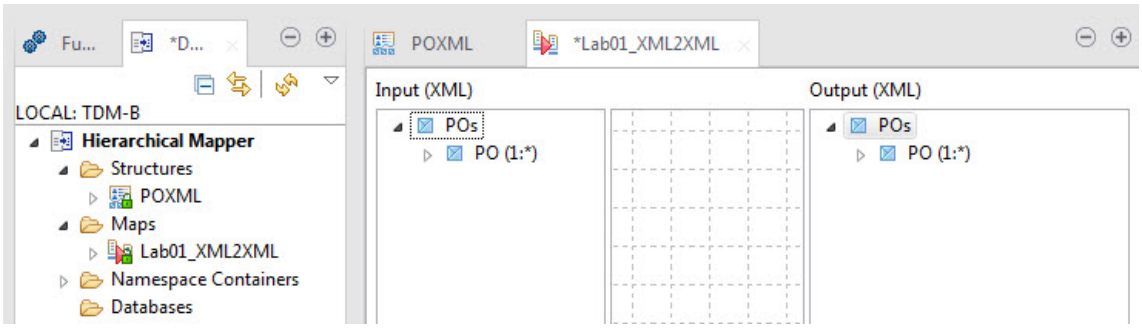
2. Select **Standard Map** and click **Next >**.



3. Type *Lab01_XML2XML* in the **Map name** box and click **Finish**.

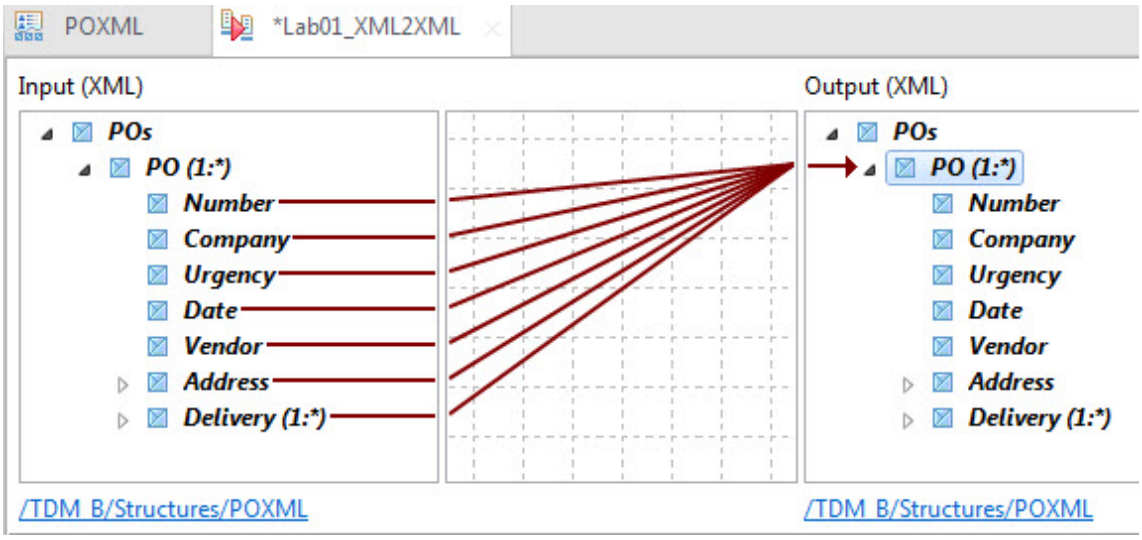


4. Drag and drop the **POXML** Structure created previously to the Input and Output areas of the Map on the right.

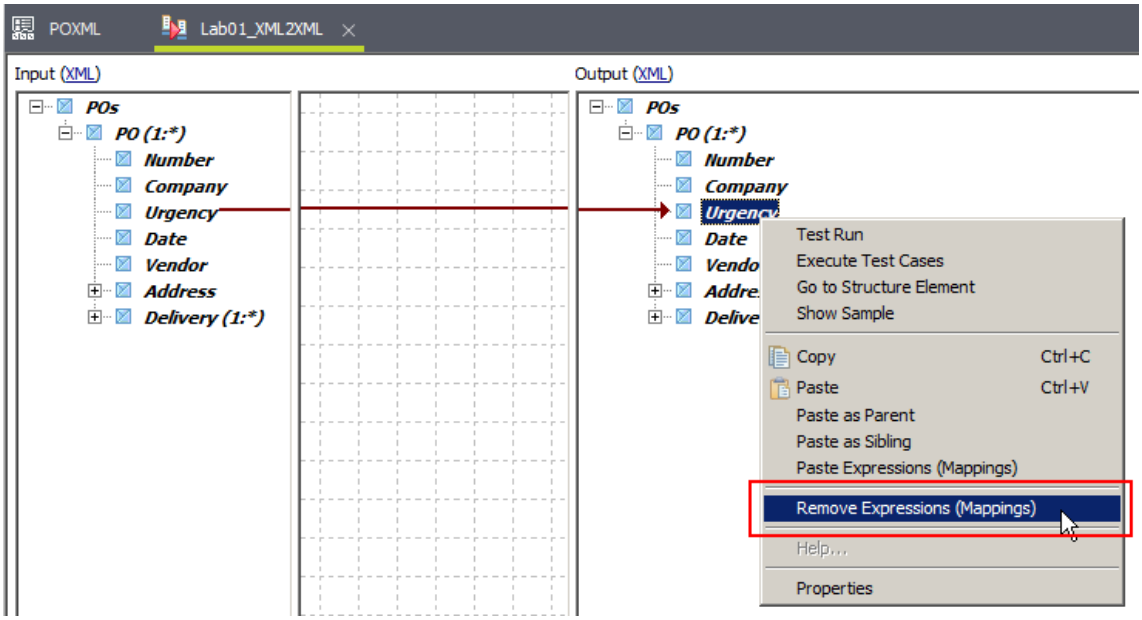


In this example, the **POXML** Structure is used for both the source and the target. In a more general use case, two different Structures would be used for the mapping.

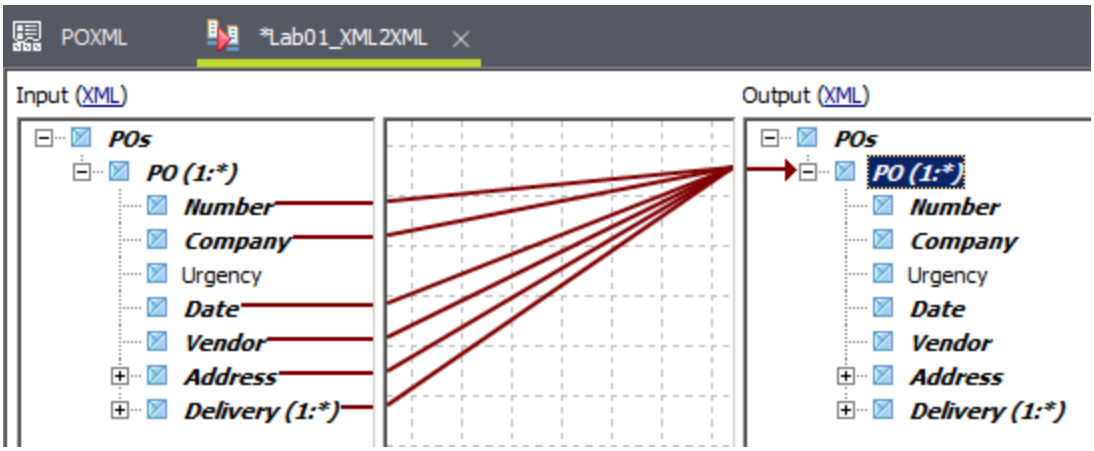
5. Drag **PO (1:*)** from the **Input** area to the matching group level on the **Output** area. Doing so will perform an automatic mapping on all fields whose names match. In this case, as the input and output structure are the same, all the fields will be mapped to each other. Now let's remove some elements from the output.



6. Select the *Urgency* node in the **Output** area, right-click it and select **Remove Expressions (Mappings)** to remove the mappings associated with this particular element. It means the node will not be part of the output.

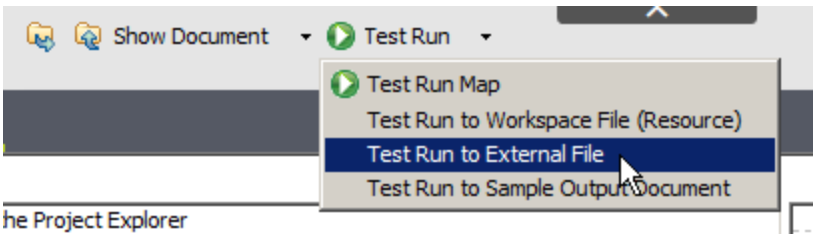


7. Note that the *Urgency* node no longer appears in bold in both the **Input** and **Output** areas, to indicate it is no longer mapped. Also, selecting the *PO (1:*)* node in the **Output** area clearly shows one of the red arrows disappeared.

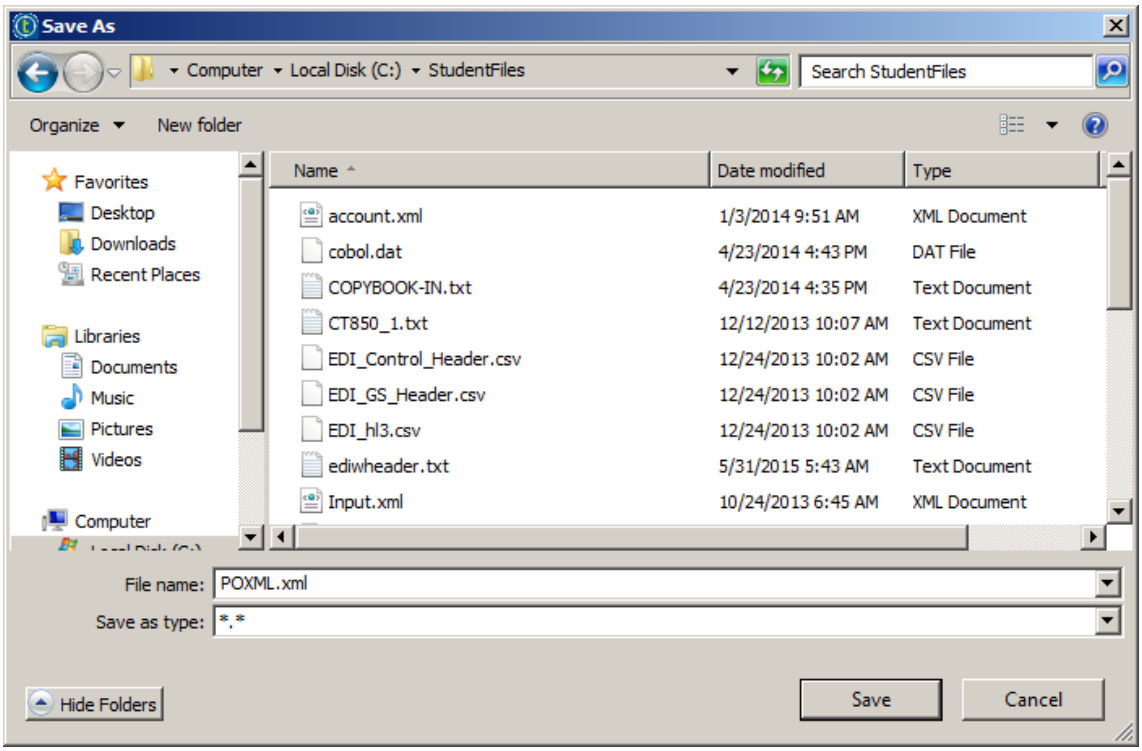


8. Select the *Vendor* node in the **Output** area, right-click it and select **Remove Expressions (Mappings)** again. This way, the unnecessary elements are removed from the output and the file will be smaller in size. Note that another red arrow disappeared, indicating a smaller number of mapped elements.

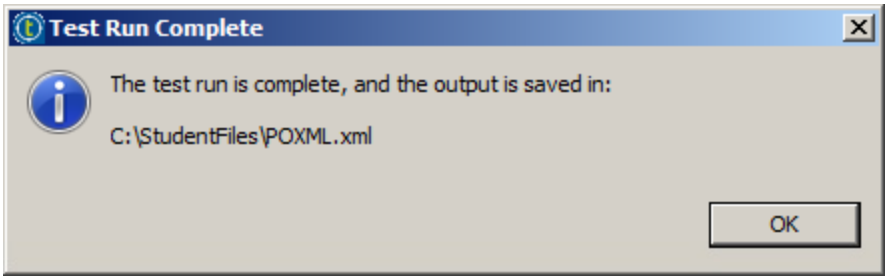
9. Save the Map by hitting **Ctrl+S** or clicking the floppy icon in the toolbar.
10. Expand **Test Run** then click **Test Run to External File**.



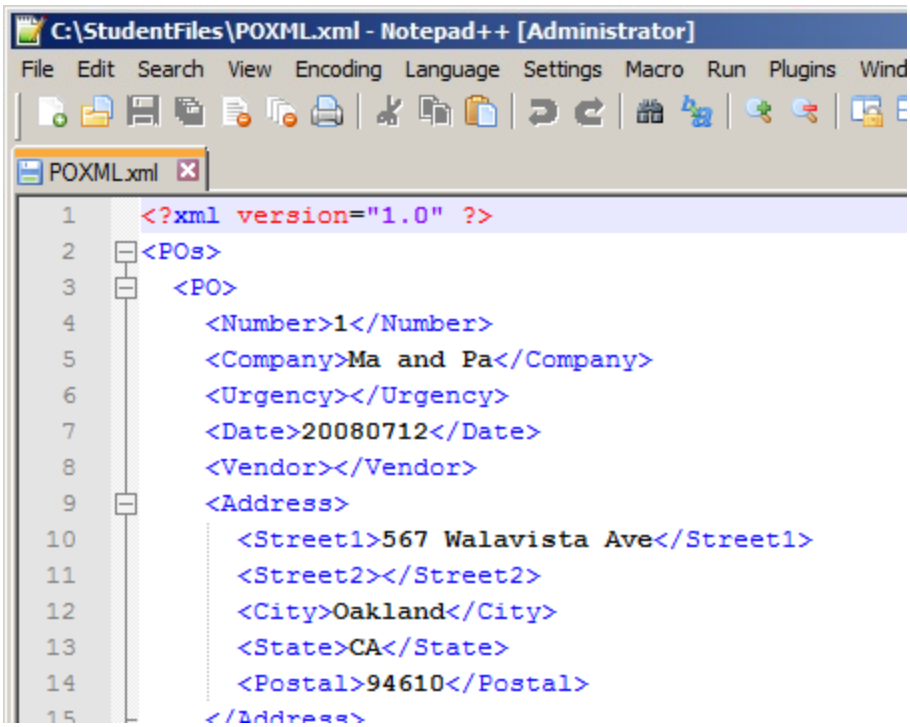
11. Browse to *C:\StudentFiles* then type **POXML.xml** in the **File name:** field and click **Save**.



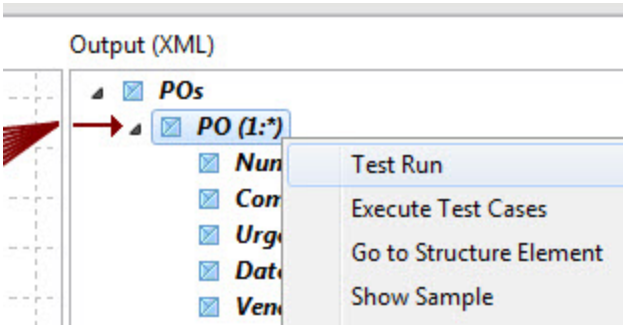
12. The Test Run results will be saved to the *POXML.xml* file specified in the previous step.



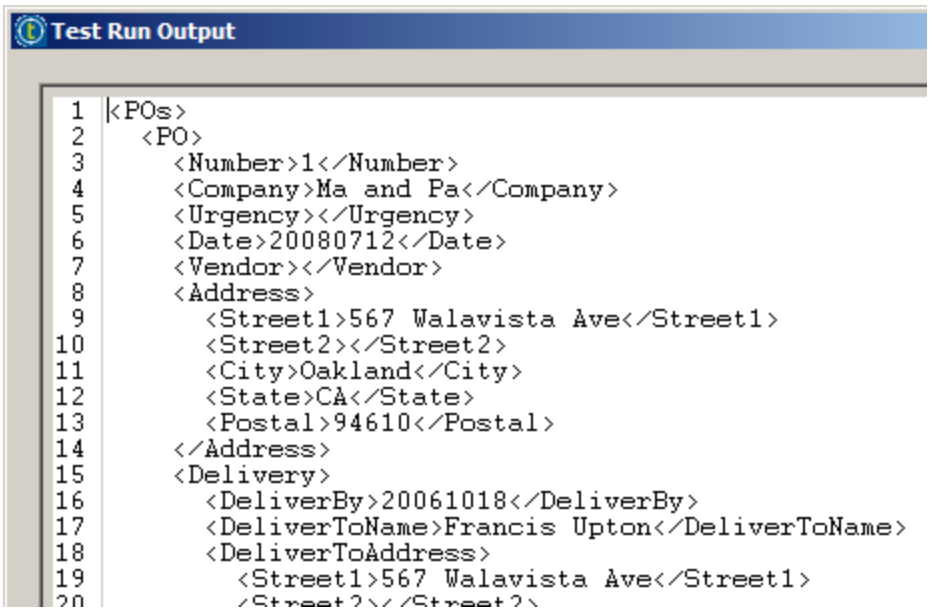
13. Open *C:\StudentFiles\POXML.xml* with Notepad++ (pre-installed on the Virtual Machine). It should be equal to the original *POs.xml* file, minus the *Urgency* and *Vendor* children that now appear as empty nodes.



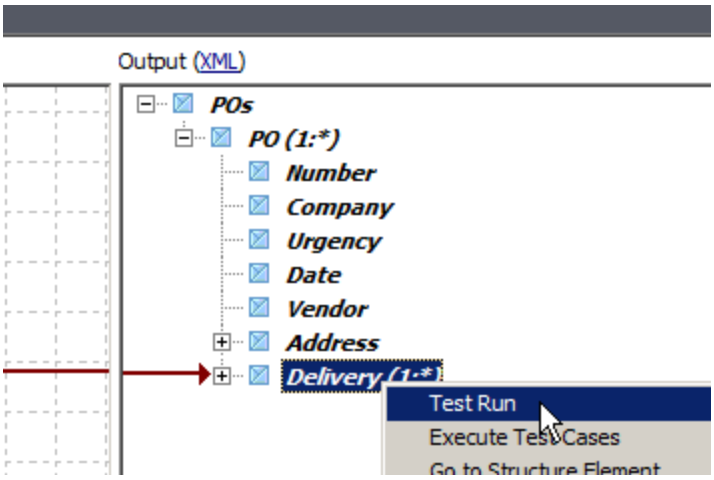
14. Test your map again by right-clicking any group level or element on the output side and selecting **Test Run**.



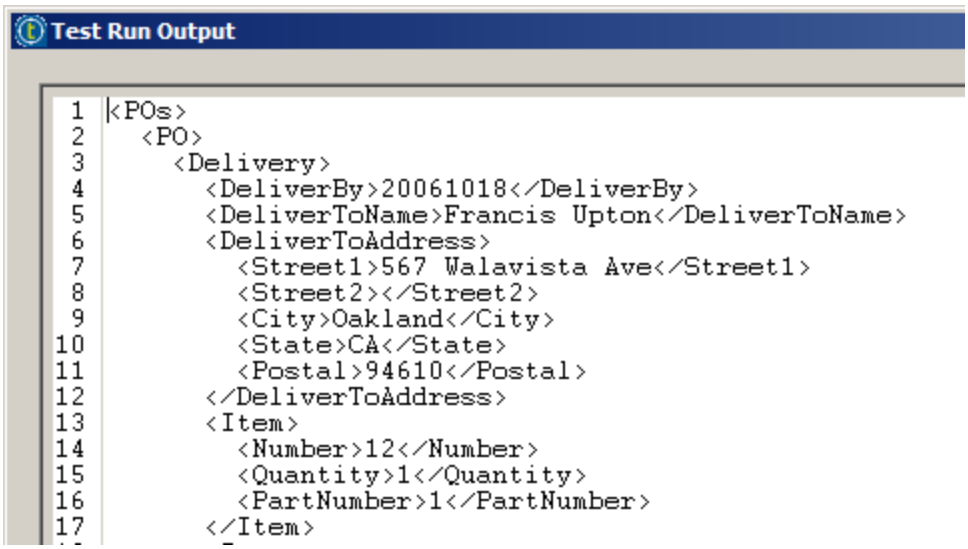
15. This time, the results are displayed in the internal XML viewer.



16. Right-Click **Delivery** in the **Output (XML)** area and select **Test Run.**



17. Note that only the **Delivery** level and everything below is displayed.



Next Step

This lesson is almost over. Head to the Wrap-Up section for a summary of the concepts reviewed in this lesson.

Wrap-Up

This first lab served as a general introduction to Talend Data Mapper.

In this lesson, you learned how to:

- Create a new Structure from an existing local XML file
- Explore a Structure and interactively display a sample of it
- Create a new Map
- Use existing Structures to specify the Input and Output areas of a Map
- Automap the elements whose names match in both the Input and Output areas of a Map
- Remove some elements from the output
- Test Run the Map and save the results to an external file, or display them in the internal viewer

Next Step

Congratulations, you successfully completed this lesson. Click the **Check your status with this unit** button below in order to save your progress. Then click **Completed. Let's continue >** on the next screen to jump to the next lesson.

LESSON 2

Simple Looping

This chapter discusses the following.

Overvieww

Lesson Overview

This lab takes the previous example a step further and explains the simple looping concept. It also shows how to sort and filter the output values of a Map.

Objectives

After completing this lesson, you will be able to:

- Add a Simple Loop manually for elements that can appear multiple times as children of a common parent node
- Sort an Output based on the value of a given Input field
- Filter an Output based on the value of a given Input field
- Change the representation and type of a given field

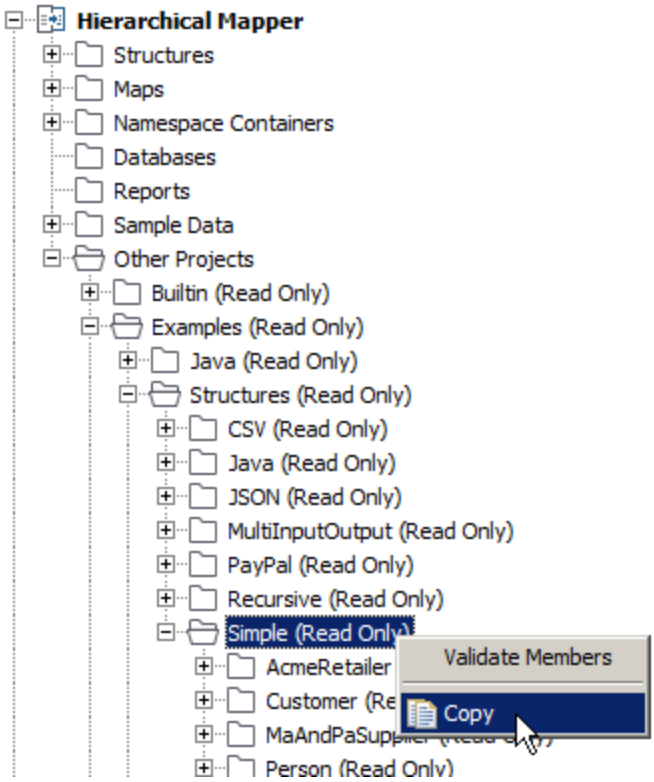
Next Step

First, let's set up and prepare the required Structures.

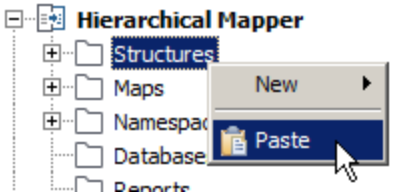
Preparation

First, let's copy some sample Structures to our workspace, to reuse them and not reinvent the wheel.

1. Right-click the **Hierarchical Mapper > Other Projects > Examples > Structures > Simple** folder and select **Copy**.



2. Right-click the main **Hierarchical Mapper > Structures** folder and select **Paste**.

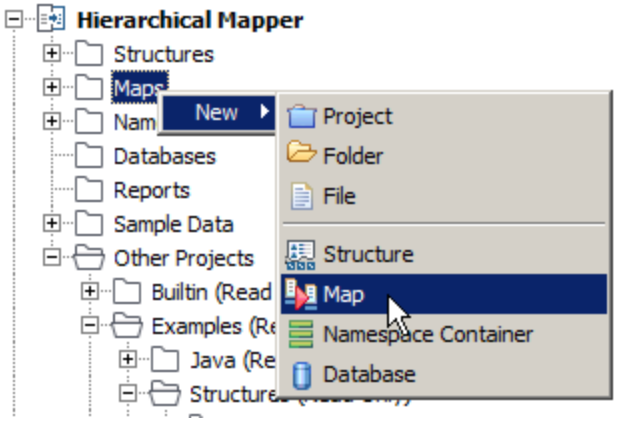


Next Step

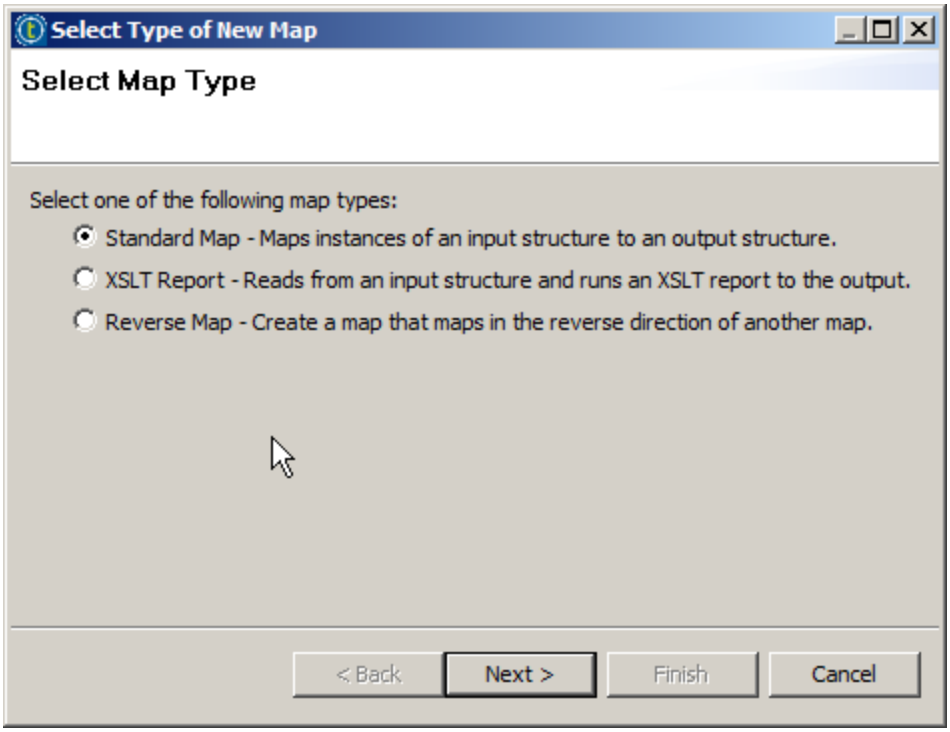
With the example Structures copied to the workspace, let's create a new Map.

Creating a New Map

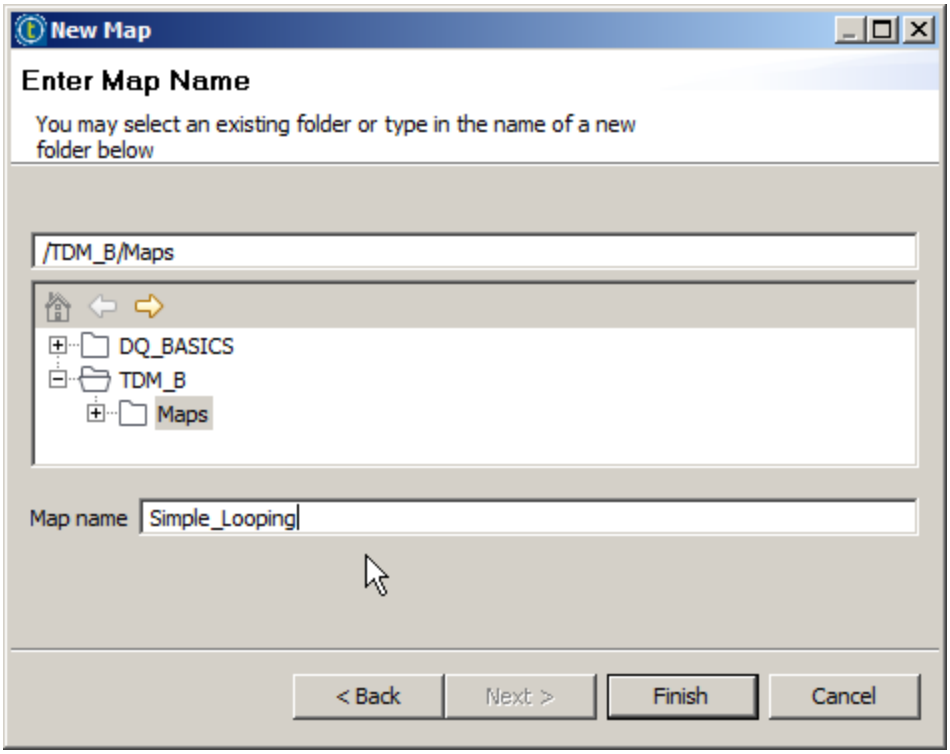
1. Right-click **Hierachical Mapper > Maps** and select **New > Map**.



2. Select the **Standard Map** option and click **Next >**.

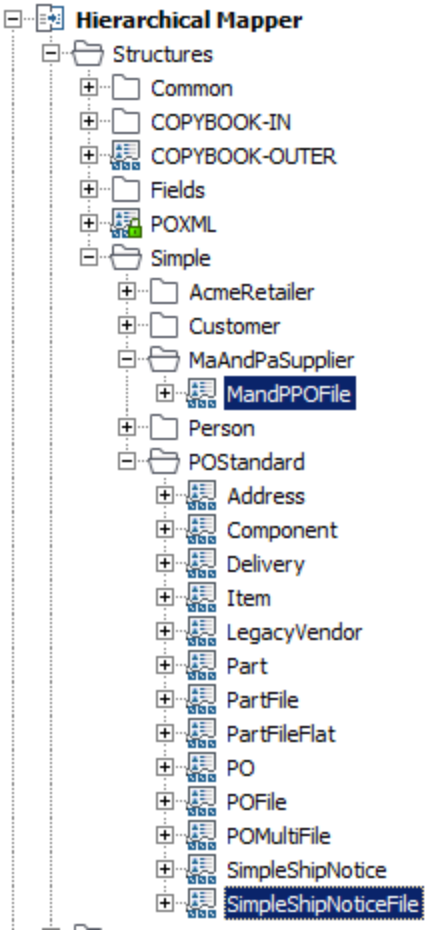


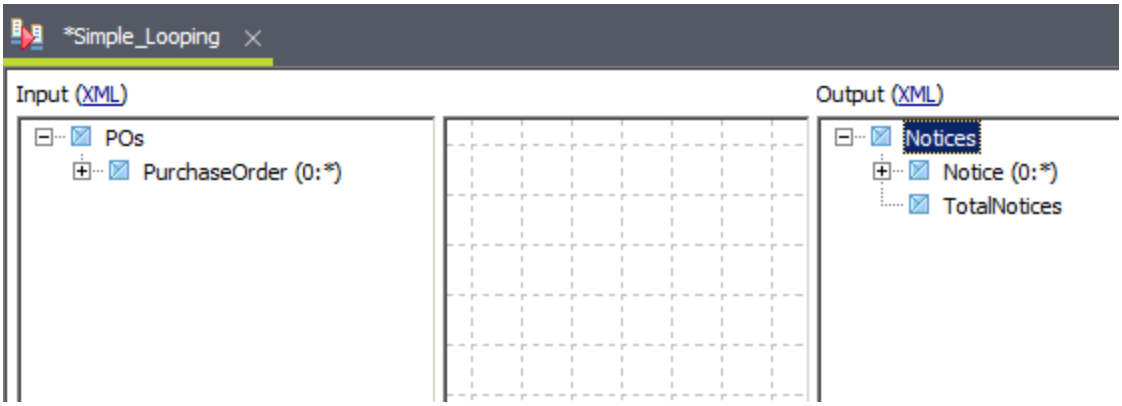
3. Call the new Map **Simple_Looping** and click **Finish**.



Defining the Input and Output

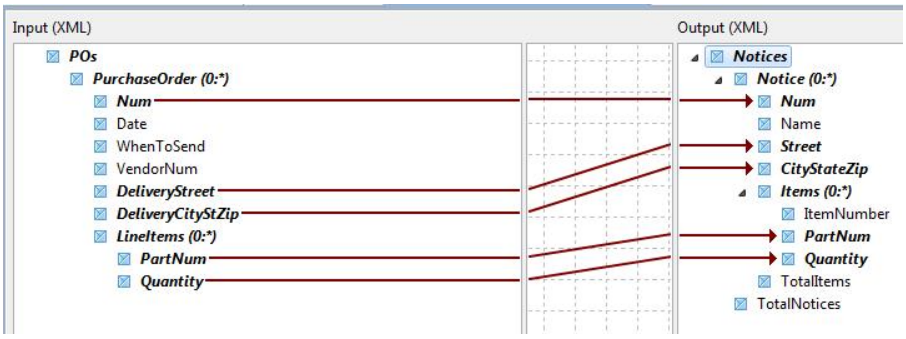
1. Drag and drop **Hierarchical Mapper > Structures > Simple > MaAndPaSupplier > MandPPOFile** to the **Input** area of the new **Simple_Looping** Map and **Hierarchical Mapper > Structures > Simple > POStandard > SimpleShipNoticeFile** to the Output area.


Hierarchical Mapper
Structures
Common
COPYBOOK-IN
COPYBOOK-OUTER
Fields
POXML
Simple
AcmeRetailer
Customer
MaAndPaSupplier
MandPPOFile
Person
POStandard
Address
Component
Delivery
Item
LegacyVendor
Part
PartFile
PartFileFlat
PO
POFile
POMultiFile
SimpleShipNotice
SimpleShipNoticeFile

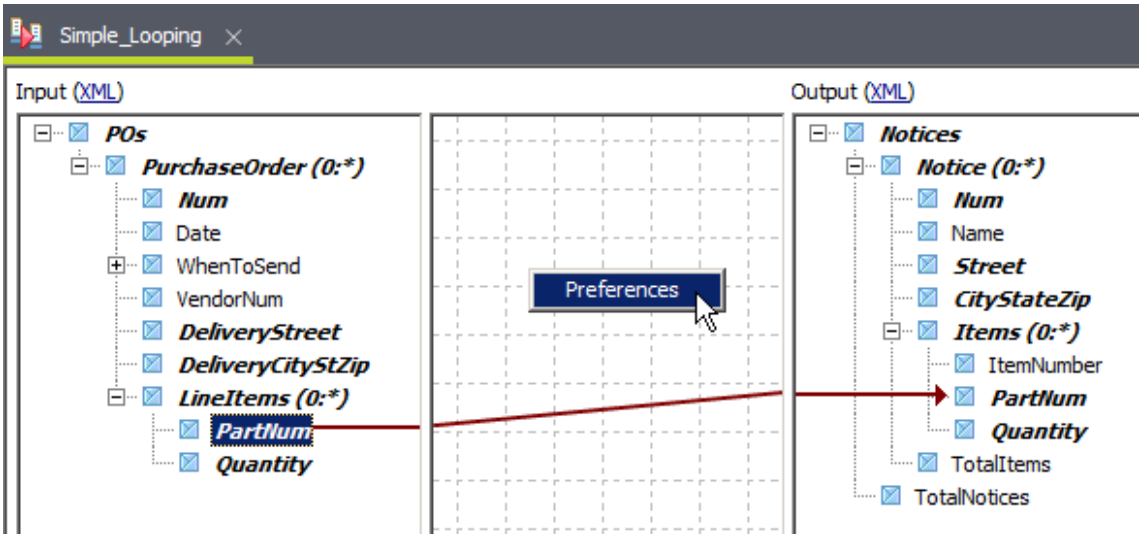

*Simple_Looping
Input (XML)
POs
PurchaseOrder (0:*)
Output (XML)
Notices
Notice (0:*)
TotalNotices

Defining the Mapping

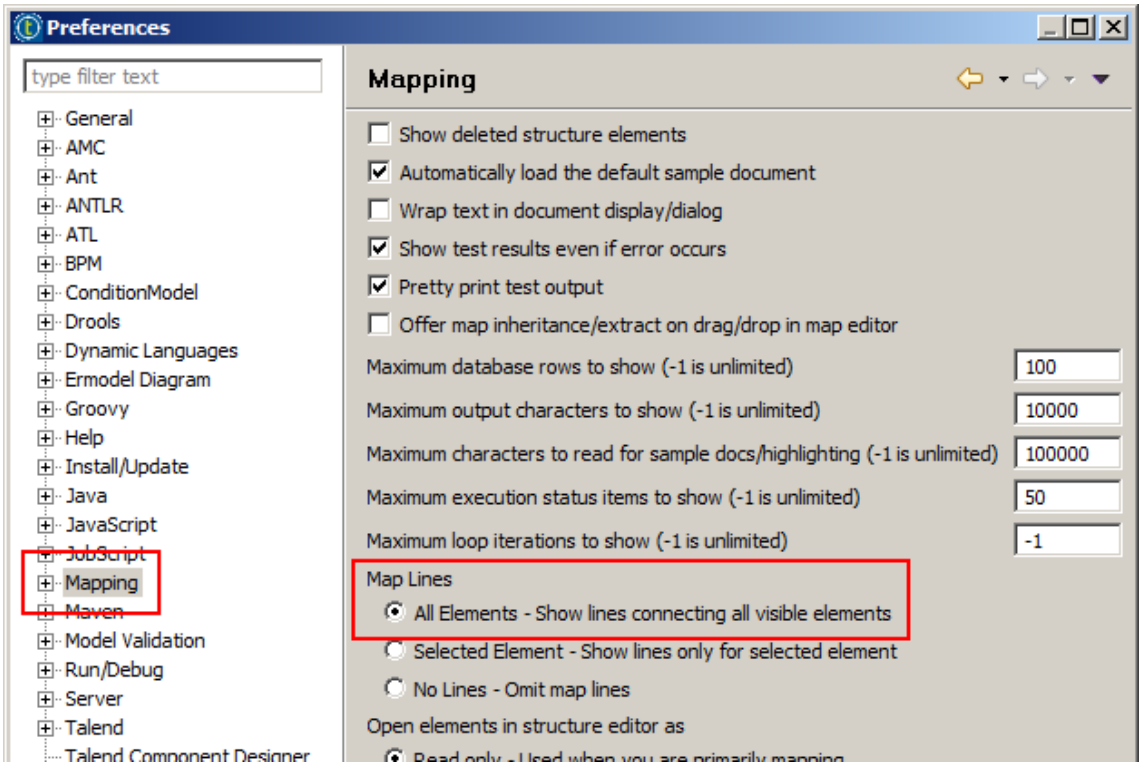
1. Map the elements as illustrated below by drag and dropping from the **Input** area to the **Output** area.



2. By default, only the active mapping is displayed on screen. To display all mappings at the same time as illustrated above, right-click the grid between the **Input** and **Output** areas and select **Preferences**.

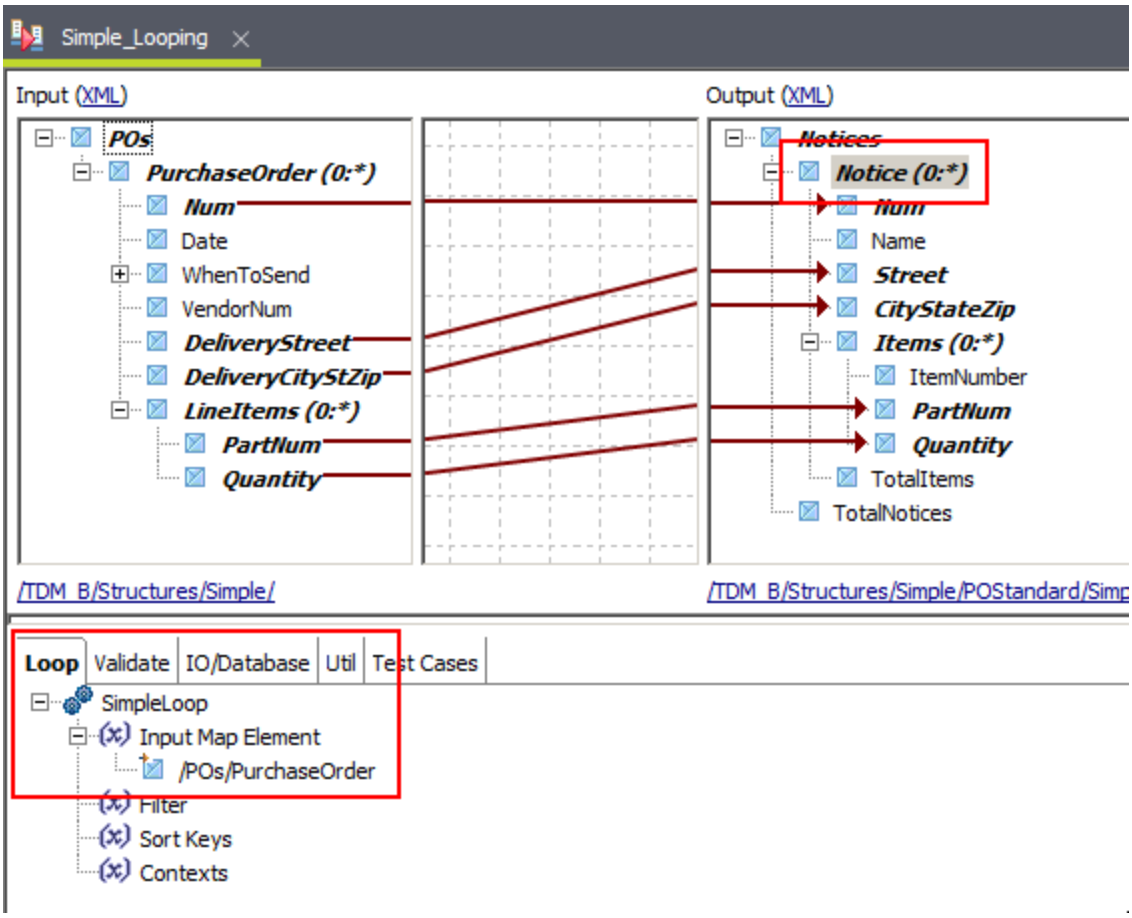


3. In the **Mapping > Map Lines**section, select **All Elements - Show lines connecting all visible elements**.

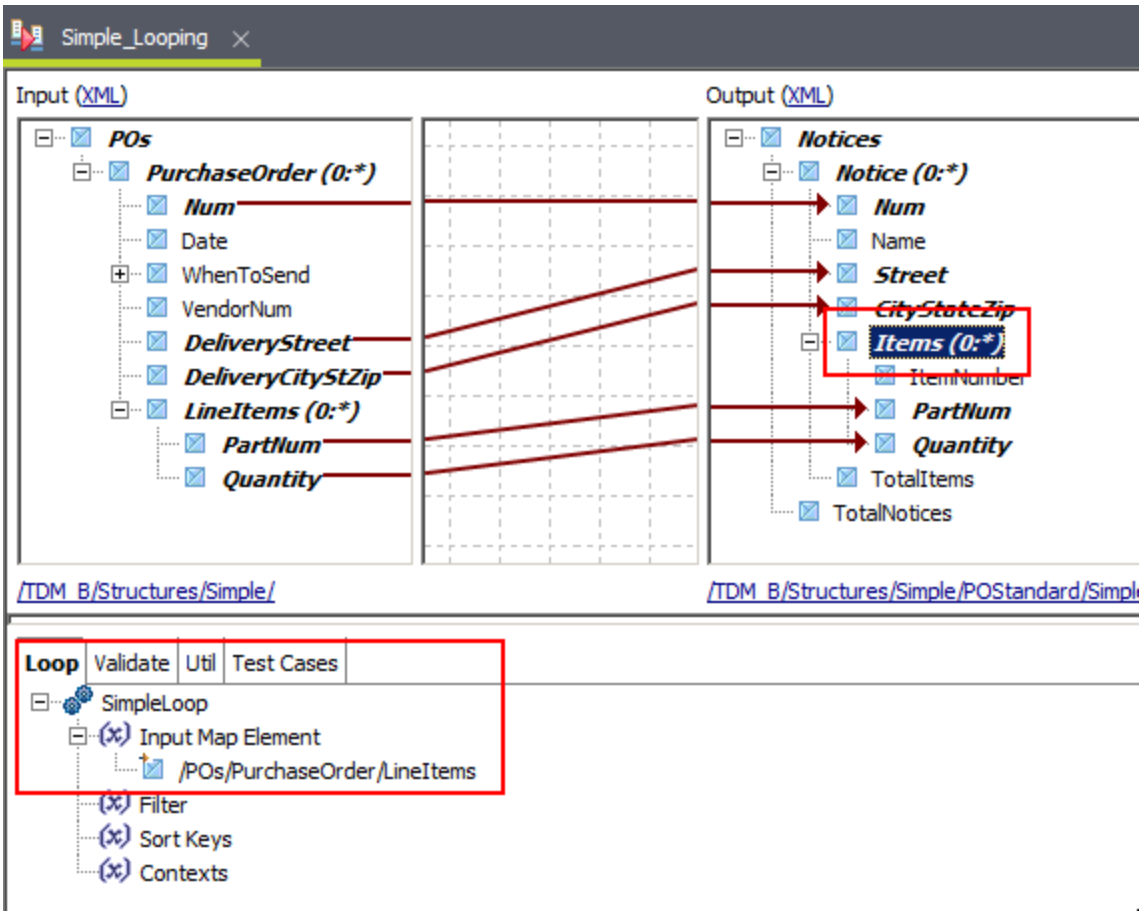


Displaying the Automatically Created Simple Loops

1. Select the **Notices > Notice (0:*)** node in the **Output** area to display the simple loop that was created automatically based on the mappings specified earlier. The simple loop appears in the **Loop** tab below the mapping area.



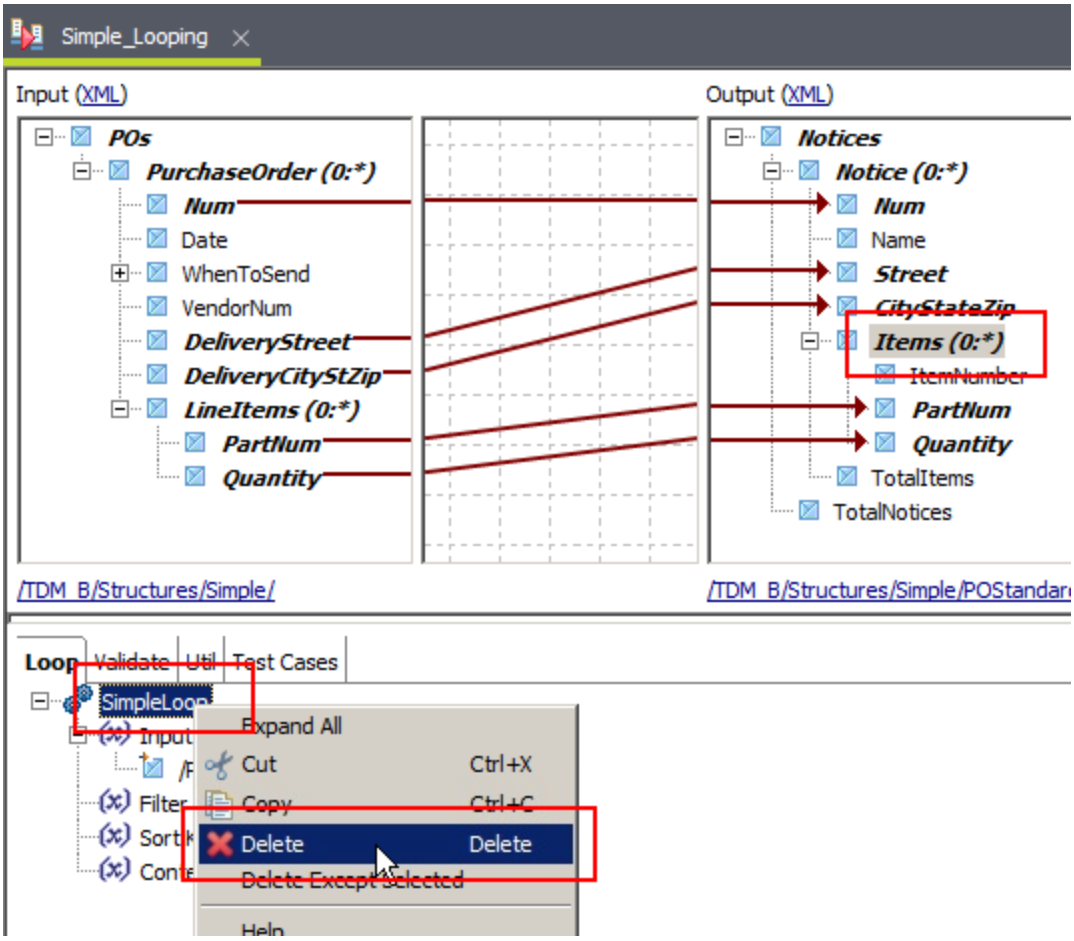
2. Select the **Notices > Notice (0:*) > Items (0:*)** node in the **Output** area to display another simple loop that was created automatically.



These Simple Loops indicate that the *Notices* node can have multiple *Notice* children. In turn, each *Notice* node can have multiple *Items* children. *Talend Data Mapper* creates Simple Loops automatically when you drag and drop a looping element to another looping element. The Map will iterate over all available children from the **Input** area to create a list in the **Output** area. It is also possible to create loops manually when needed. To illustrate this, let's remove one of the Simple Loop and create it back manually.

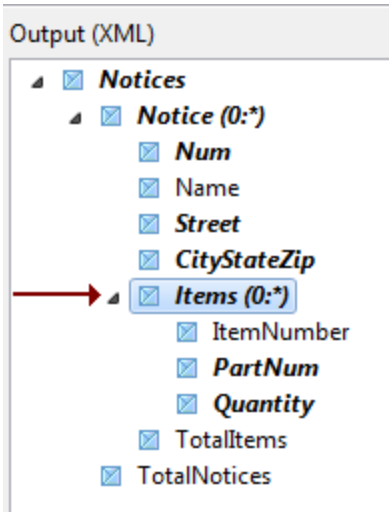
Removing the Automatic Simple Loop

1. Select the **Items (0:*)** node in the **Output** area, right-click the **SimpleLoop** element in the **Loop** tab and select **Delete**.

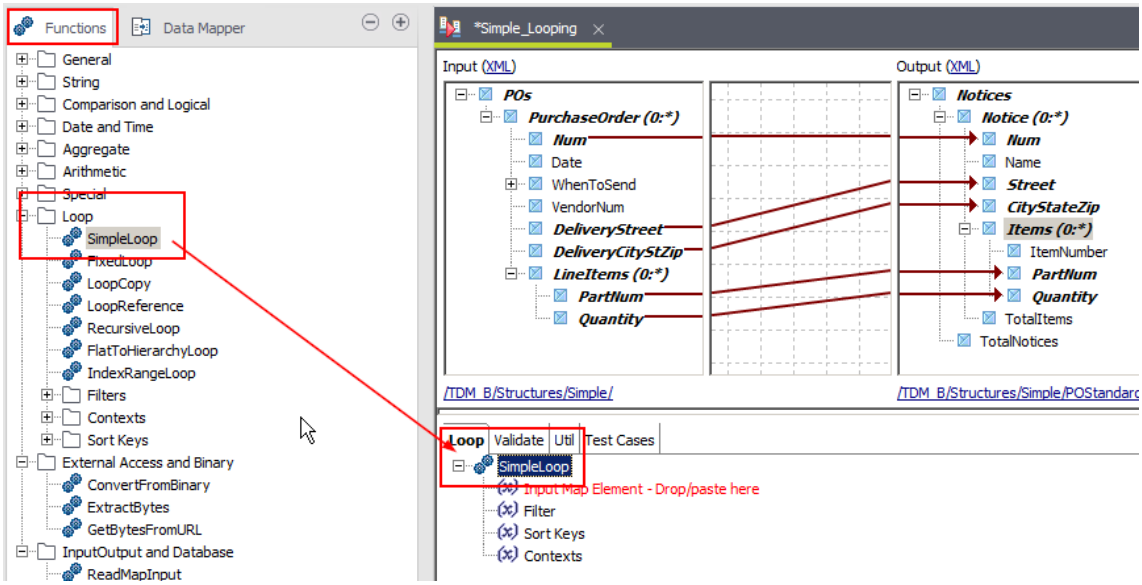


Adding Back the Simple Loop

1. Make sure the **Items (0:*)** is still selected.

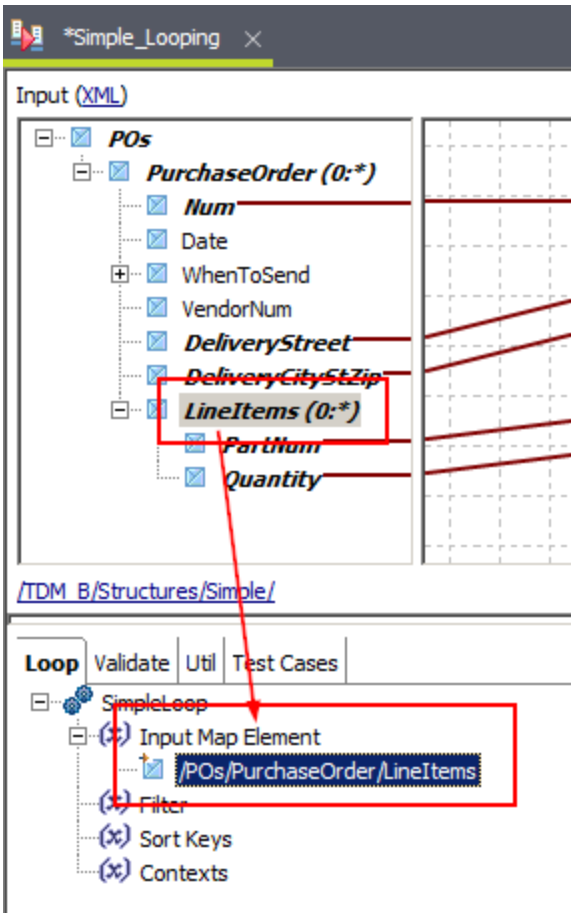


2. Drag a **SimpleLoop** element from the **Functions** tab on the left and drop it in the **Loop** tab.



Setting the Input Map Element

1. Drag the **LineItems (0:*)** node from the Input area and drop it over the **Input Map Element - Drop/paste here** under **SimpleLoop**.



Important: do not click the **LineItems (0:*)** node *before* performing the drag and drop operation. Clicking the node will select it and refresh the Loop tab at the bottom, so the **Input Map Element - Drop/paste here** area will disappear from view. Make sure the **Items (0:*)** node is selected in the **Output** area, then drag and drop the **LineItems (0:*)** node from the **Input** area to the Loop tab without any intermediary click.

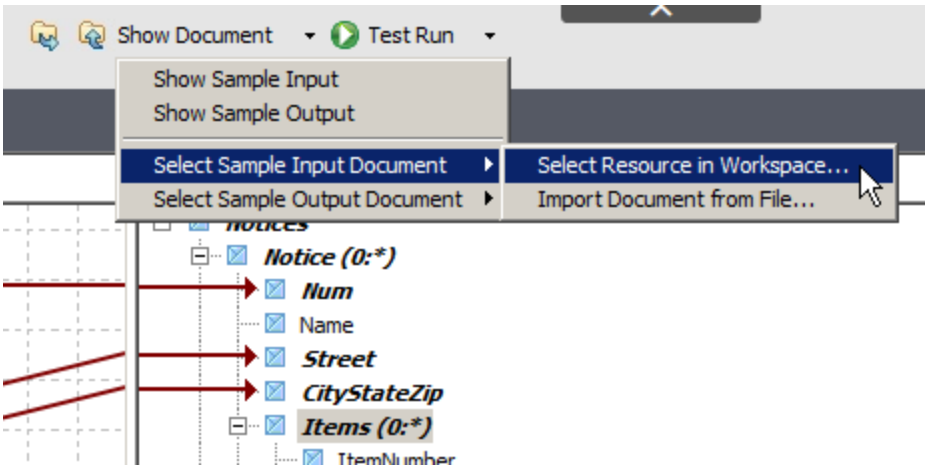
Next Step

The mapping is done, and Simple Loops were added automatically or manually. Now let's test the Map.

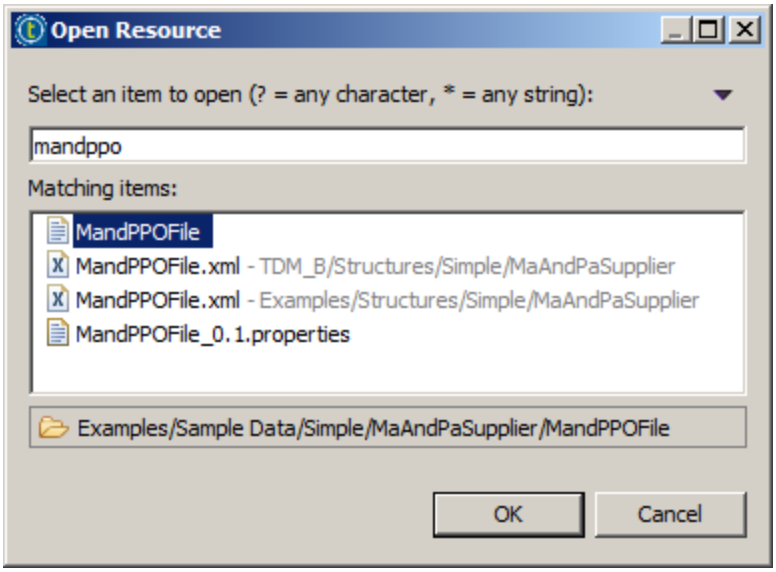
Testing the Map

Configuring the Sample Document to Use

1. Expand the **Show Document** button in the toolbar and select **Select Sample Input Document > Select Resource in Workspace...**.

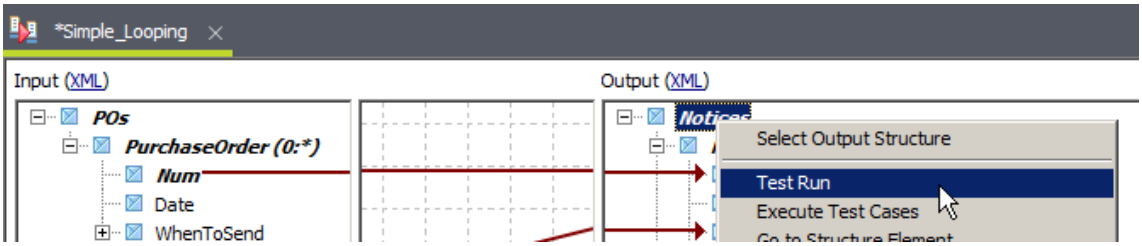


2. Enter **mandppo** to filter the available Resources.
3. Select the **MandPPOFile** item located in the **Examples/Sample Data/Simple/MaAndPaSupplier** folder.

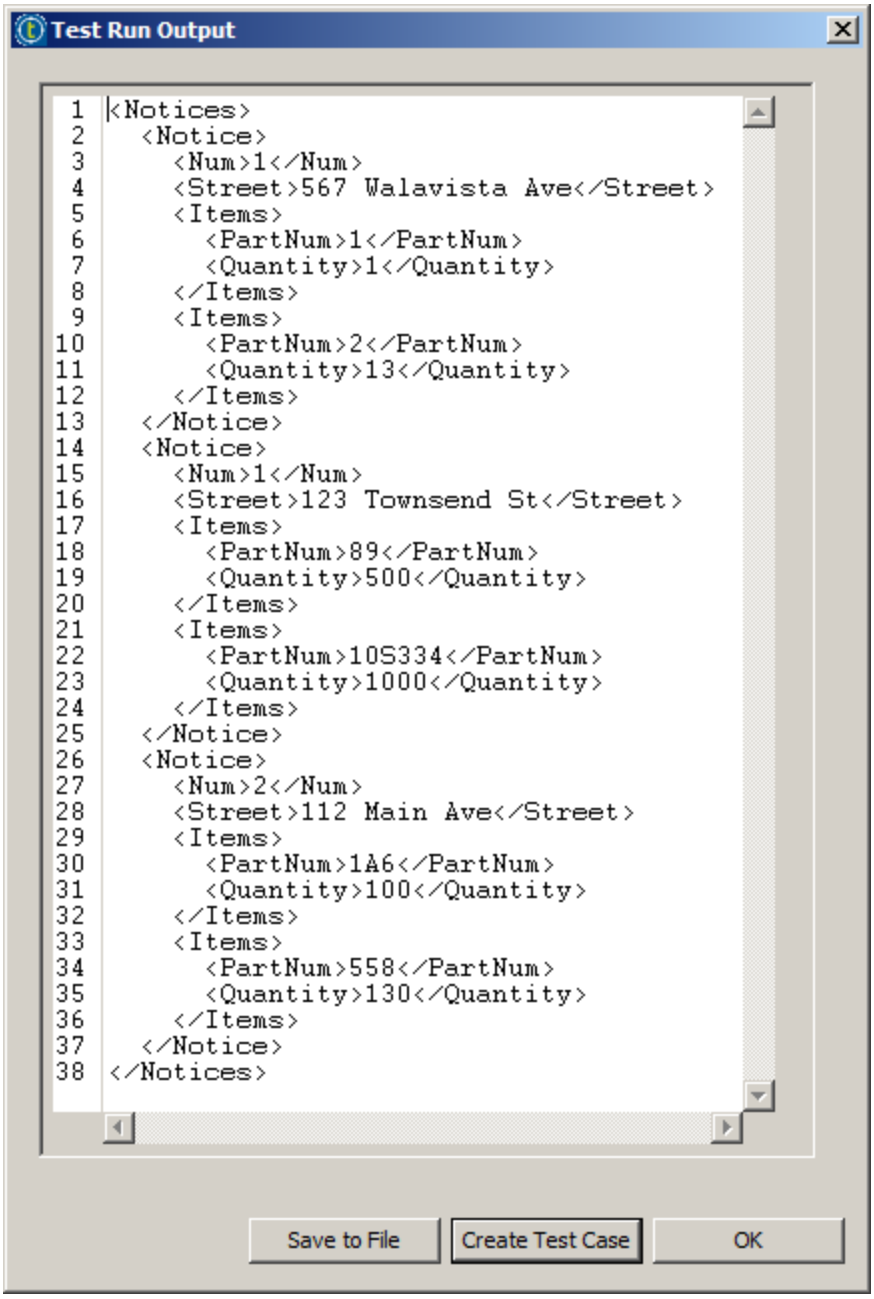


Testing the Map using the Selected Sample Document

1. Right-click the **Notices** in the **Output** area and select **Test Run**:

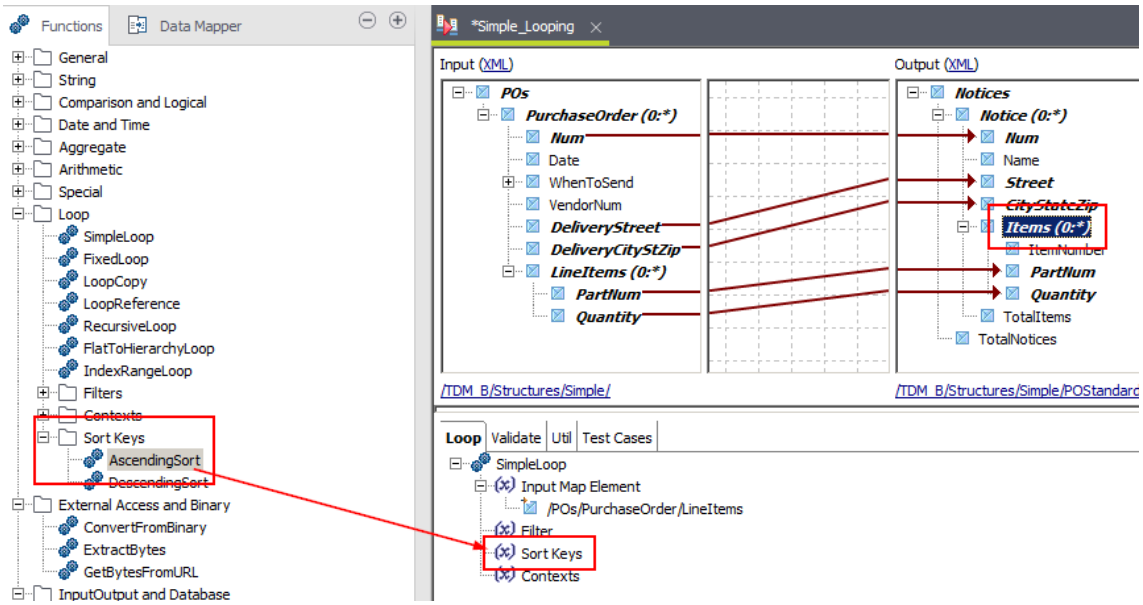


2. Check the result looks like the following:

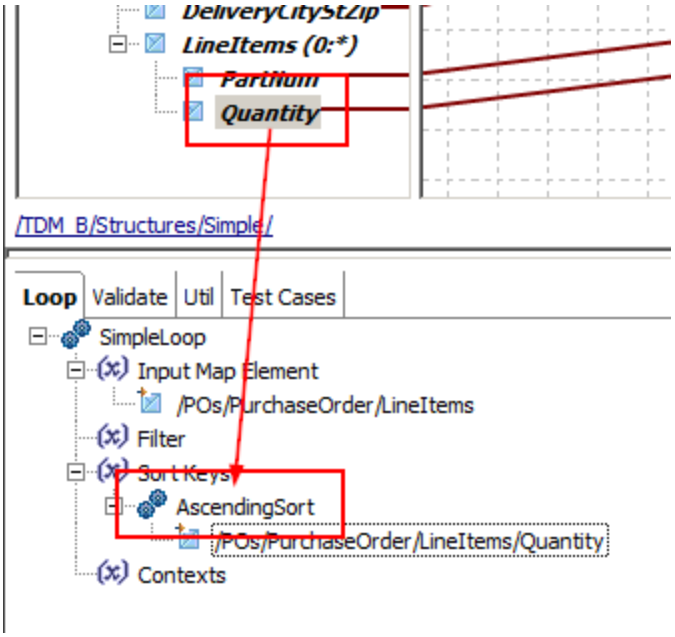


Sorting the Output by Quantity

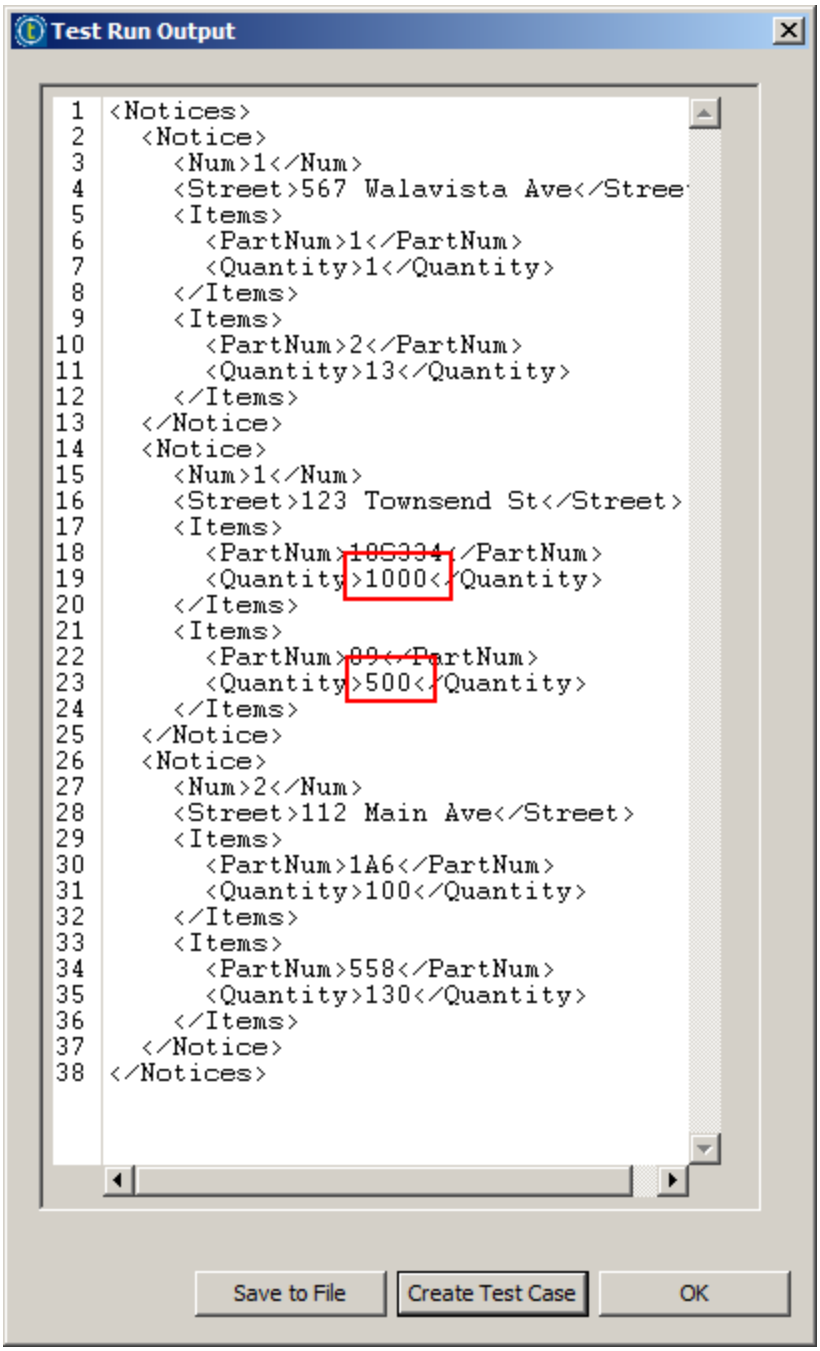
1. Select the **Items (0:*)**element in the Output area.
2. Drag an **AscendingSort** element from Functions > Loop > Sort Keys category on the left. Drop it on SimpleLoop > Sort Keys element in the Loop tab.



3. Drag the **Quantity** element from the Input area and drop it on top of the **AscendingSort** entry. Again, make sure to perform the drag and drop in a single operation, with no intermediary click on the **Quantity** element. Selecting the **Quantity** element with a single left click prior to performing the drag and drop will make the **Loop** tab disappear, and the drop step will be impossible to perform.



4. Click **Test Run** and check the result looks like the following:



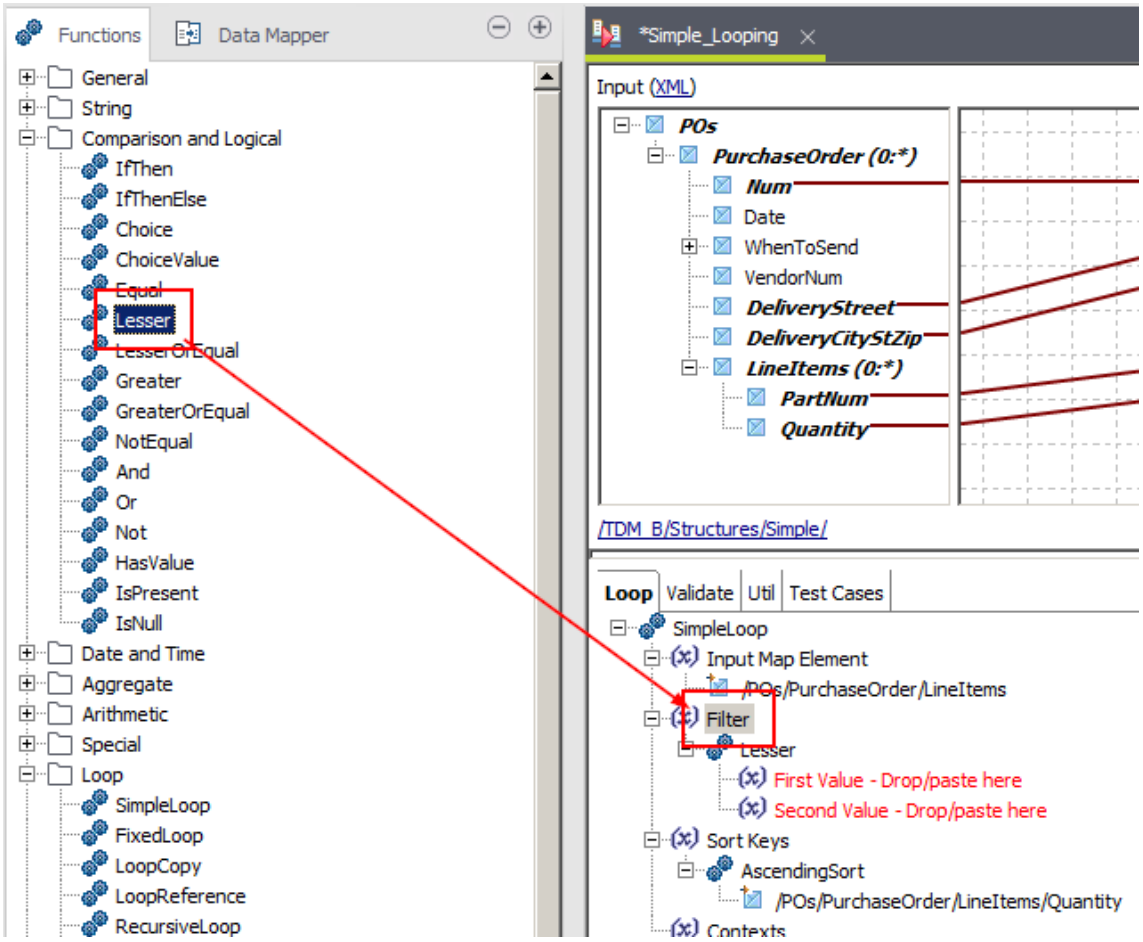
Note that **Quantity** is a string field, so the sort order highlighted above is actually correct. **1000** is smaller than **500** when represented as a string.

Next Step

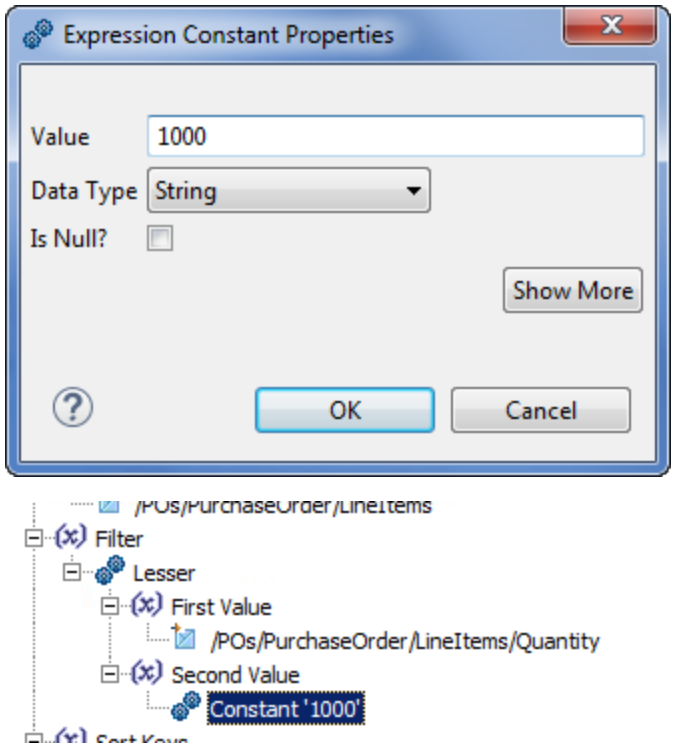
The output is sorted by ascending quantity. Now let's add a filter in order to remove the entries whose Quantity value is less than 1000.

Adding a Filter

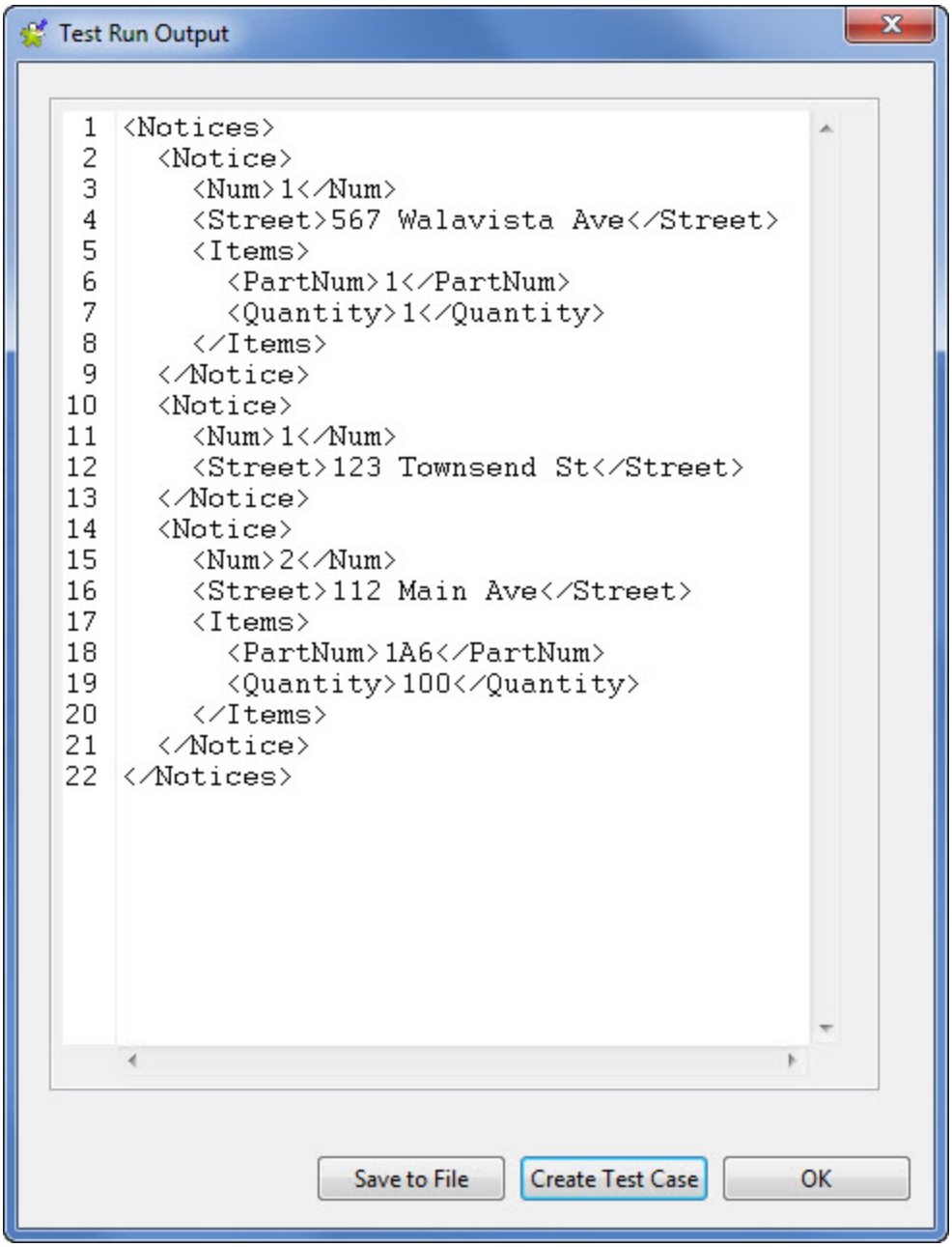
1. Select the **Items (0:*)** node in the Output area.
2. Drag and drop a **Functions > Comparison and Logical > Lesser** function to the **Filter** option in the **Loop** tab:



3. Drag the **Quantity** node from the **Input** area to the **First Value** field of the **Lesser** function.
4. Drag a **Functions > General > Constant** function to the **Second Value** field of the **Lesser** function. Double-click it, enter **1000** for the **Value** and click **OK**.

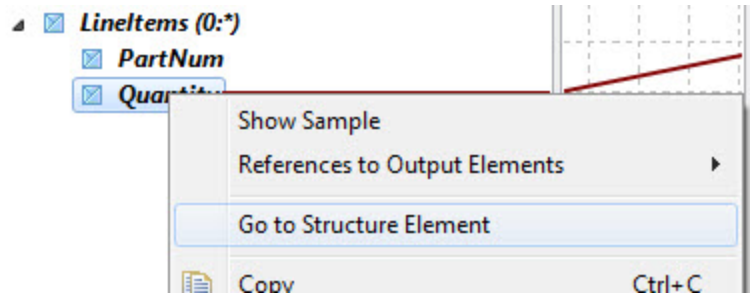


5. Click **Test Run** and check the result looks like the following:

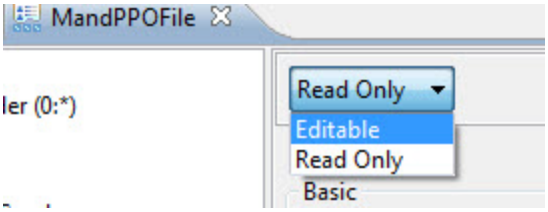


This result might seem counter-intuitive at first, but remember the **Quantity** field is considered as a string. The filter removes items whose **Quantity** value *as a string* is less than 1000. For a more human-readable output, let's see how to change the internal representation and type of the **Quantity** field, so it is considered as an integer and filtering makes more sense from a pure mathematical point of view.

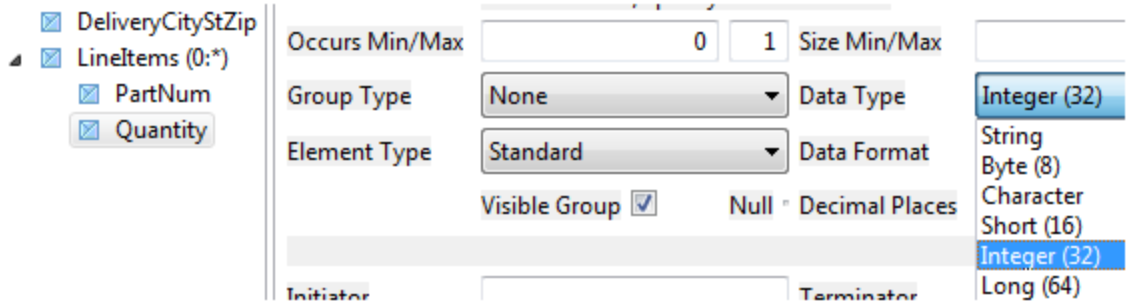
6. Right-click the **Quantity** element in the Input area and select **Go to Structure Element**.



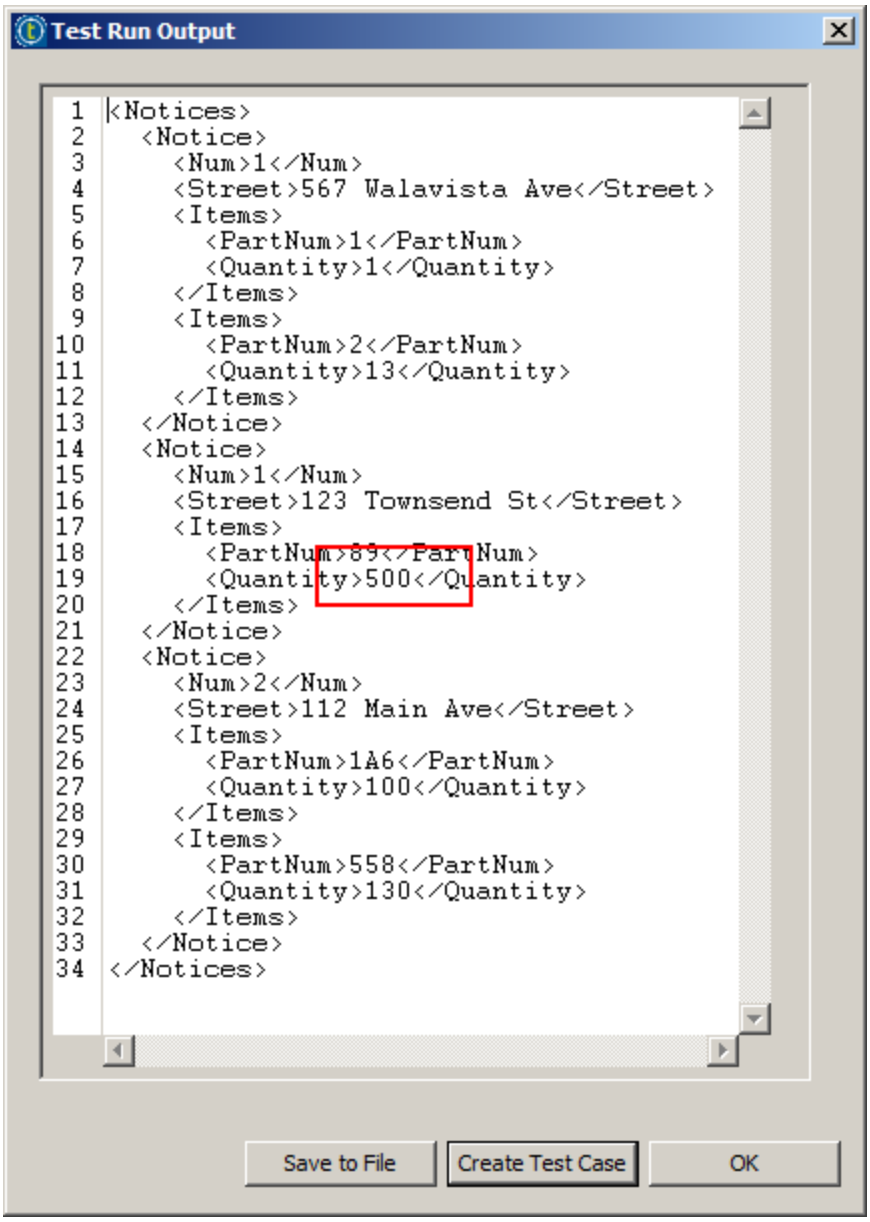
7. Change the Structure Element from **Read Only** to **Editable**.



8. Set the **Data Type** field to *Integer (32)* (instead of *String* by default).



9. Save your changes to *MandPPOFile* by hitting **Ctrl+S** or clicking the floppy disk icon in the toolbar (on the far left). Then go back to the *Simple_Looping* Map.
10. Repeat steps 6-9 for the **Quantity** element in the **Output** area.
11. Click **Test Run** and check the results are actually different. For example, there is now a **Notice** element with a **Quantity** value of 500, as 500 is considered less than 1000 when represented as a number. It was not the case when both values were stored as strings.



Next Step

This lesson is almost over. Head to the Wrap-Up section for a summary of the concepts reviewed in this lesson.

Wrap-Up

In this lesson, you learned how to:

- Add a Simple Loop (automatically and manually) for elements that can appear multiple times as children of a common parent node
- Sort an Output based on the value of a given Input field
- Filter an Output based on the value of a given Input field
- Change the representation and type of a given field

Next Step

Congratulations, you successfully completed this lesson. Click the **Check your status with this unit** button below in order to save your progress. Then click **Completed. Let's continue >** on the next screen to jump to the next lesson.

LESSON 3

Nested Looping

This chapter discusses the following.

Overview

Lesson Overview

This lab goes beyond the Simple Loop concept introduced in the previous exercise. It introduces the Nested Looping concept

Objectives

After completing this lesson, you will be able to:

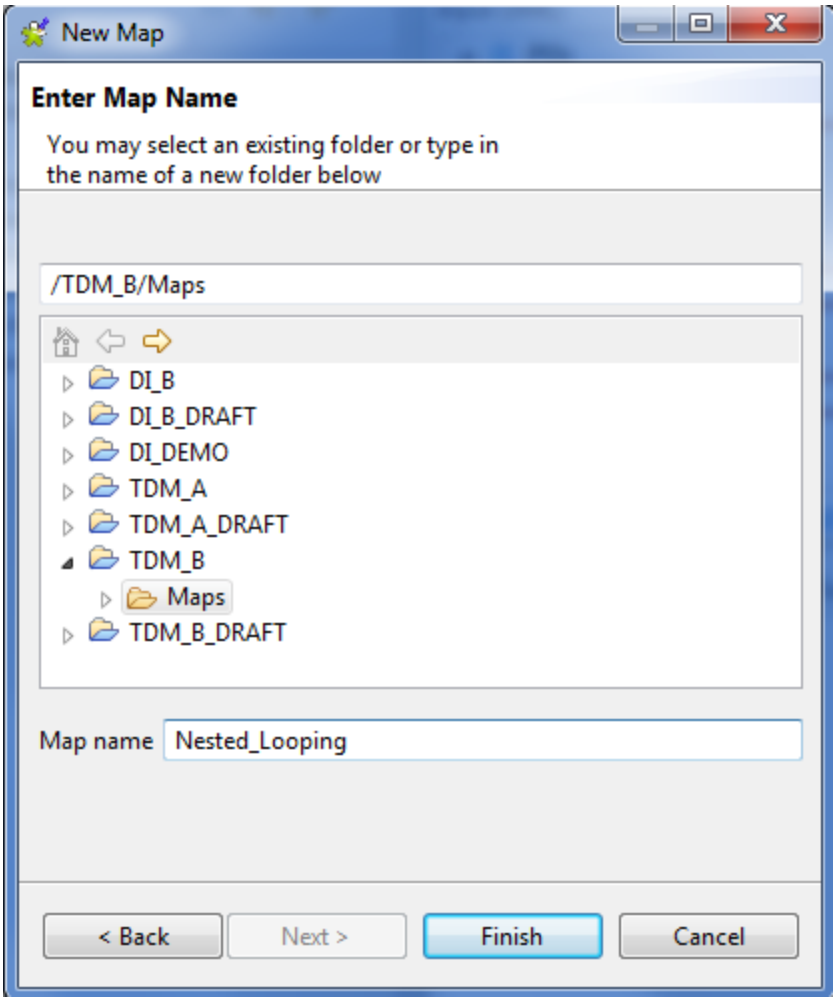
- Set up an Nested Loop (automatically and manually) for elements that require several layers of looping to be accessed

Next Step

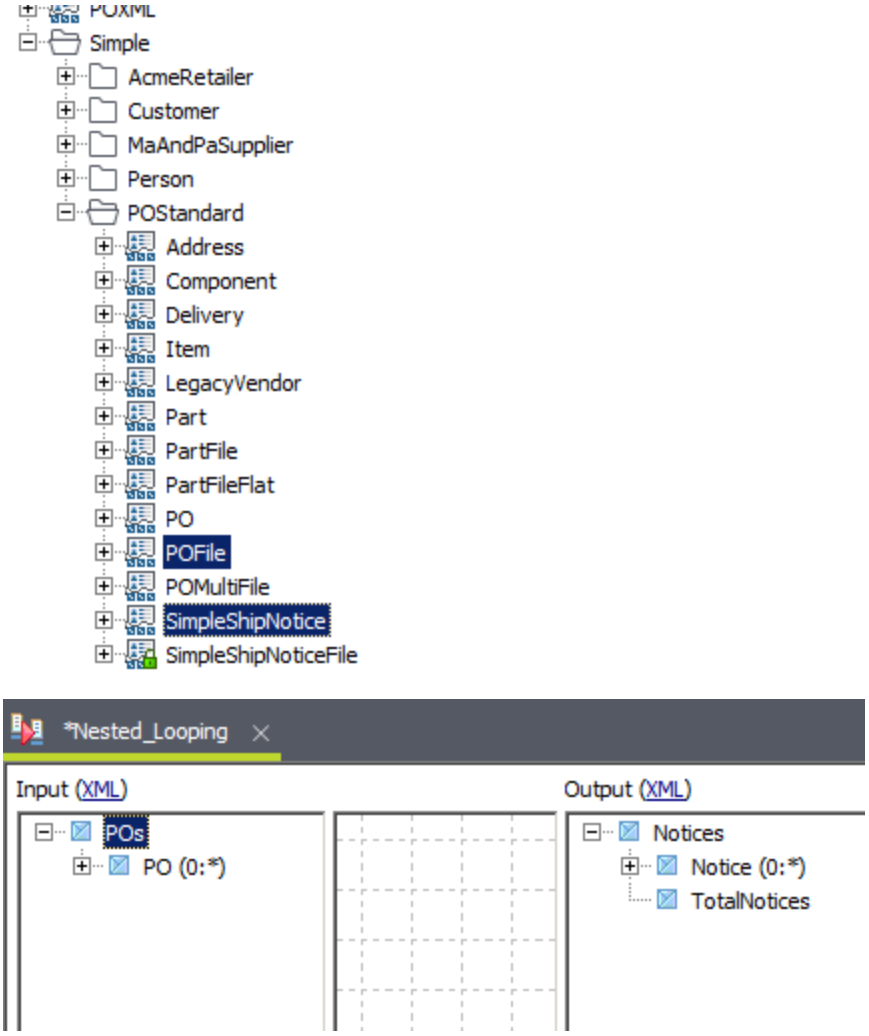
The first step is to create a new Map.

Creating a New Map

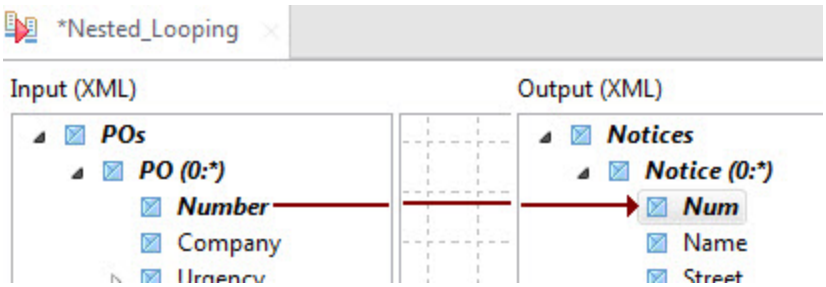
1. Right-click **Hierarchical Mapper > Maps** and create a new **Standard Map** called **Nested_Looping**:



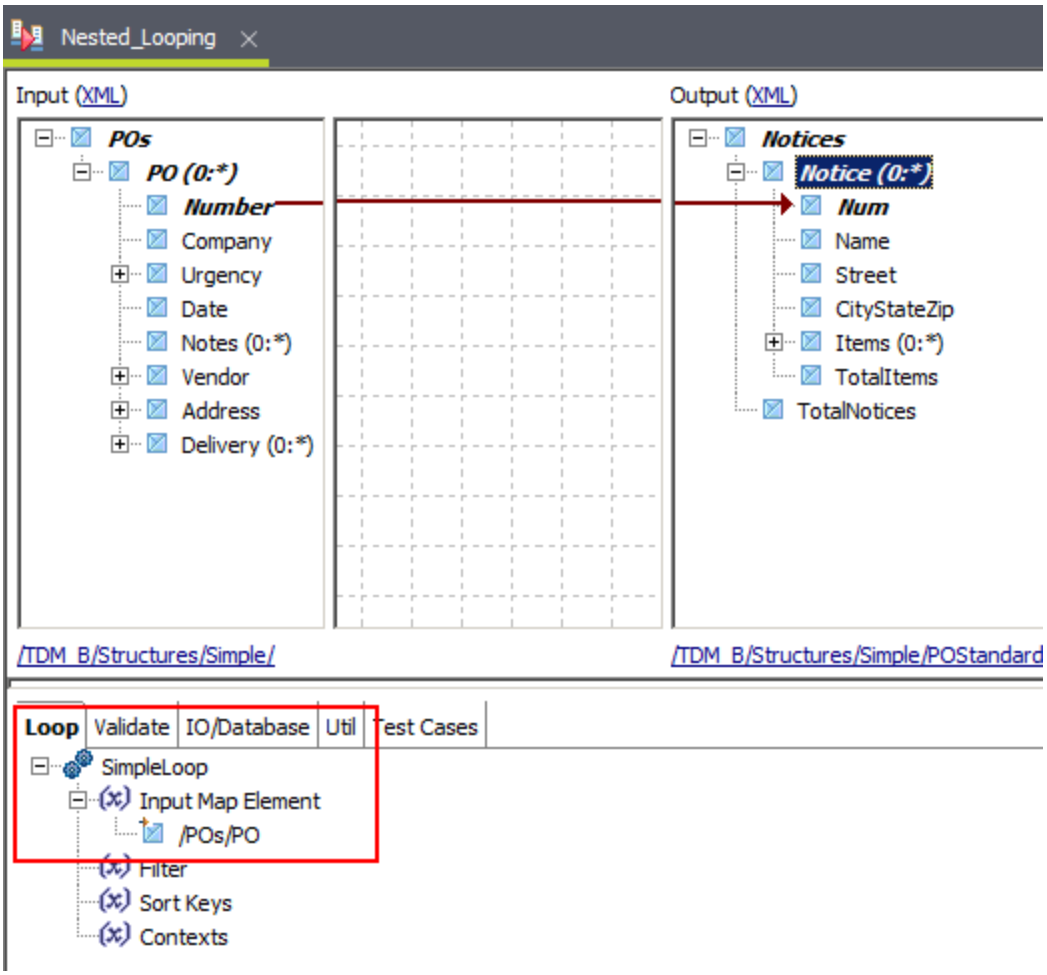
2. Drag **Hierarchical Mapper > Structures > Simple > POStandard > POFile** to the **Input** area.
3. Drag **Hierarchical Mapper > Structures > Simple > POStandard > SimpleShipNoticeFile** to the **Output** area.



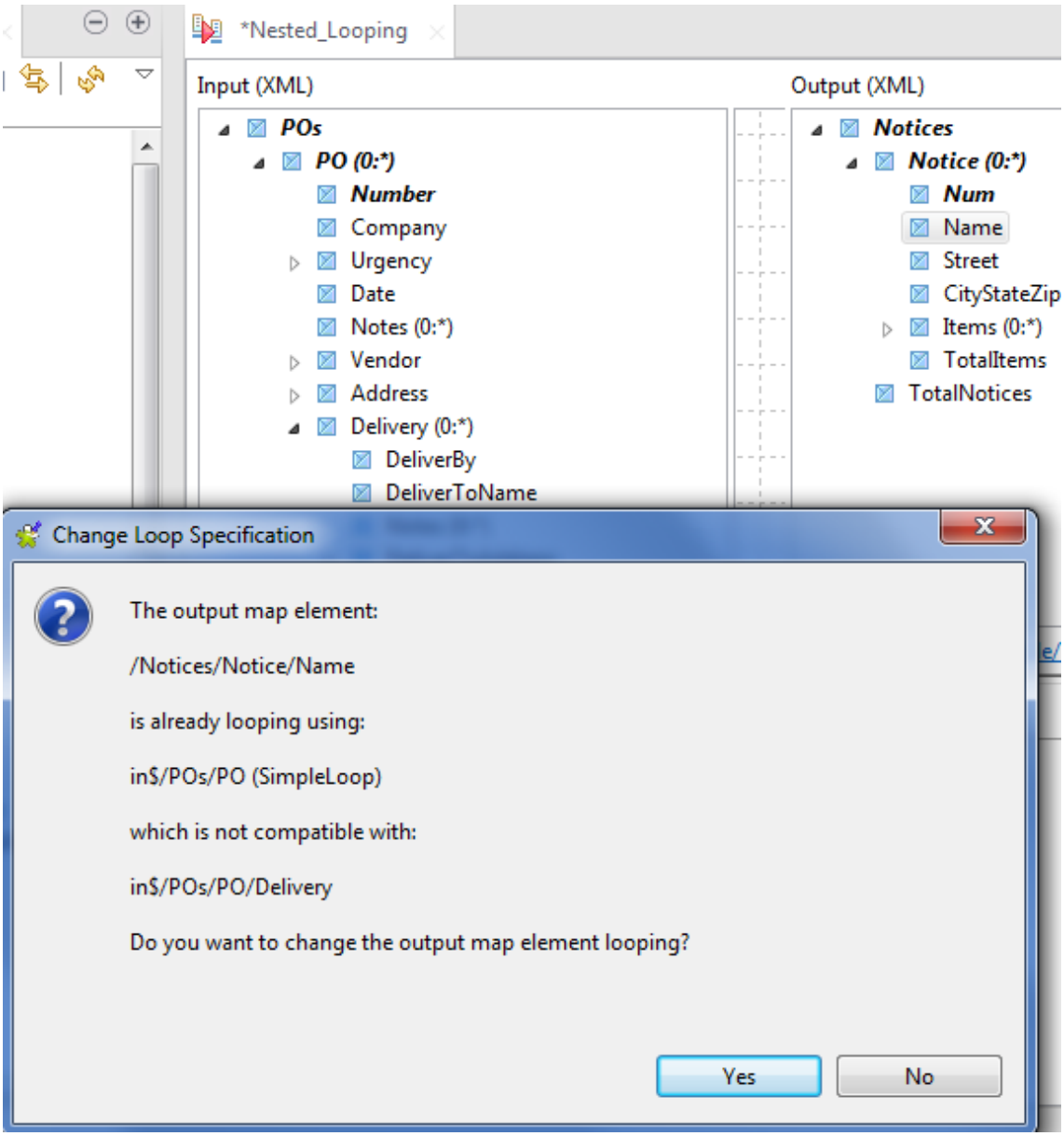
4. Map the **Number** element to the **Num** element.



5. Select the **Notice (0:*)** element in the **Output** area. Notice that the SimpleLoop element was added automatically.

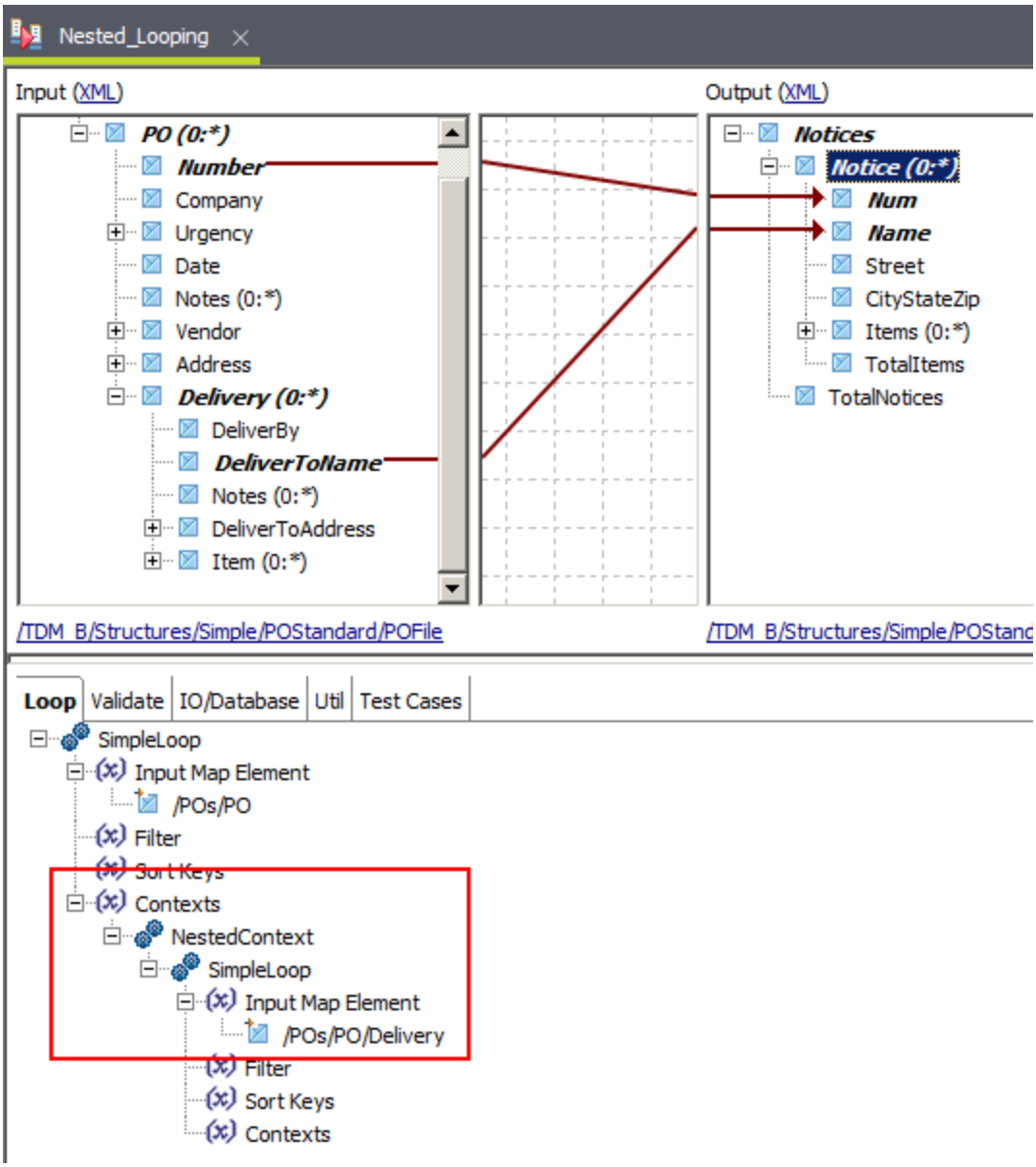


6. Map **POs > PO > Delivery (0:*) > DeliverToName** to **Notices > Notice (0:*) > Name** and take some time to read the message before clicking **Yes**:



This **Change Loop Specification** pop-up window appears to warn you that the new mapping is breaking the original Simple Loop on **Notice(0:*)**. The original loop is no longer compatible because it needs to loop over **PO** and then loop again over **Delivery** to get to the **DeliverToName** element. Clicking **Yes** will add a Simple Loop inside the existing Simple Loop, resulting in what is called a Nested Loop (because two Simple Loops are nested in each other).

7. To check that this Nested Loop was indeed created automatically, select the **Notice(0:*)** element again. Notice the new **SimpleLoop** element in the **Contexts > NestedContext** branch:

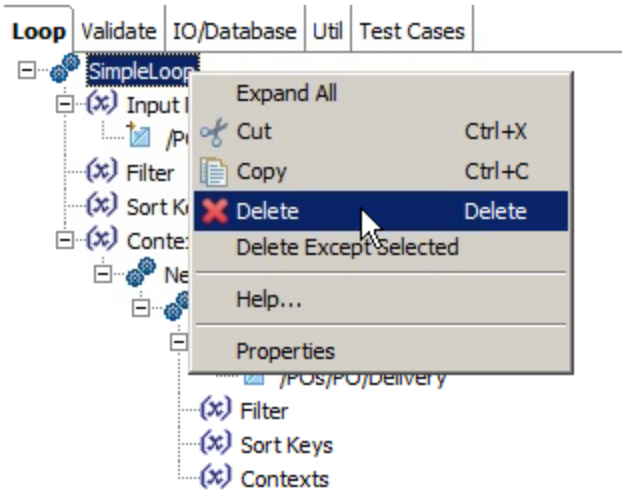


This loop was created automatically when you clicked **Yes** in the pop-up dialog. It loops over **POs > PO** and again over **POs > PO > Delivery**.

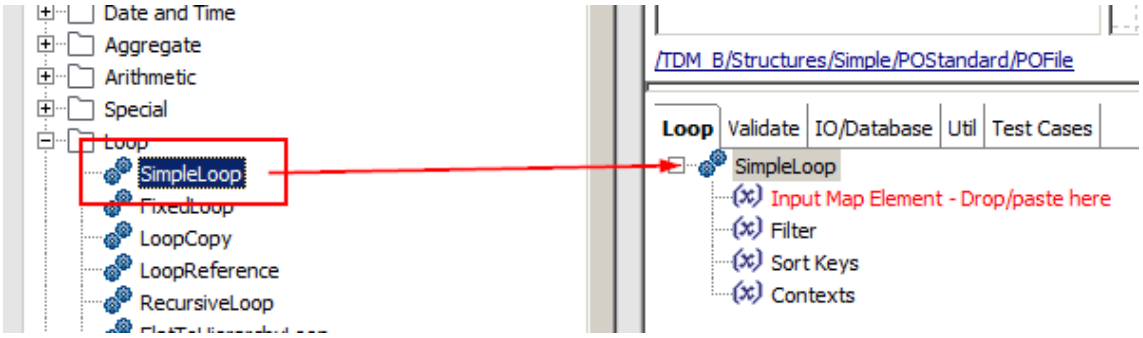
Talend Data Mapper detected this use case and offered to create the Nested Loop automatically. However, in a more general use case, it is possible to create Nested Loops manually. Let's see how to proceed.

Removing the Nested Loop and Adding It Back Manually

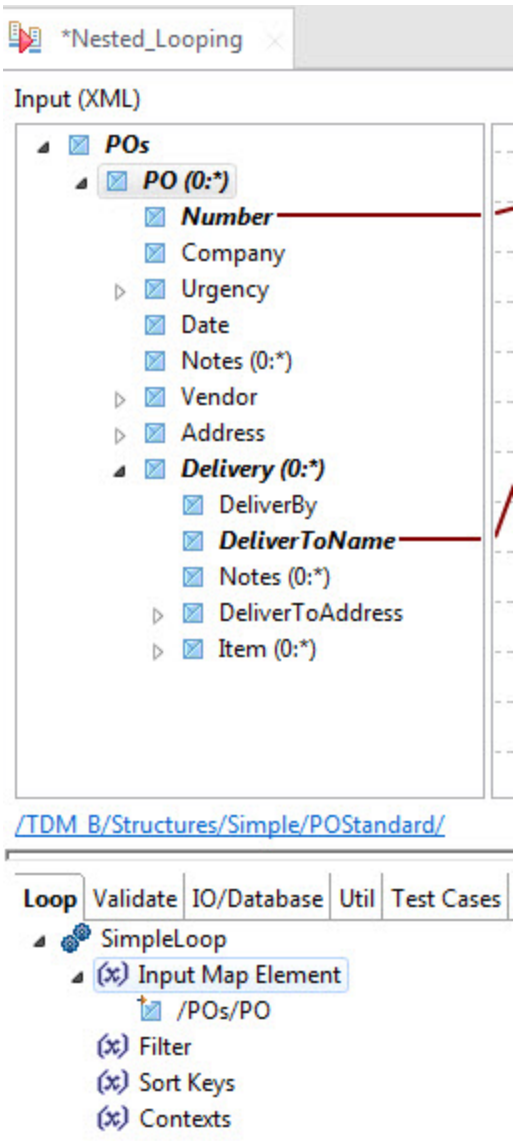
1. To remove the **SimpleLoop** over the **Notice (0:*)**element, right-click the outermost **SimpleLoop** and select **Delete**:



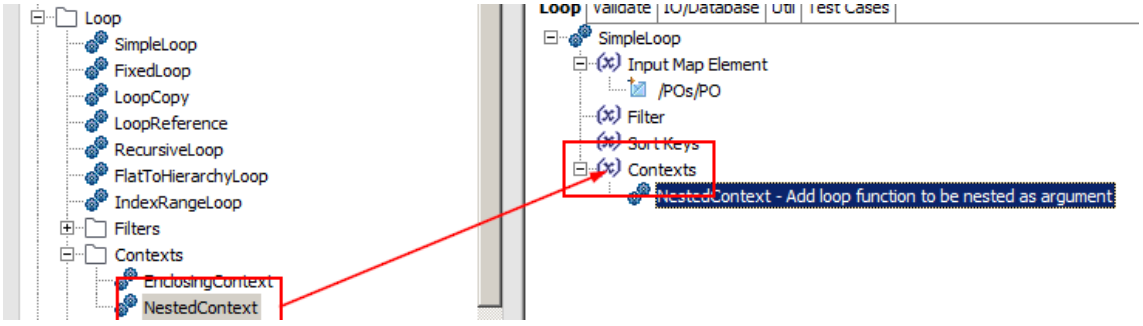
2. Drag a **Functions > Loop > SimpleLoop** function and drop it inside the **Loop** tab to recreate the outermost Simple Loop.



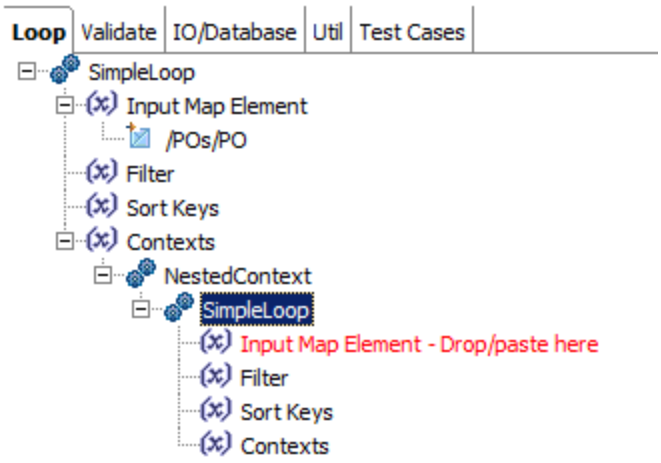
3. Drag and drop the **PO (0:*)** element to the **Input Map Element** field of the **SimpleLoop**:



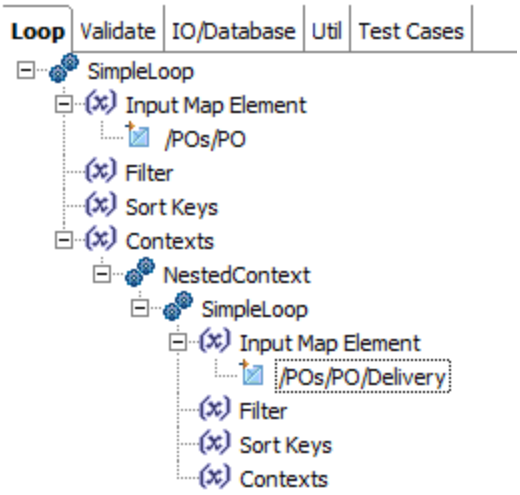
4. Drag a **Functions > Loop > Contexts > NestedContext** function under **SimpleLoop > Contexts**:



5. Drag a second **Functions > Loop > SimpleLoop** function under the **NestedContext** just created:



6. Finally, drag **Delivery (0:*)** to the **Input Map Element** field of the nested **SimpleLoop**:



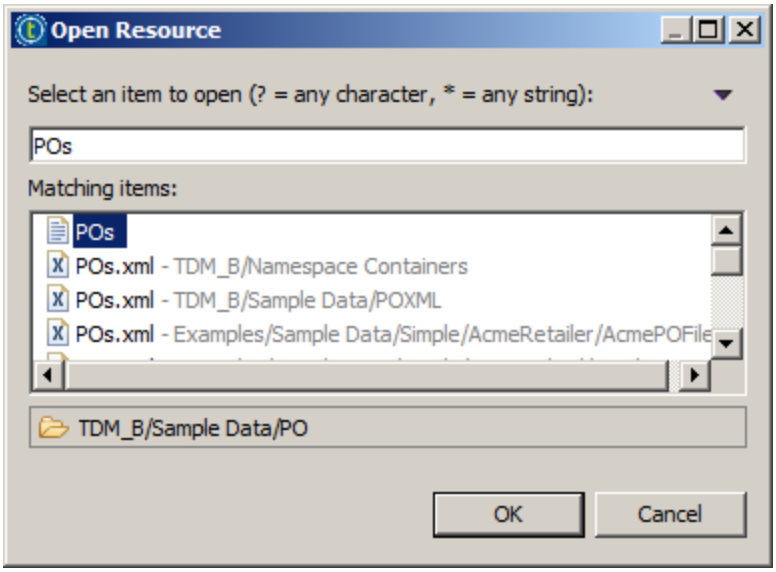
7. The result is identical to the automatic Nested Loop created when you clicked **Yes** in the pop-up dialog.

Next Step

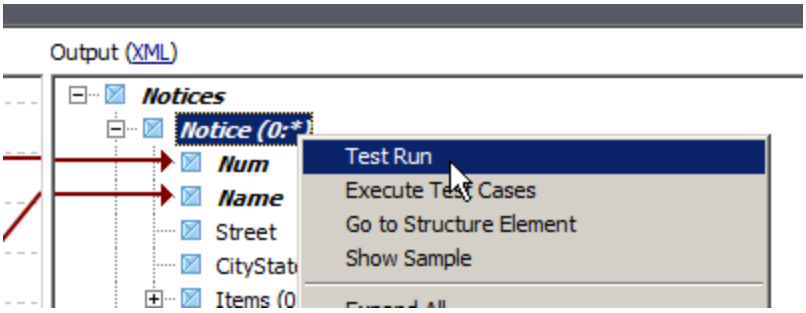
The mapping is done, and Nested Loops were added automatically or manually. Now let's test the Map.

Testing the Map

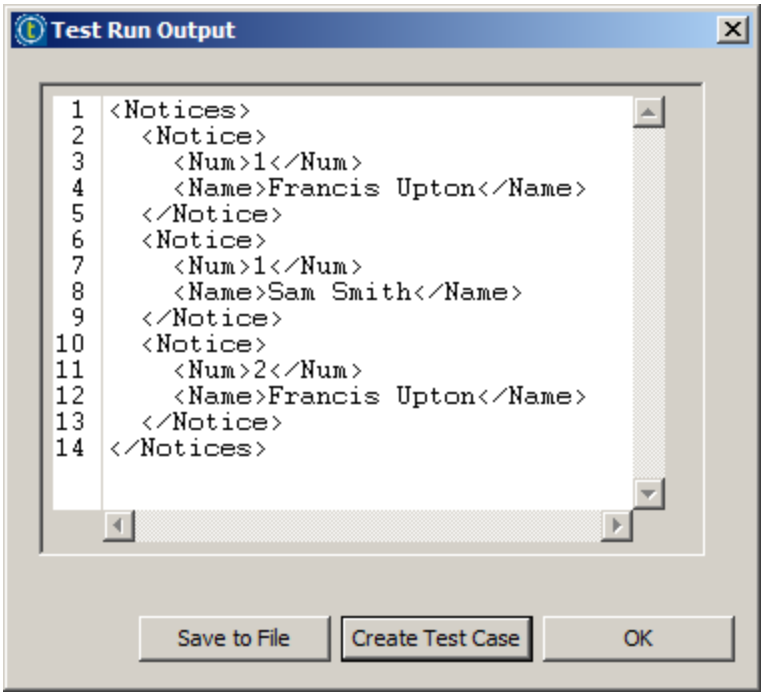
1. Expand the **Show Document** button in the toolbar and select **Select Sample Input Document > Select Resource in Workspace...**.
2. Enter **POs** to filter the available Resources.
3. Select the **POs** item located in the **TDM-Essentials > Sample Data > PO** folder.



4. Right-click the **Notice (0:*)** element and select **Test Run**:



5. Check the results look like the following:



Next Step

This lesson is almost over. Head to the Wrap-Up section for a summary of the concepts reviewed in this lesson.

Wrap-Up

In this lesson, you learned how to:

- Set up an Nested Loop (automatically and manually) for elements that require several layers of looping to be accessed

Next Step

Congratulations, you successfully completed this lesson. Click the **Check your status with this unit** button below in order to save your progress. Then click **Completed. Let's continue >** on the next screen to jump to the next lesson.

This page intentionally left blank to ensure new chapters start on right (odd number) pages.

LESSON 4

Aggregation Looping

This chapter discusses the following.

Overview

Lesson Overview

This lab extends the Simple and Nested Loops exercises and presents the concept of Aggregation Looping.

Objectives

After completing this lesson, you will be able to:

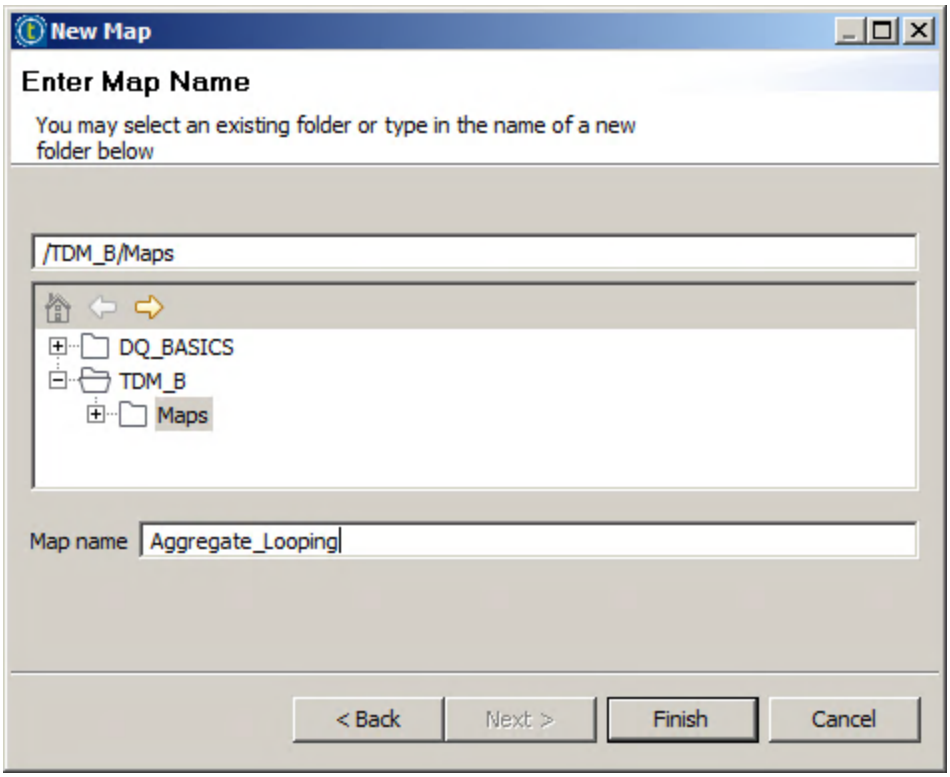
- Set up an Aggregation Loop to count the number of elements of a given entity in the Output

Next Step

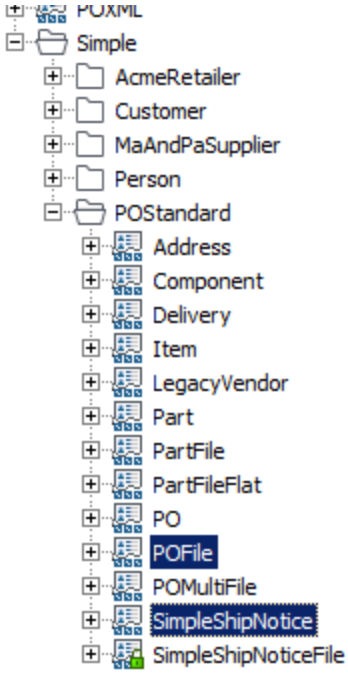
The first step is to create a new Map.

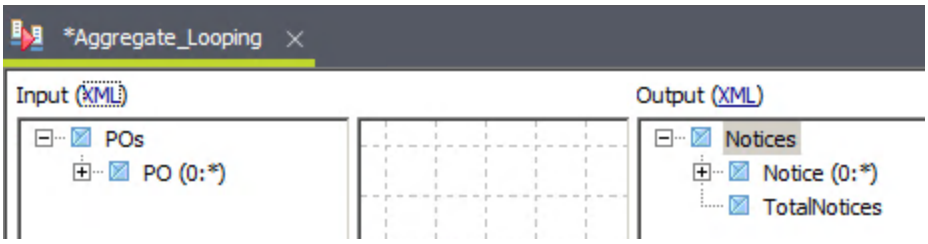
Aggregation Looping

1. Create a new **Standard Map** called **Aggregate_Looping**.

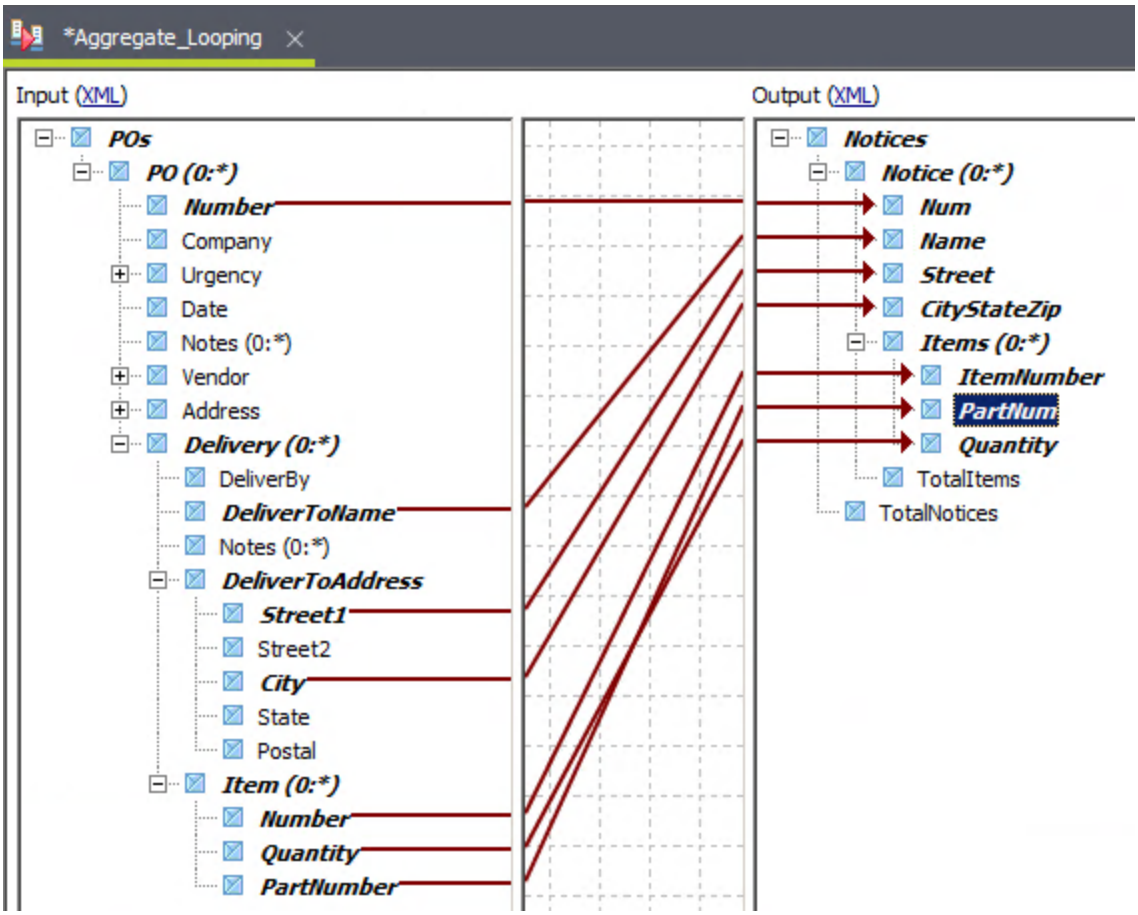


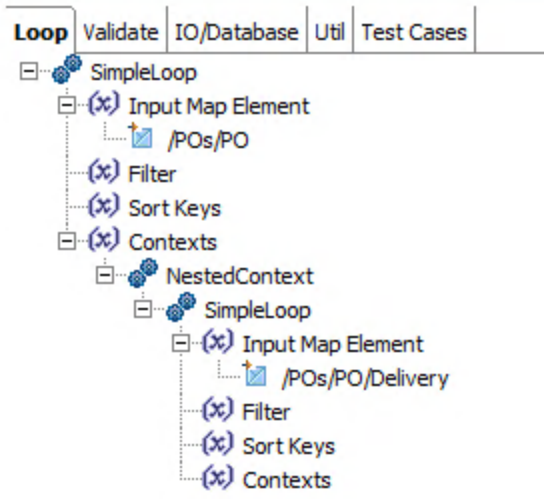
2. Drag **Hierarchical Mapper > Structures > Simple > POStandard > POFile** to the **Input** area.
3. Drag **Hierarchical Mapper > Structures > Simple > POStandard > SimpleShipNoticeFile** to the **Output** area.





4. Map the following elements, click **Yes** in **Change Loop Specification** dialogs when needed and check the loop expression for **Notice (0:*)** at the end. Note that a Nested Loop was created automatically for you.

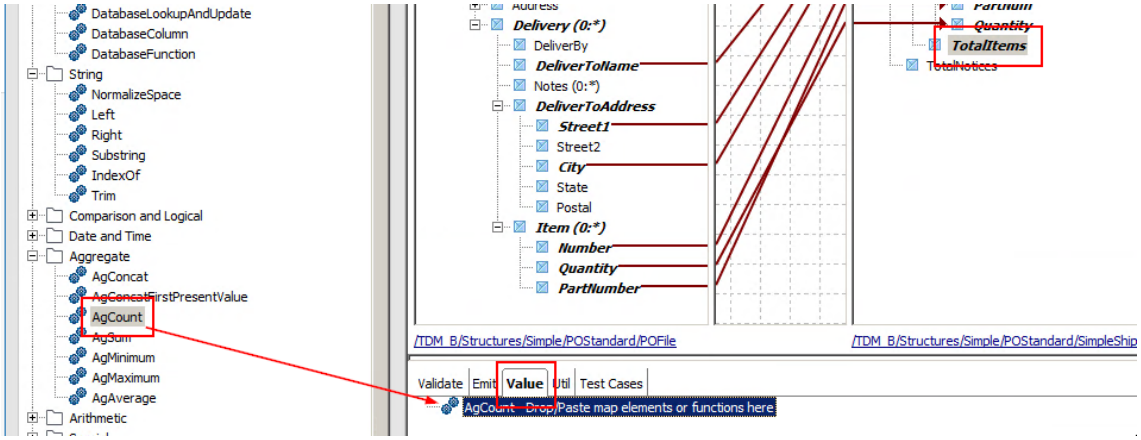




Setting Up Aggregation for Total items

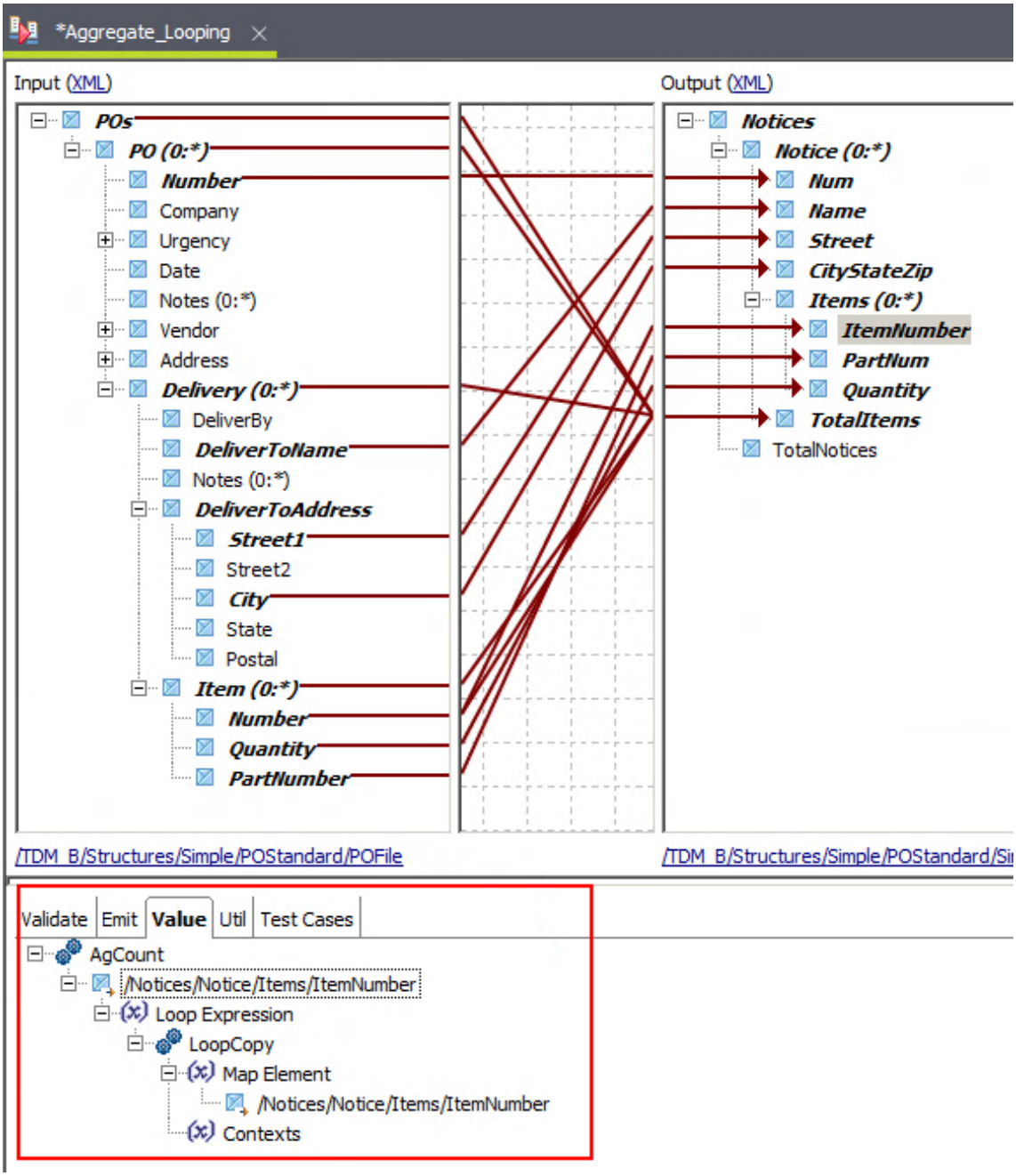
Suppose you need to count the total number of items per notice. This information is not available directly in the Input data. Aggregation can help by applying simple mathematical functions like count, sum, minimum... to the nodes.

1. Set up the aggregation by selecting the **TotalItems** node in the **Output** area. Then drag a **Functions > Aggregate > AgCount** function to the **Value** tab.



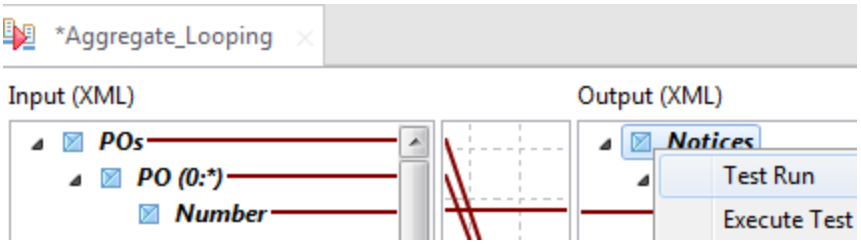
2. Drag the **Items(0:*) > ItemNumber** element from the Output area to the **AgCount** function. A **SimpleLoop** expression

will be created automatically based on the selected element:

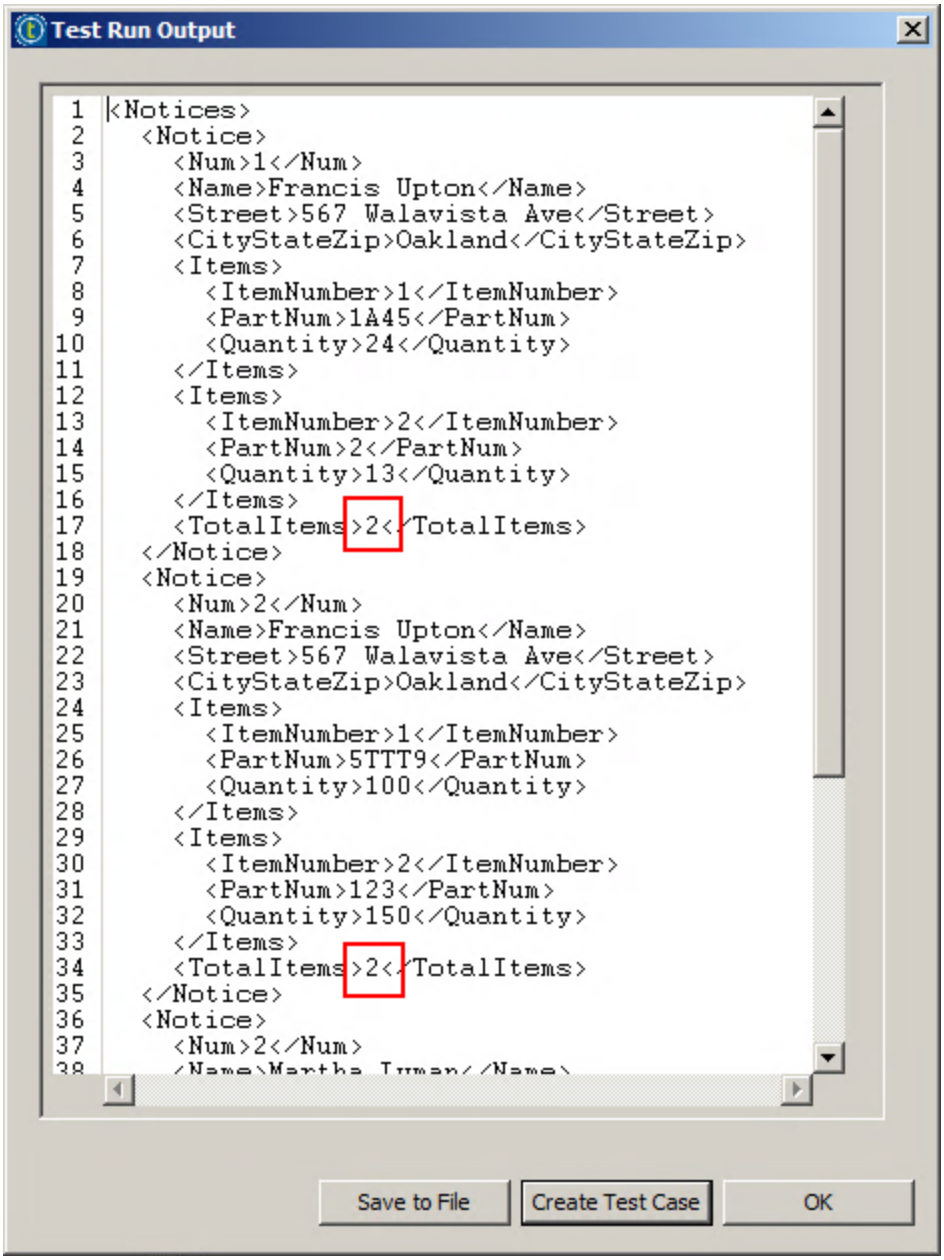


Testing

1. Right-click **Notices** in the **Output** area and select **Test Run**.



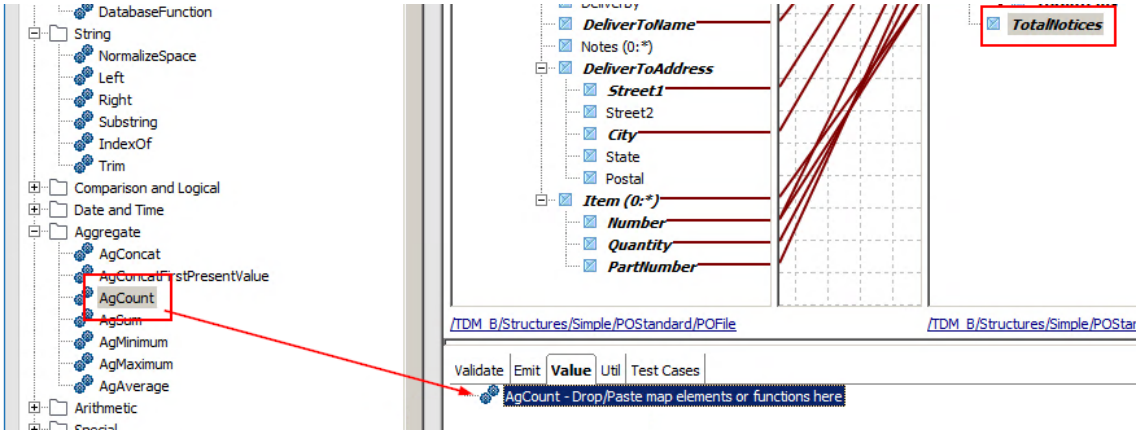
2. Review the results. Each Purchase Order should be associated with two Items.



Setting Up Aggregation for Total Notices

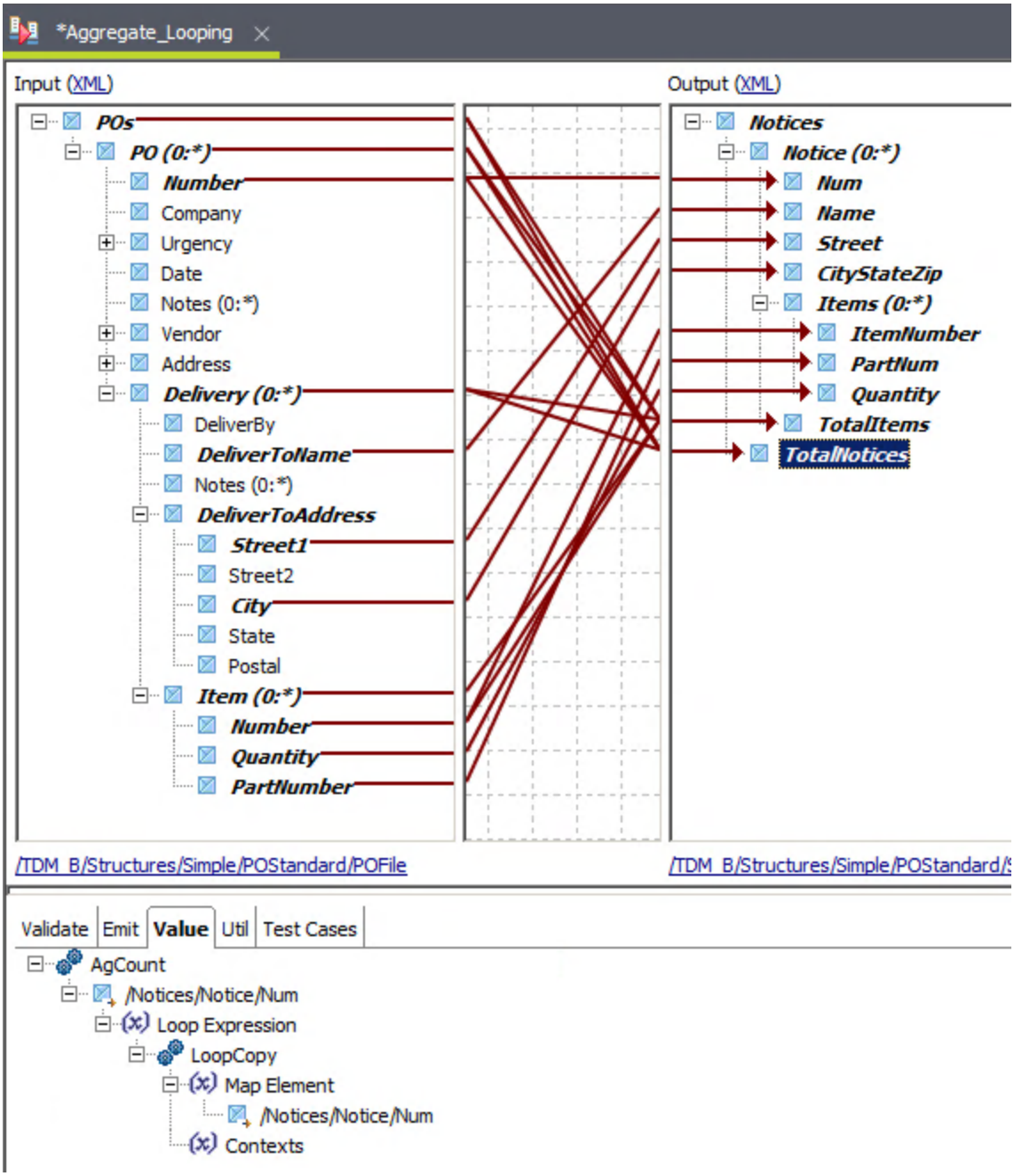
Let's apply the same principle to count the total number of Notices per document.

1. Set up the aggregation by selecting the **TotalItems** node in the **Output** area. Then drag a **Functions > Aggregate > AgCount** function to the **Value** tab.



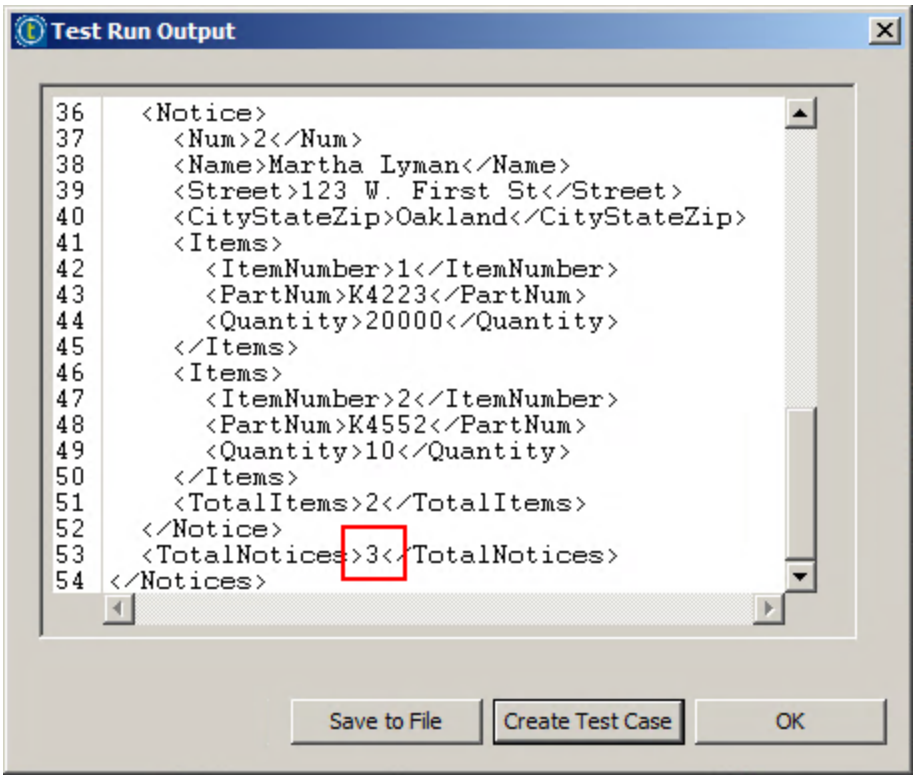
2. Drag the **Notice(0:*) > Num** element from the Output area to the **AgCount** function. A **SimpleLoop** expression will be cre-

ated automatically based on the selected element:



Testing

1. Right-click **Notices** in the **Output** area and select **Test Run**.
2. Review the results. There should be 3 Notices in the XML output.



Next Step

This lesson is almost over. Head to the Wrap-Up section for a summary of the concepts reviewed in this lesson.

Wrap-Up

In this lesson you learned how to:

- Set up an Aggregation Loop to count the number of elements of a given entity in the Output

Next Step

Congratulations, you successfully completed this lesson. Click the **Check your status with this unit** button below in order to save your progress. Then click **Completed. Let's continue >** on the next screen to jump to the next lesson.

This page intentionally left blank to ensure new chapters start on right (odd number) pages.

LESSON 5

Using More Functions

This chapter discusses the following.

Overview

Lesson Overview

After duplicating the *Lab01_XML2XML* Map created in a previous exercise, you will learn how to use different types of Functions, such as **Concatenation** and **If-Then-Else** logic.

The **Concatenation** Function will be used to add the word **Company** to the *Company* input node. For example, *"Fargo"* will turn into *"Fargo Company"*.

The **If-Then-Else** Function will be used to make decisions based on the value of one or several existing nodes. For example, the value of the *Number* node will be used to decide whether to add *" Company"* or *" CO."* to the *Company* input node.

Objectives

After completing this lesson, you will be able to:

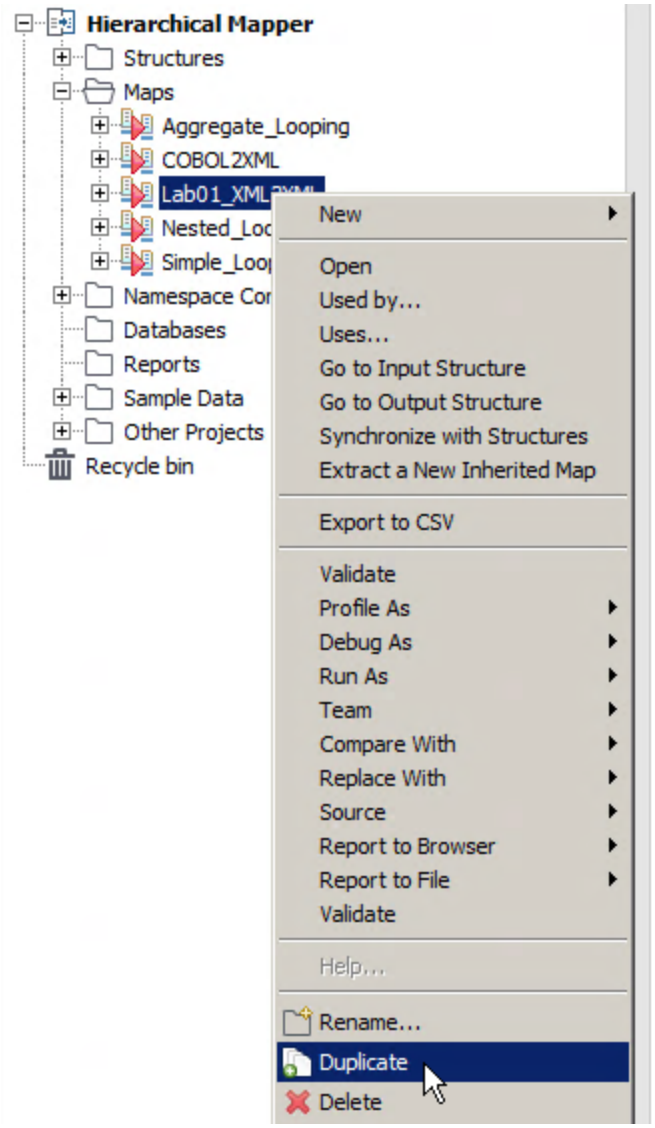
- Duplicate an existing Map
- Use a Concatenation Function to alter the output of a Map, like adding a *" Company"* suffix to the *Company* input node
- Use an If-Then-Else Function to make decisions based on the value of one or several existing nodes. For example, you used the *Number* value to decide whether to add *" Company"* or *" CO."* to the *Company* input node.

Next Step

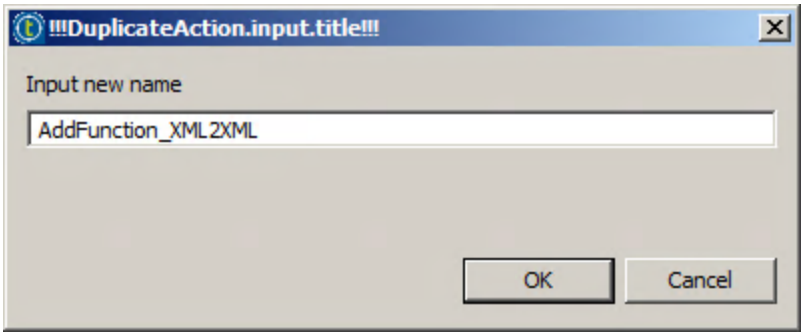
Let's start with using a Concatenation Function to add the **Company** word to the *Company* field.

Using a Concatenation Function

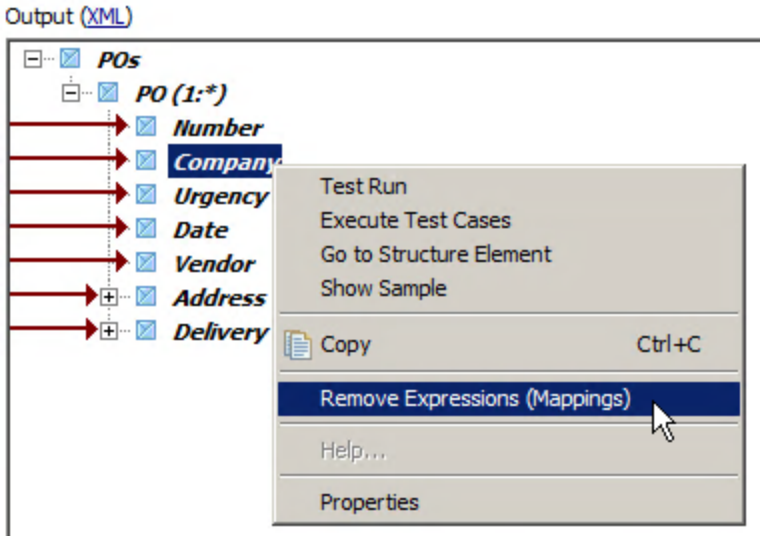
1. First, let's duplicate the *Lab01_XML2XML* Map created in a previous exercise, so you do not have to start from scratch. Right-click on **Hierarchical Mapper > Maps > Lab01_XML2XML** and select **Duplicate**:



2. Enter **AddFunction_XML2XML** for the new name and click **OK**:

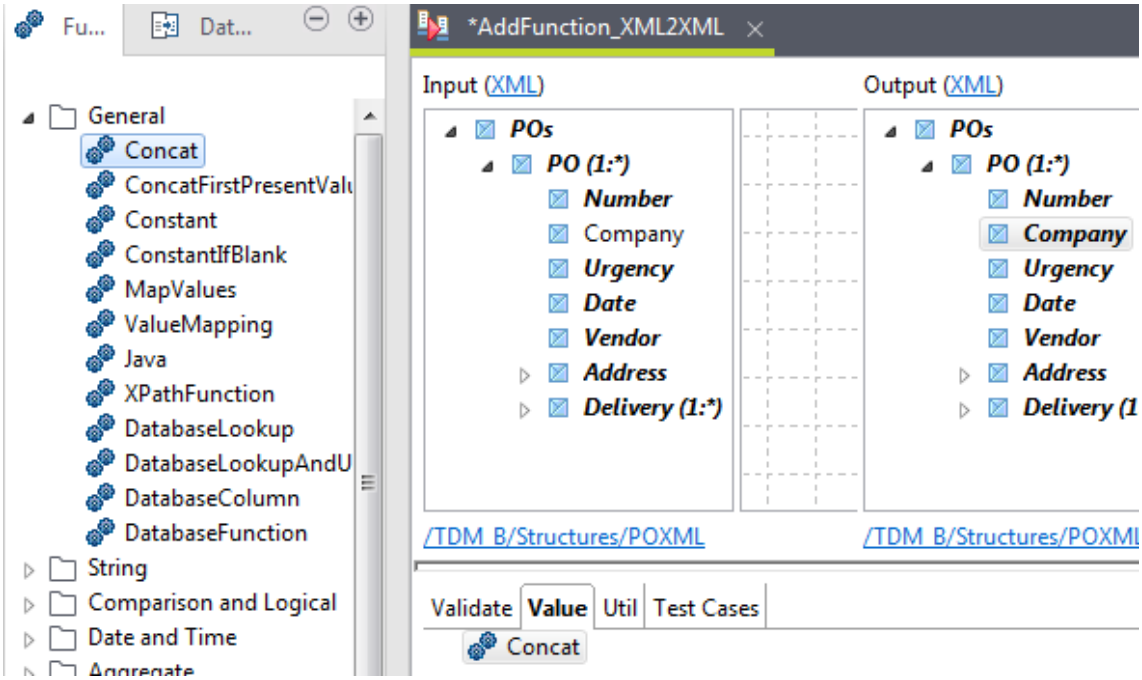


3. Open the new **AddFunction_XML2XML** Map.
4. Right-click the output element **Company** then click **Remove Expressions (Mapping)**.

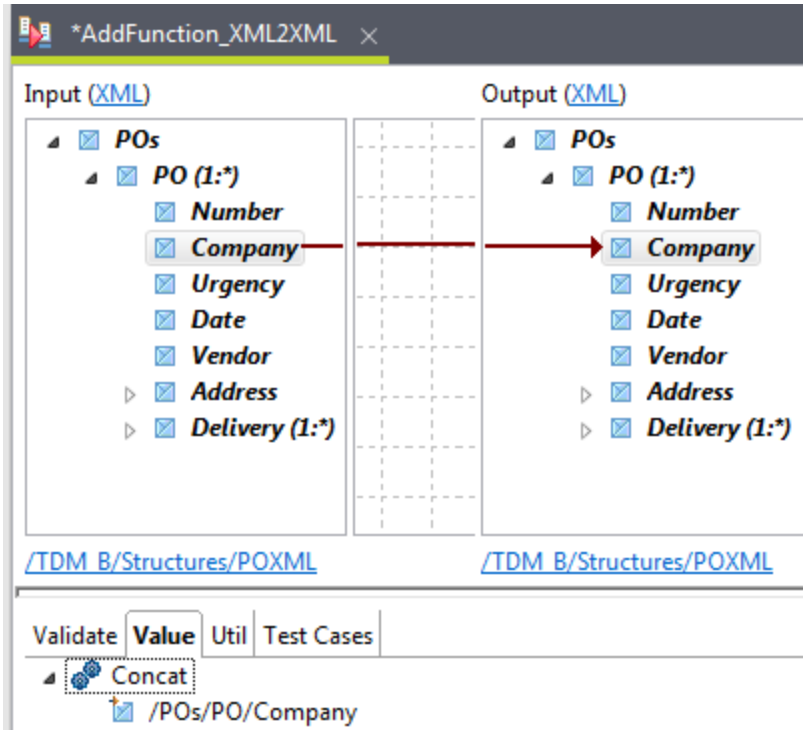


5. Notice the Value tab is now empty for this particular node.

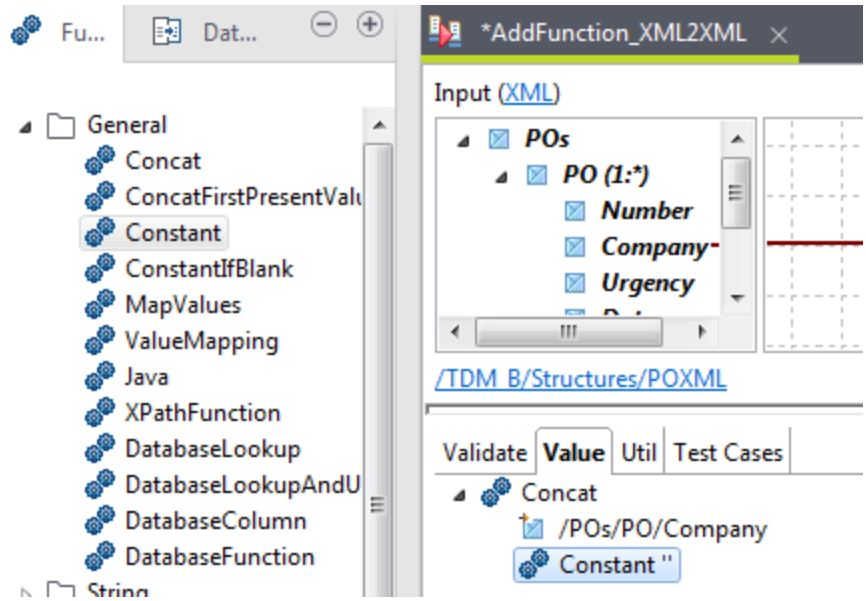
6. Drag a **Functions > General > Concat** Function and drop it to the **Value** tab:



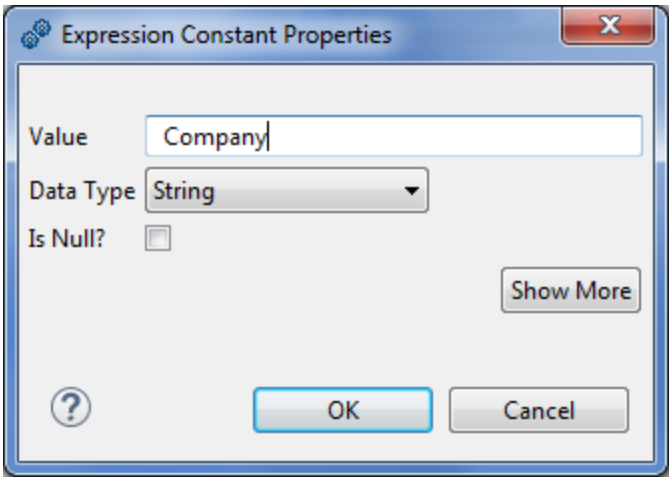
7. Drag the **Company** element from the **Input** area to the **Concat** function in the **Value** tab:



8. Now let's add the word "*Company*" to the Concatenation expression. Drag a **Functions > General > Constant** Function to the **Value** tab, below the **Company** element (until a black horizontal line appears) and then drop it:

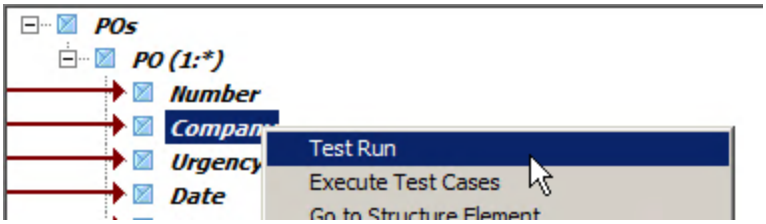


9. Double-click the **Constant** Function in the **Value** tab. Enter **Company** in the Value field and click **OK**

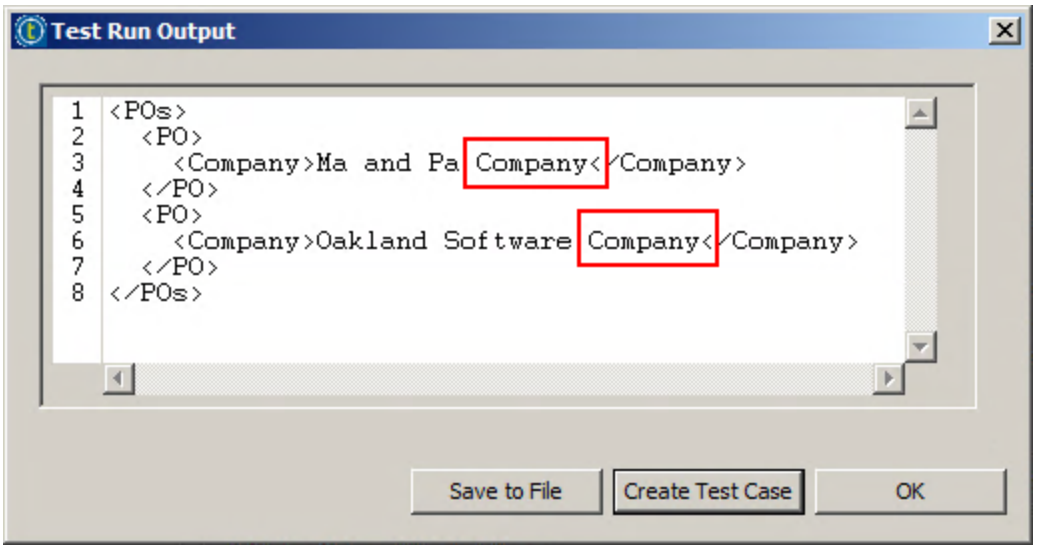


Note the space before **Company**. Without this extra space at the beginning, *"Fargo"* would turn into *"FargoCompany"* instead of the desired *"Fargo Company"*.

10. Perform a **Test Run** on the **Company** element in the **Output** area and examine the output.



11. Notice that each **Company** element ends with *" Company"*:

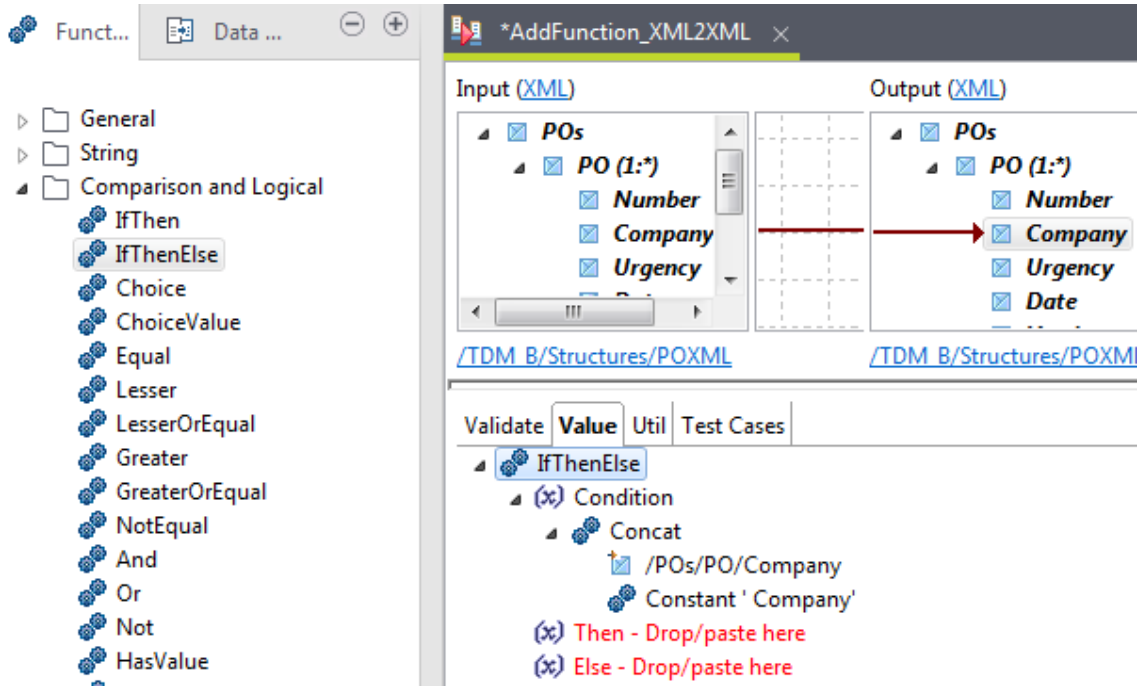


Next Step

Now imagine you want to use a *Company* suffix for the first PO only, and a *Co.* suffix for every other PO. This can be achieved by adding an If/Then/Else condition in order to change the behavior of the Concatenation Function based on the PO Number value.

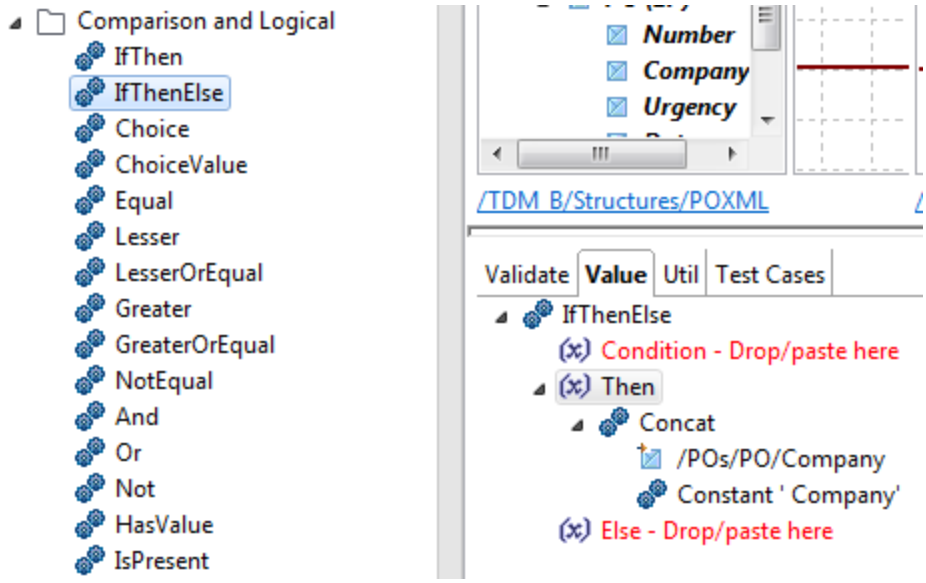
Using an If/Then/Else Condition

1. Drag a **Functions > Comparison and Logical > IfThenElse** Function to the **Value** tab just above the **Concat** Function until a thick black horizontal line appears, and then drop it:

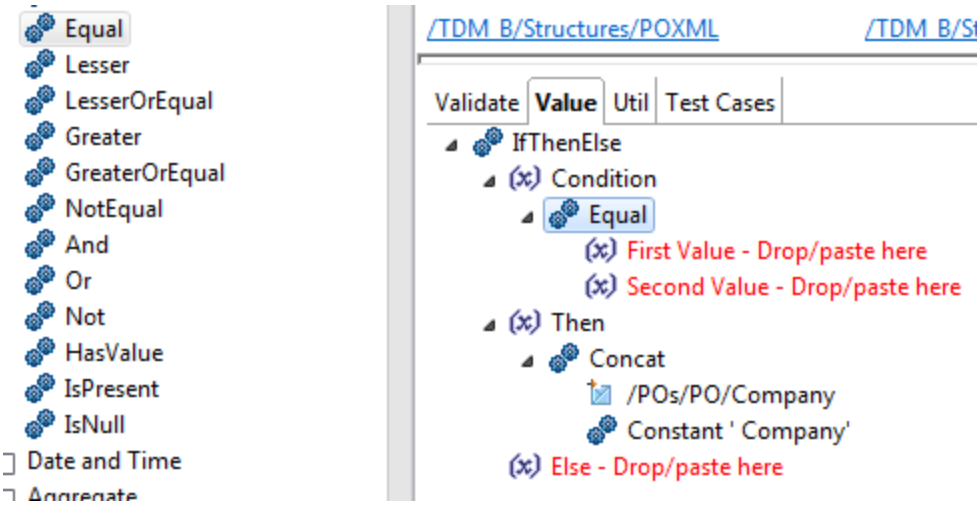


Note that the **Concat** Function is under the **Condition** branch by default. Let's move it to the **Then** branch, as **Concat** is the action to be performed rather than the test itself.

2. Select the **Concat** function and drag it directly on top of the **Then** branch:



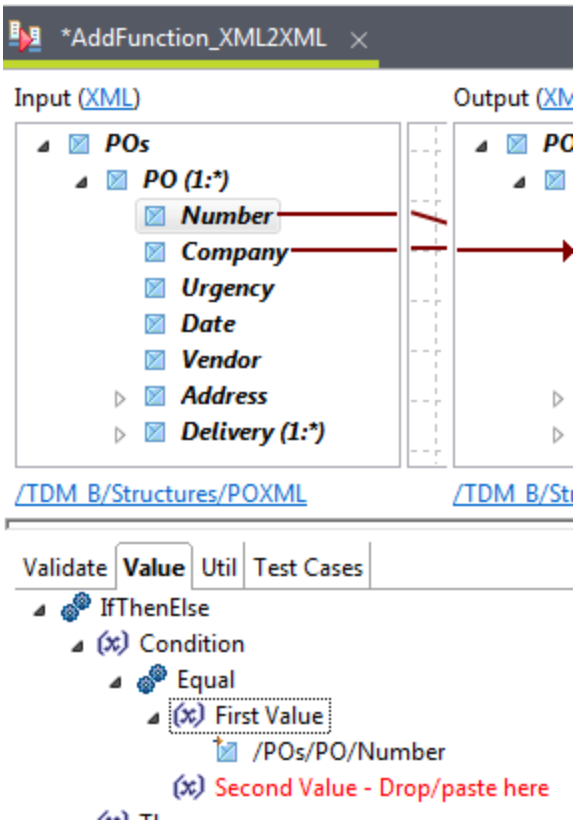
3. Drag a **Functions > Comparison and Logical > Equal** Function to the **Condition** branch:



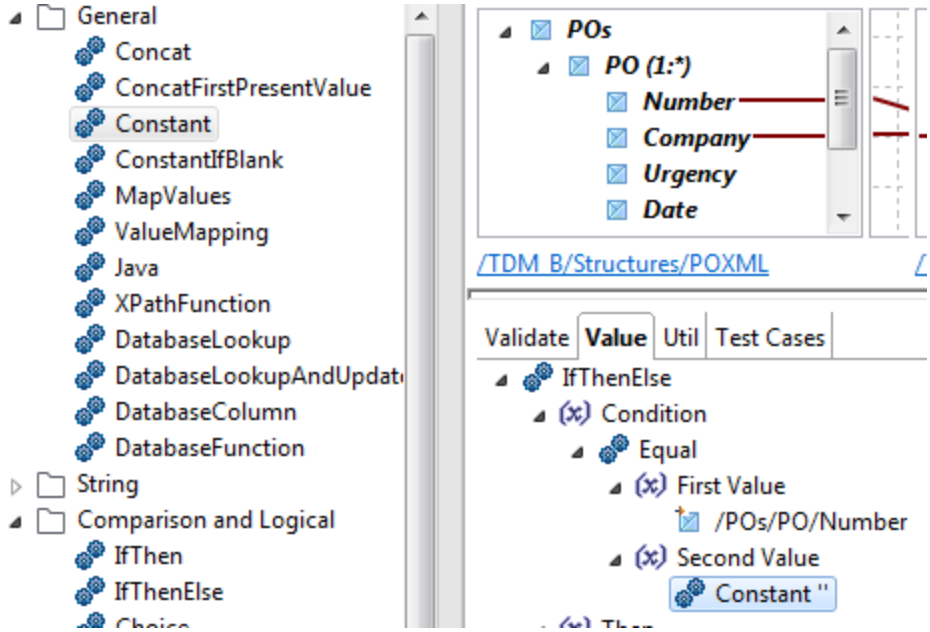
Now let's define and apply the actual test. It can be expressed as the following pseudo code:

```
if Number == 1 then
    concatenate with ' Company'
else
    concatenate with ' Co.'
```

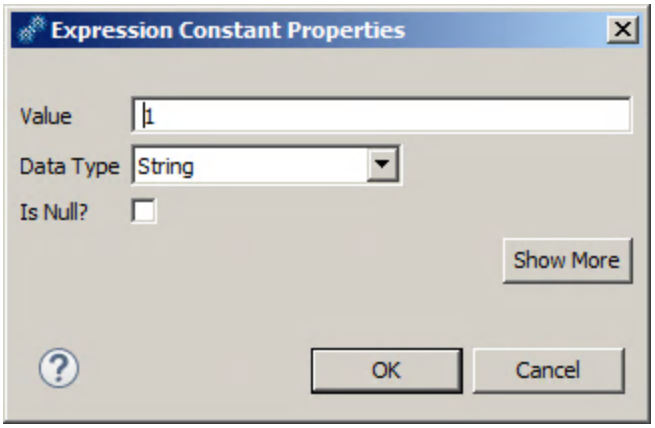
4. Drag the **Number** element from the **Input** area to the **First Value** field of the **Equal** branch.



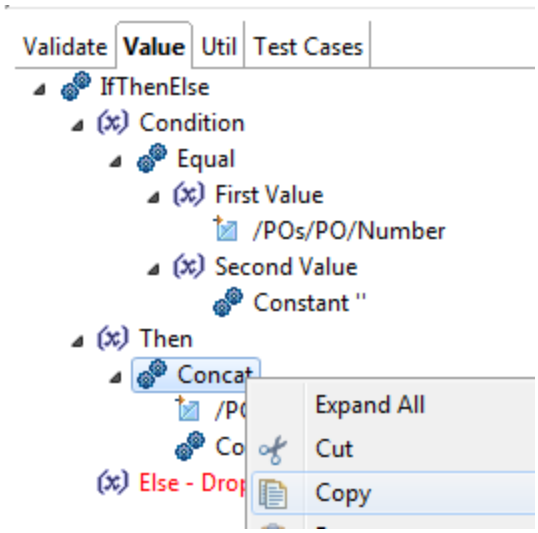
5. Drag a **Functions > General > Constant** Function to the **Second Value** field of the **Equal** branch.



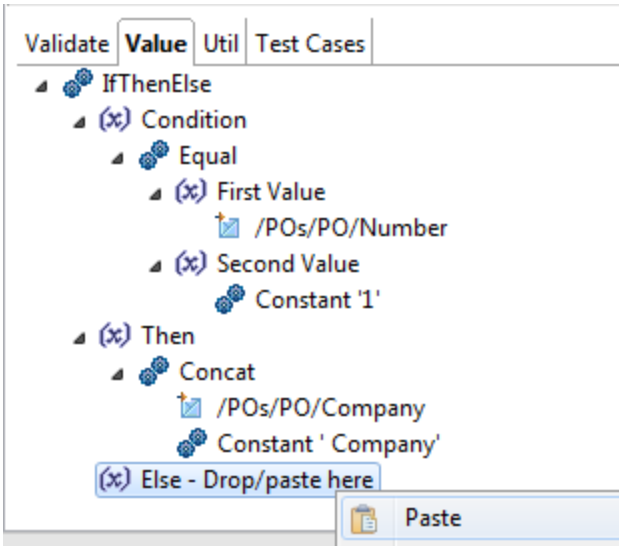
6. Double-click the **Constant** Function to edit its value, enter *1* and click **OK**.



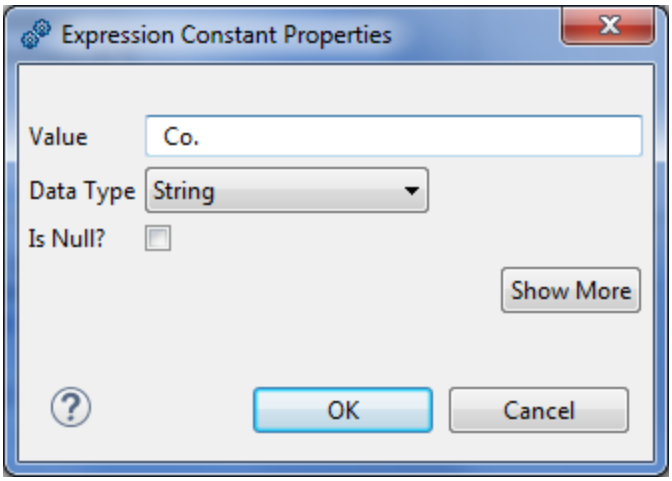
7. To make things easier and avoid redefining a **Concat** action from scratch in the **Else** branch, right-click the **Concat** function from the **Then** branch and click **Copy**:



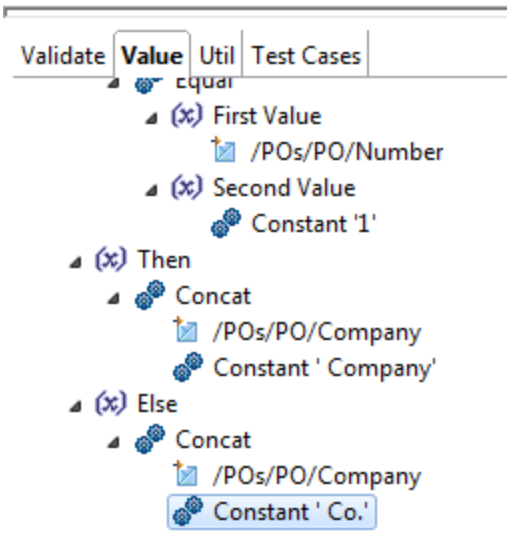
8. Right-click the **Else** branch and click **Paste**:



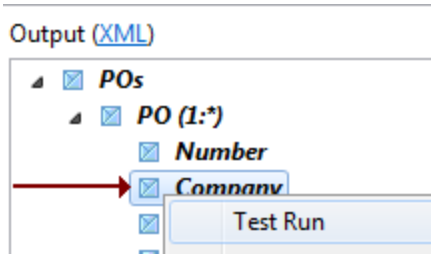
9. Double-click the **Constant** in the **Else** branch, enter *Co.* (with a leading space) and click OK.



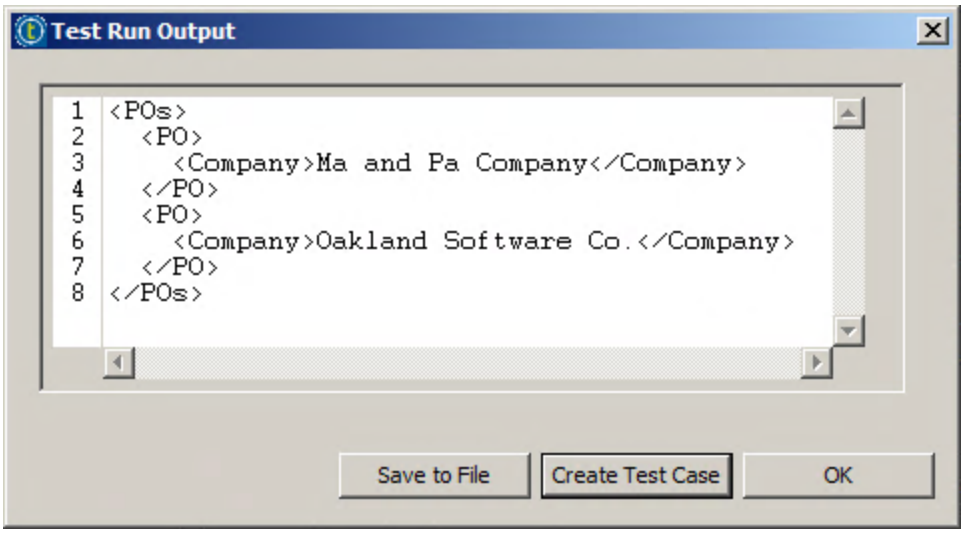
10. Check that the completed function looks like this.



11. Right-click **Company** in the **Output** area and select **Test Run**.



12. Check that the first PO ends with *Company*, and the second one ends with *CO*.



Next Step

This lesson is almost over. Head to the Wrap-Up section for a summary of the concepts reviewed in this lesson.

Wrap-Up

In this lesson, you learned how to:

- Duplicate an existing Map
- Use a Concatenation Function to alter the output of a Map, like adding a Company suffix to the Company input node
- Use an If-Then-Else Function to make decisions based on the value of one or several existing nodes. For example, you used the Number value to decide whether to add Company or CO. to the Company input node.

Next Step

Congratulations, you successfully completed this lesson. Click the **Check your status with this unit** button below in order to save your progress. Then click **Completed. Let's continue >** on the next screen to jump to the next lesson.

This page intentionally left blank to ensure new chapters start on right (odd number) pages.

LESSON 6

Using the tHMap Component

This chapter discusses the following.

Overview

Lesson Overview

The **tHMap** Component can be used in Data Integration (DI) Jobs to execute transformations between different sources and destinations by harnessing the capabilities of *Talend Data Mapper*, available in the **Mapping** perspective.

Objectives

After completing this lesson, you will be able to:

- Use the tHMap Wizard to create a Map interactively, from Schemas and Structures already available or imported manually

Next Step

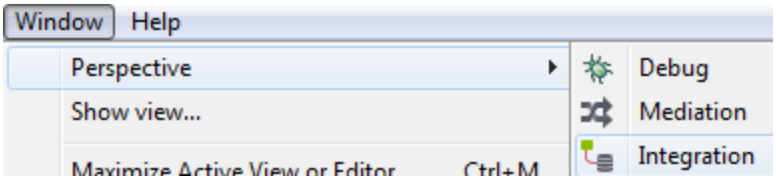
Let's learn how to use the tHMap Component in the **Integration** perspective.

Using the tHMap Component

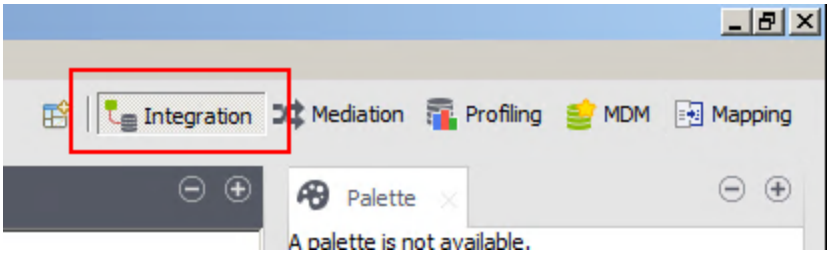
Importing Data

1. First, let's create some data to operate on.

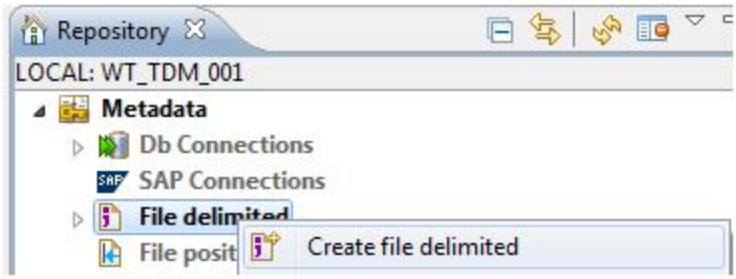
   Click **Window > Perspective > Integration** to switch to the **Integration** perspective and be able to create a Data Integration Job.



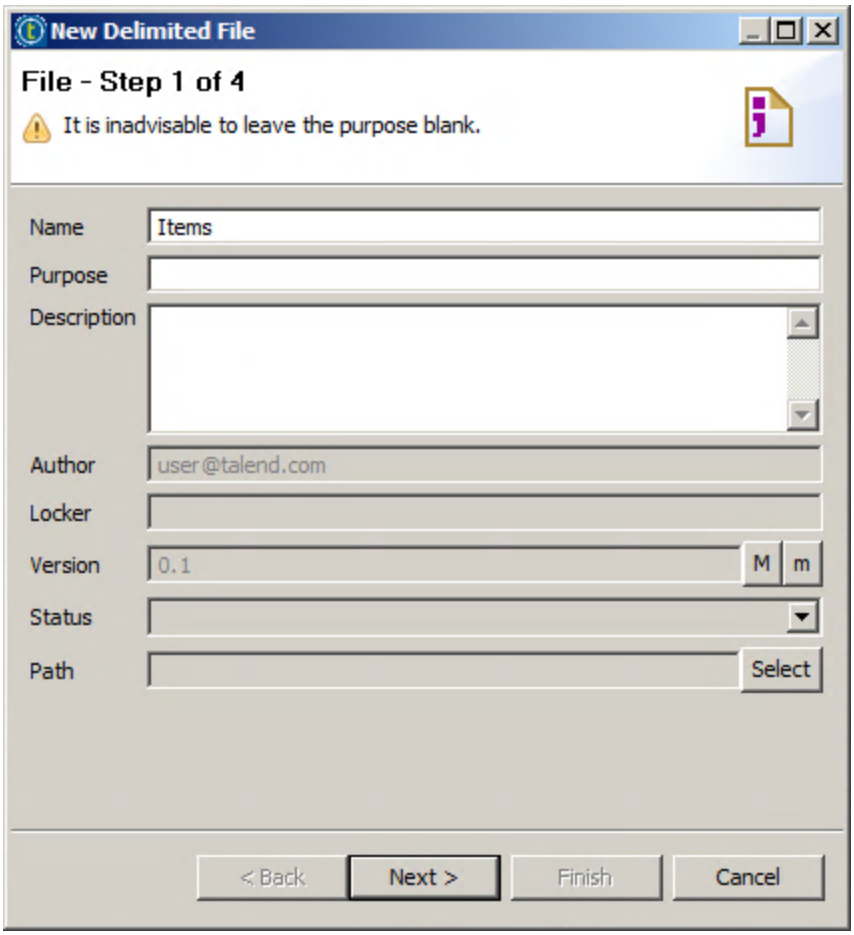
   Note that you can also click the **Integration** button at the top right of the Studio to achieve the same result.



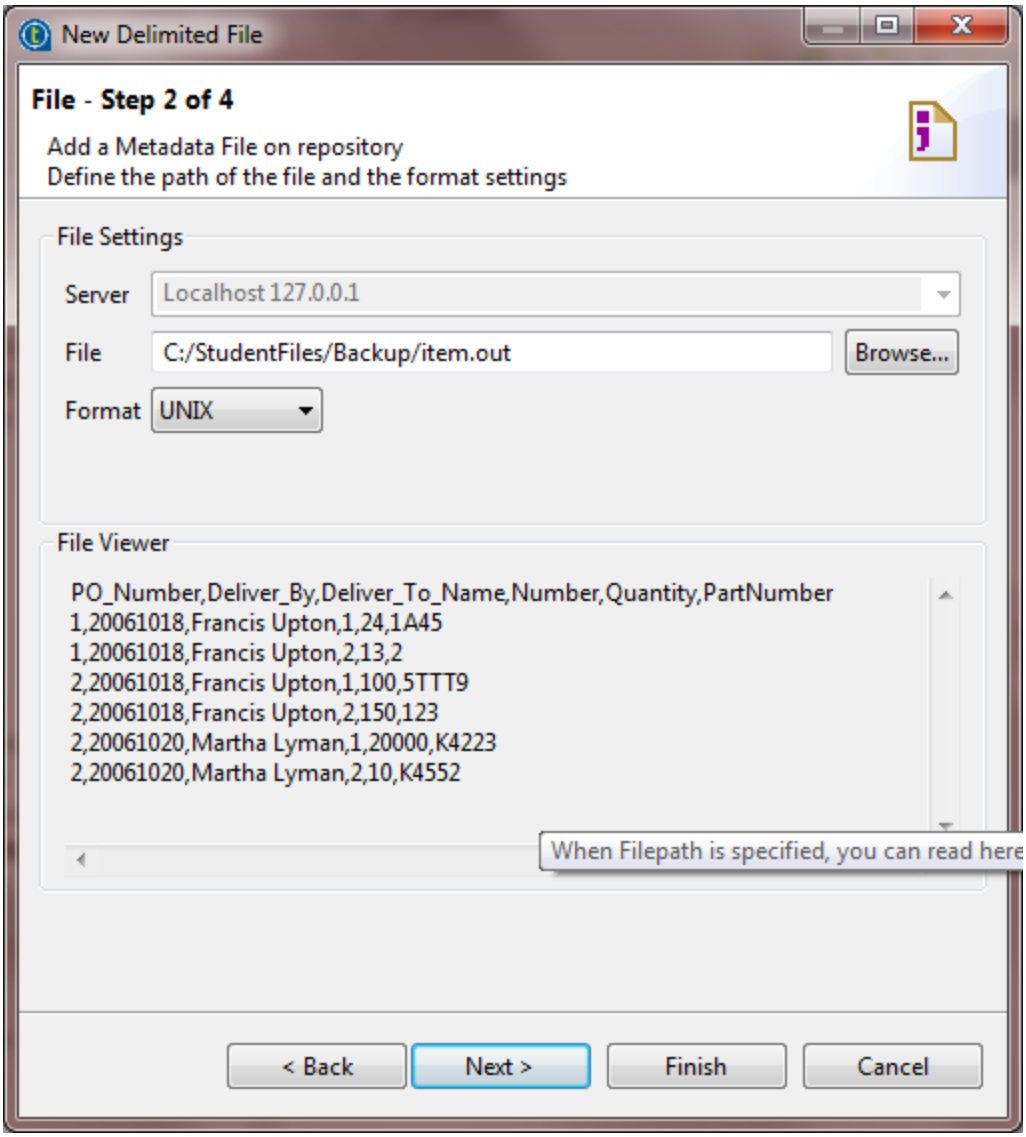
2. In the **Integration** perspective, right-click **Metadata > File delimited** and select **Create file delimited**.



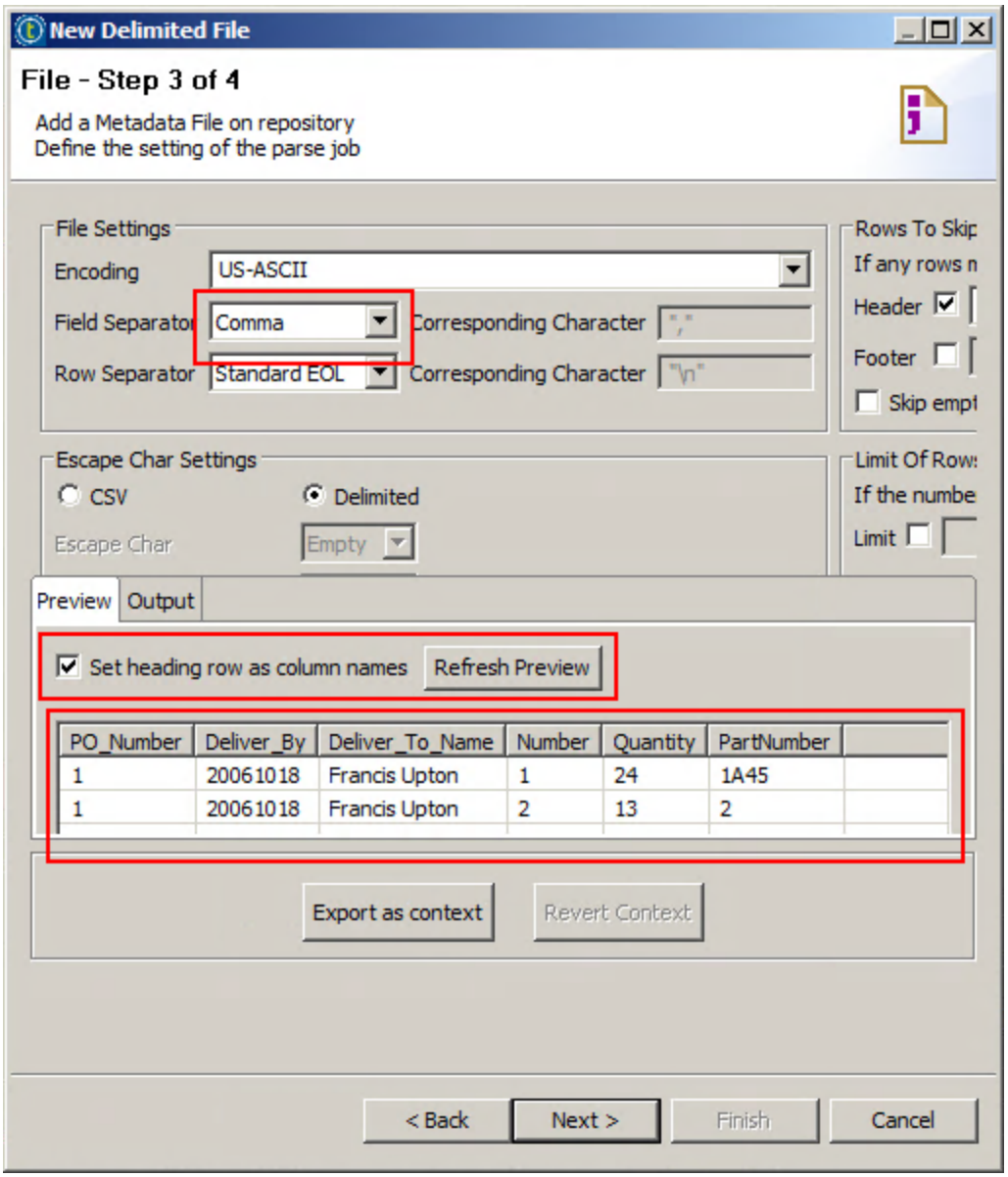
3. Enter the Name **Items** then click **Next**.



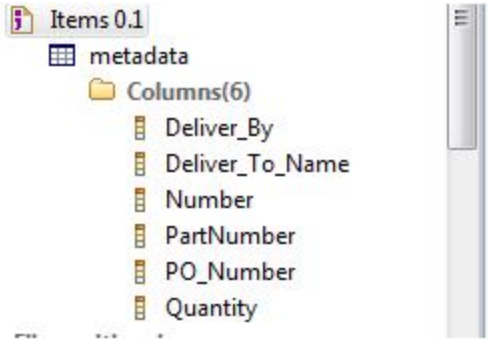
4. Click **Browse...**, change the extension filter to *.*, select *C:\StudentFiles\item.out* and click **Next**.



5. Set **Field Separator** to **Comma**. Check the **Set heading row as columns names** box then click **Refresh Preview** to get a live preview of the document.

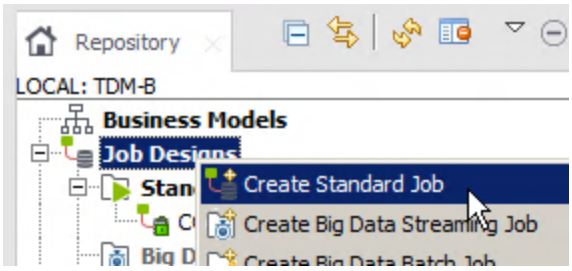


6. Click **Next** to explore and validate the schema. Note that some of the fields are imported as Integers and then click **Finish**. The imported file appears in the **Metadata > File delimited** branch of the **Repository**.

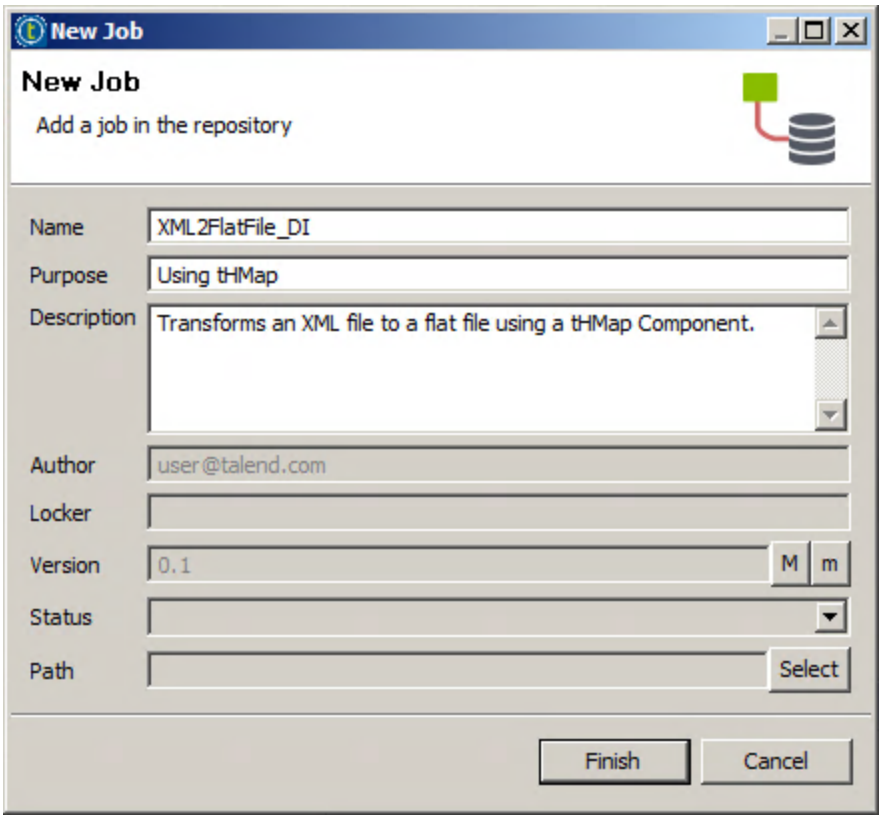


Creating the DI Job

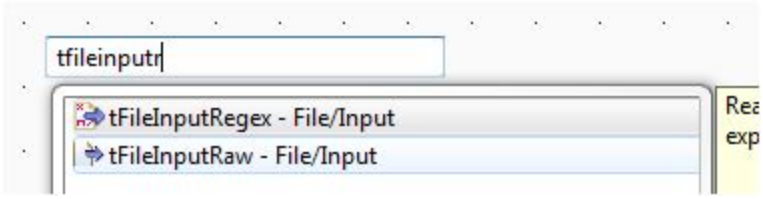
1. In the **Integration** perspective, right-click **Job Designs** and select **Create Standard Job** (or **Create Job** depending on the installed version):



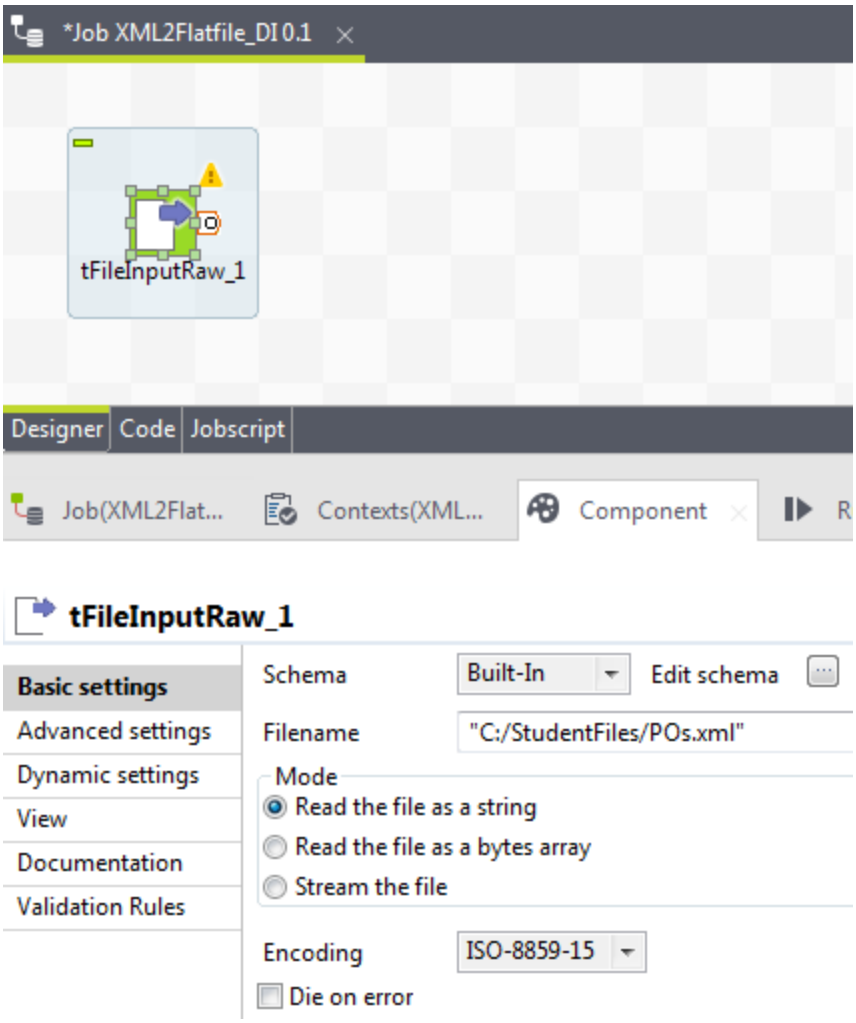
2. Configure the Job as illustrated below and click **Finish**.



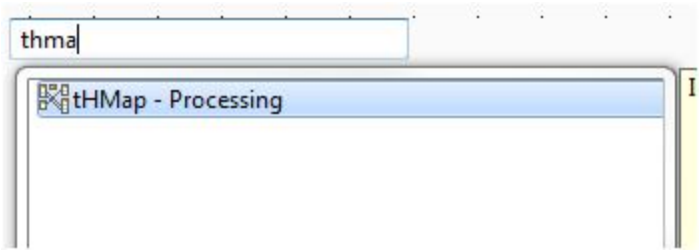
3. Place a **tFileInputRaw** component onto the canvas :



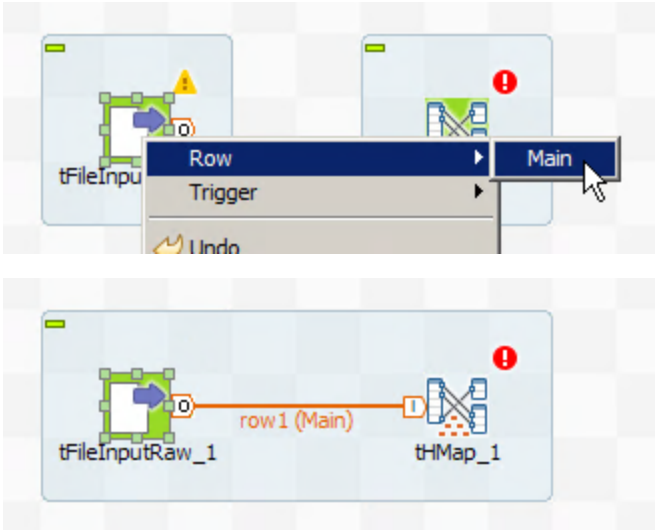
4. Double-click **tFileInputRaw** to display its **Component** view.
5. Click the **...** button next to the **Filename** field and select *C:\StudentFiles\POs.xml* :



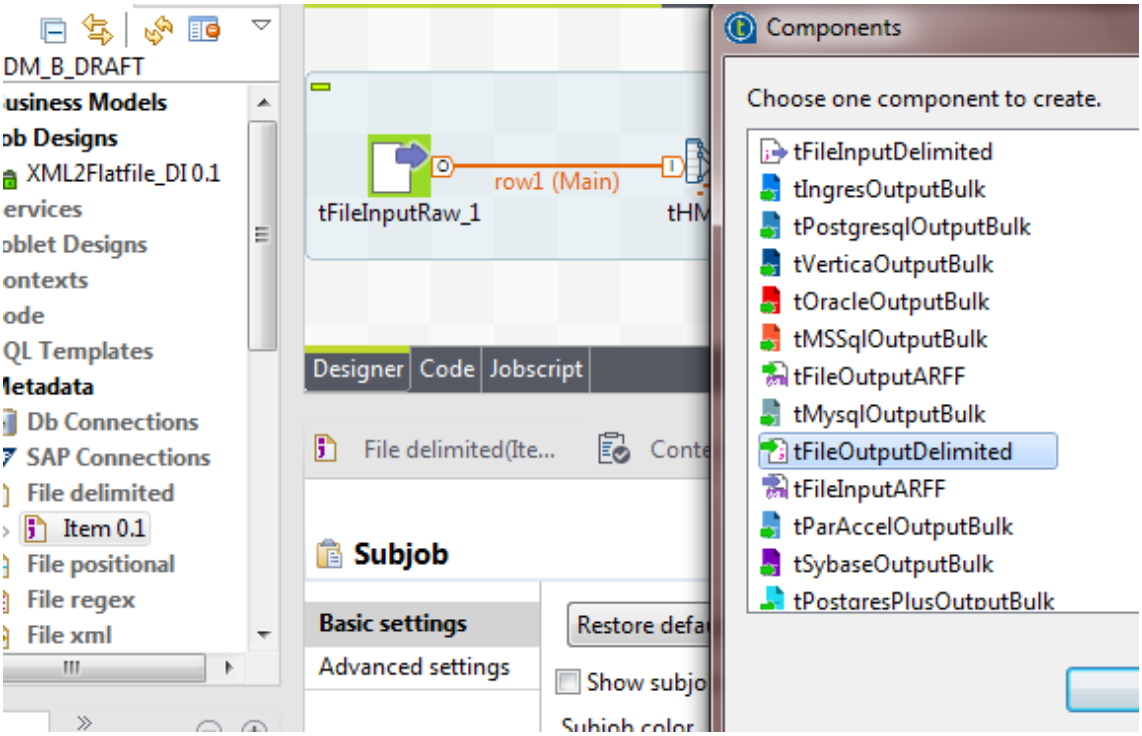
6. Place a **tHMap** component onto the canvas and acknowledge the red exclamation mark stating that the Map is not found. It has not been specified yet.



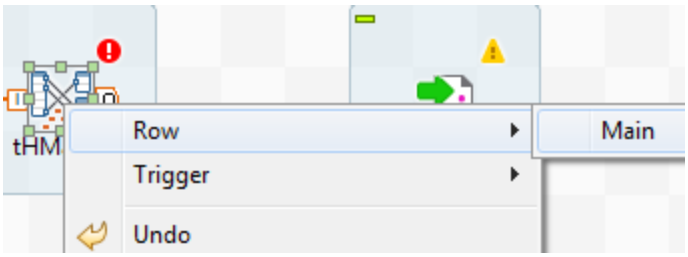
7. Right-click **tFileInputRaw** and select **Row > Main**. Drop the link on the **tHMap** Component:



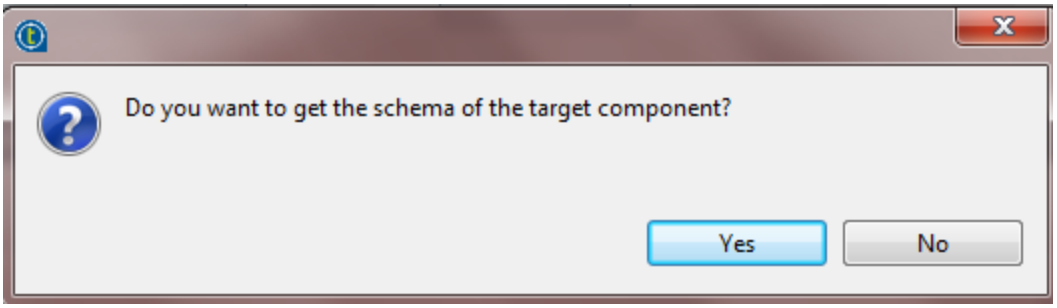
8. Drag and drop **Metadata > File delimited > Items 0.1** to the canvas. Select **tFileOutputDelimited** when asked which component to create, then click **OK**.



9. Right-click **tHMap** and select **Row > Main**. Drop the link on the **tFileOutputDelimited** Component.

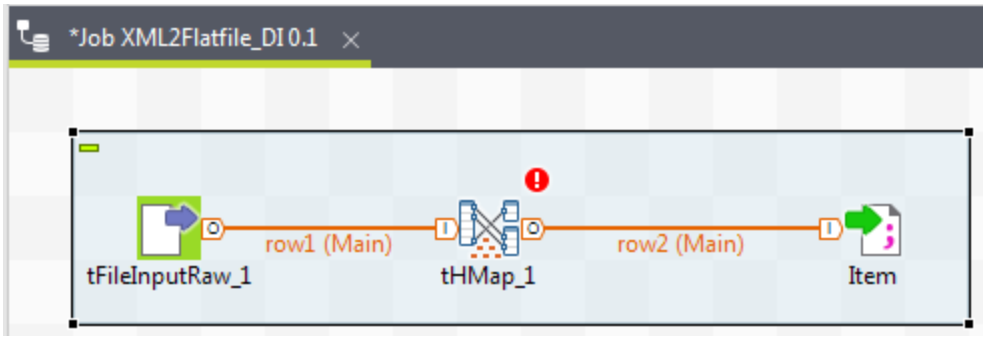


10. Click **Yes** when asked whether to propagate the schema from the **tHMap** Component to the **tFileOutputDelimited** Component.



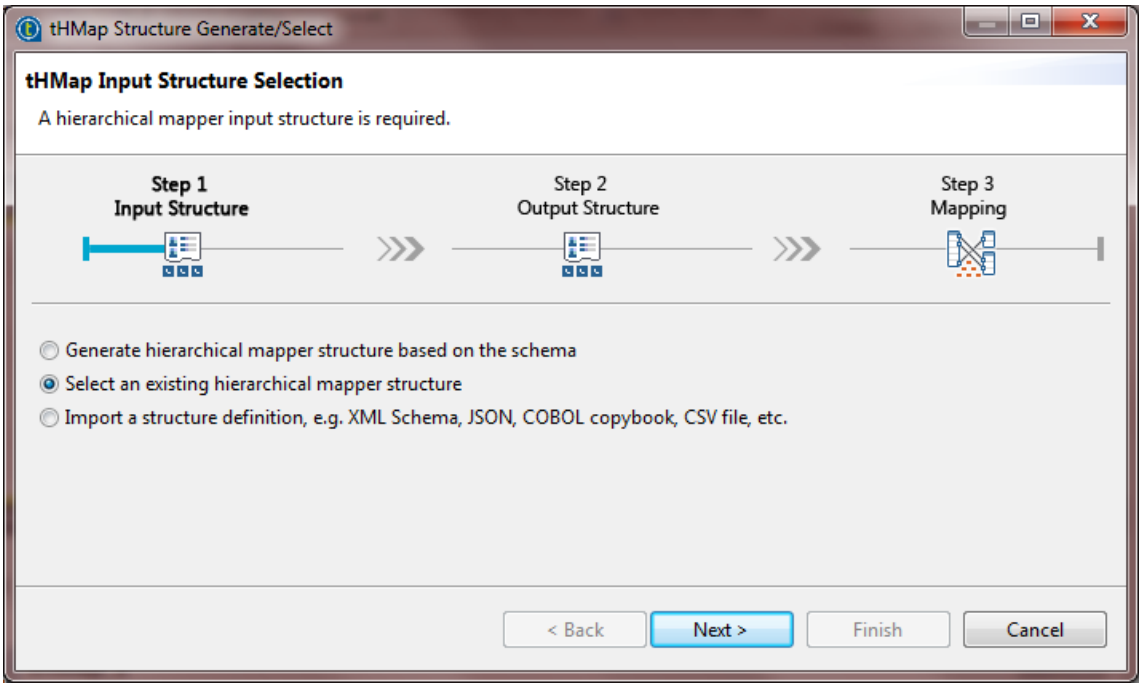
Using the tHMap Wizard

1. Double-click the **tHMap** Component to start the interactive wizard that will help you configuring the mapping.

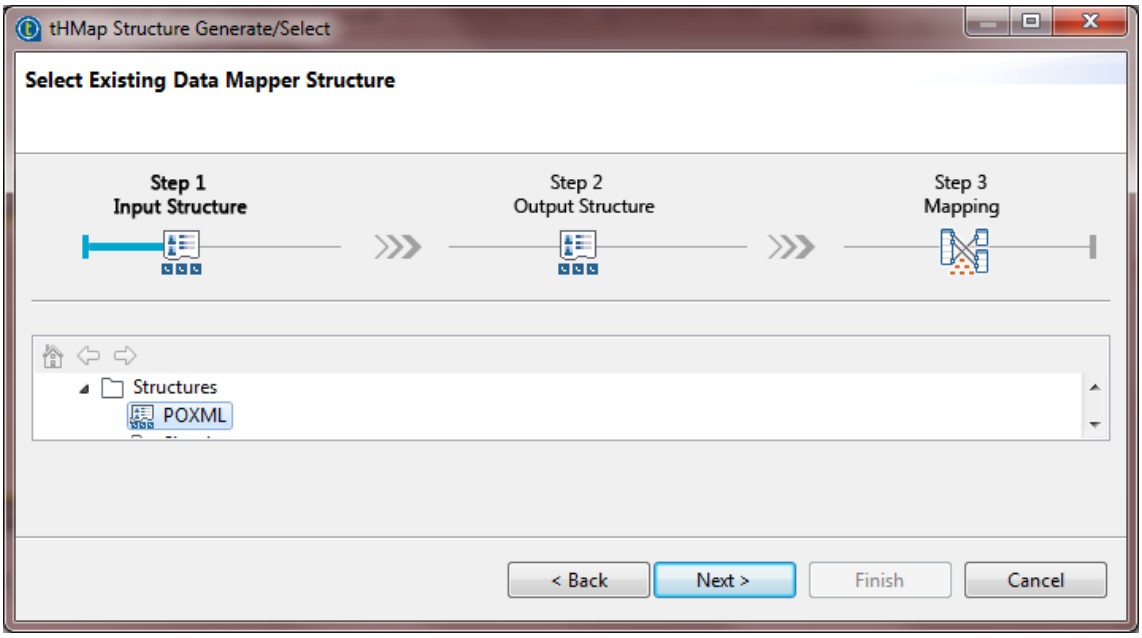


2. Step 1 of the Wizard is about selecting an input Structure.

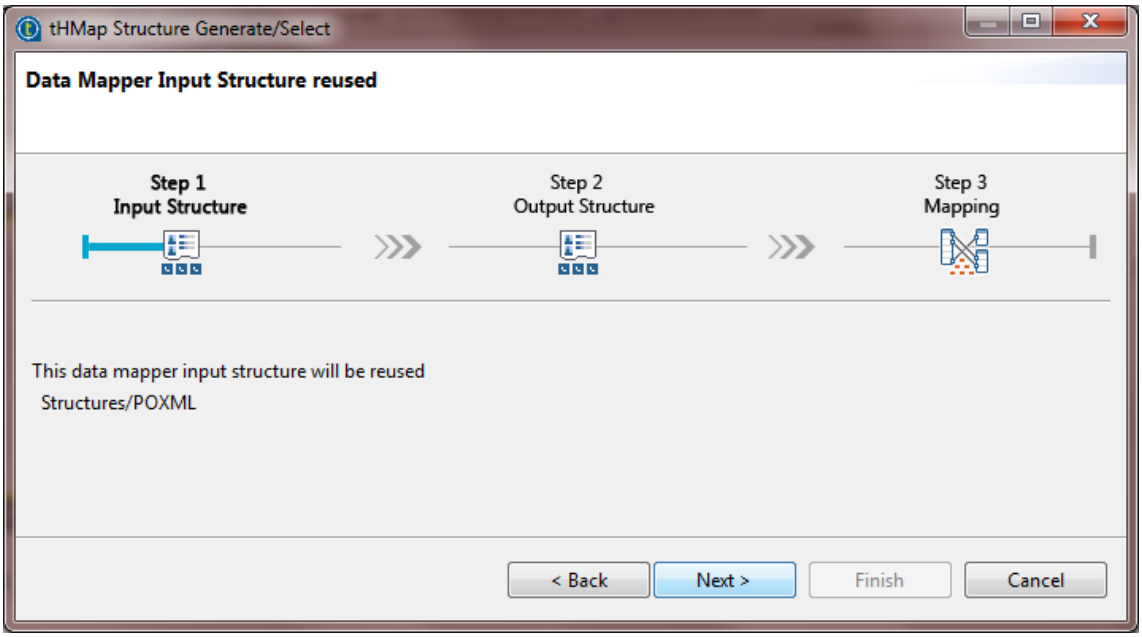
   Check the **Select an existing hierarchical mapper structure** option and click **Next**.



3. Select the *POXML* Structure in **TDM-Essentials > Structures** and click **Next >**.

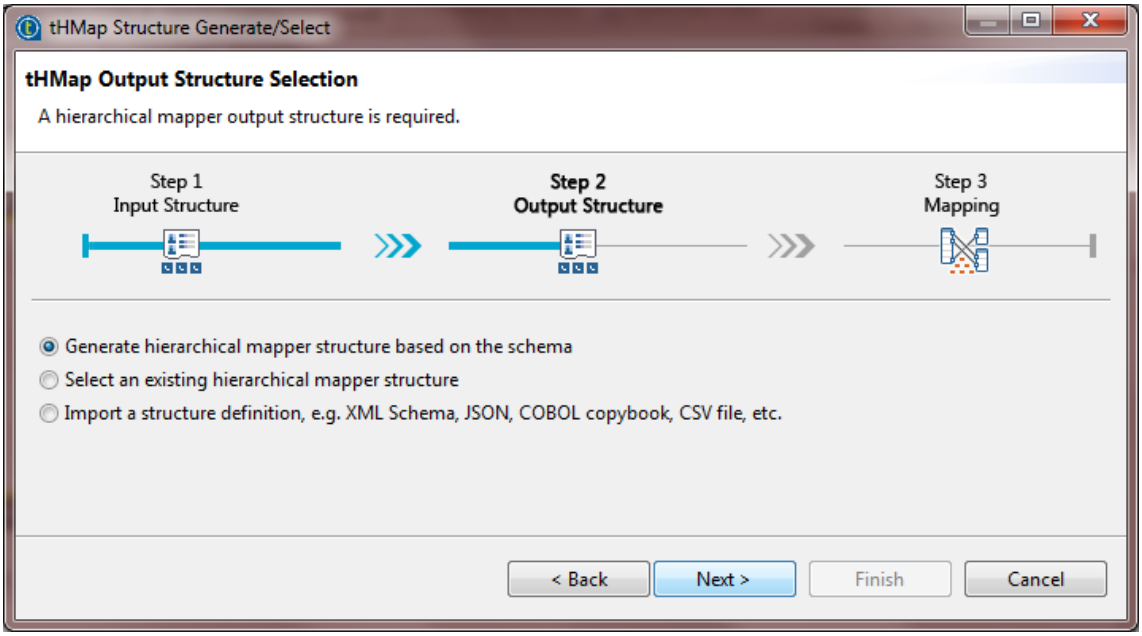


4. Click **Next** on the confirmation screen.

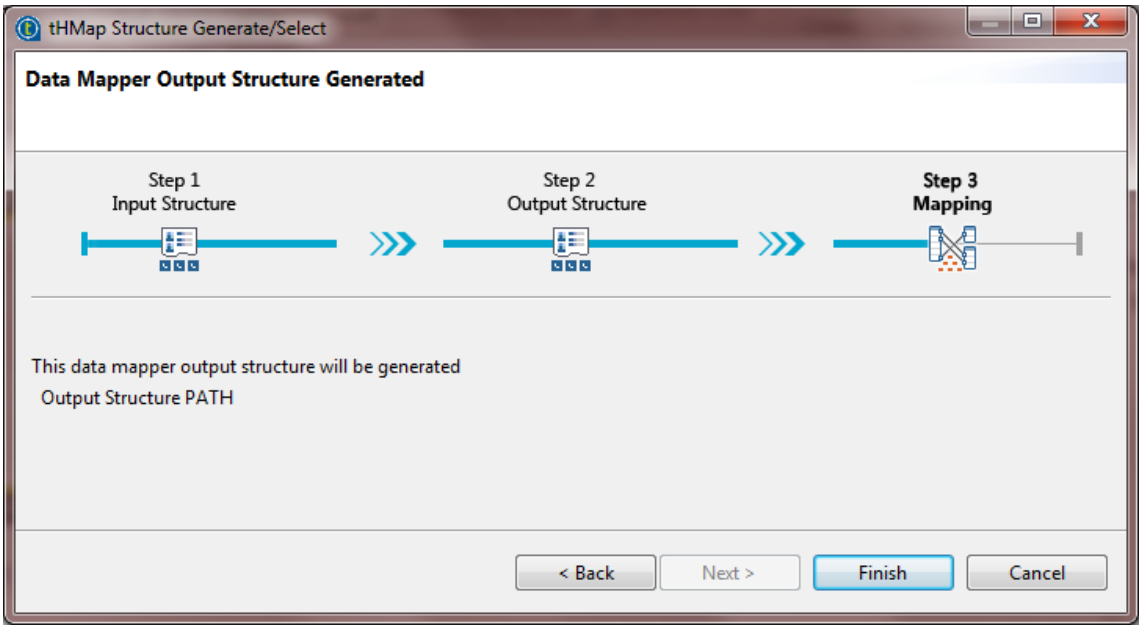


5. Step 2 of the wizard is about selecting an output Structure.

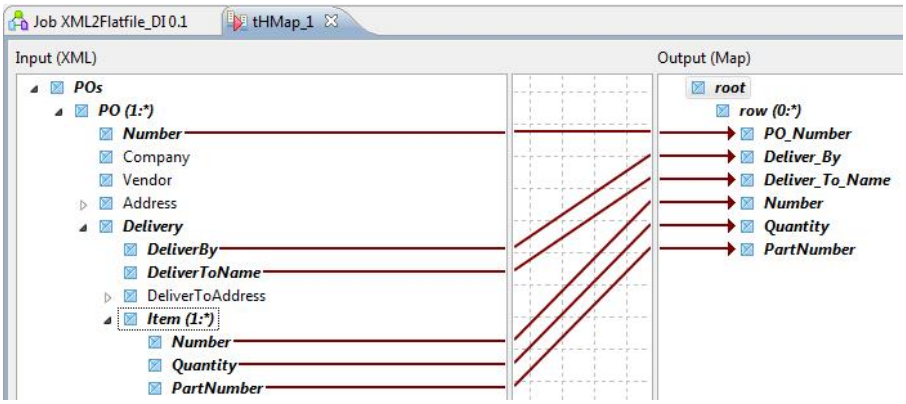
   Keep the default **Generate hierarchical mapper structure based on the schema** option and click **Next**.



6. Click **Next** on the confirmation screen.
7. Step 3 is a confirmation that the data mapper is going to be generated. Click **Finish**.



8. The Map is indeed created automatically. Map the elements as follows, accepting any Nested Loop creation in the process.

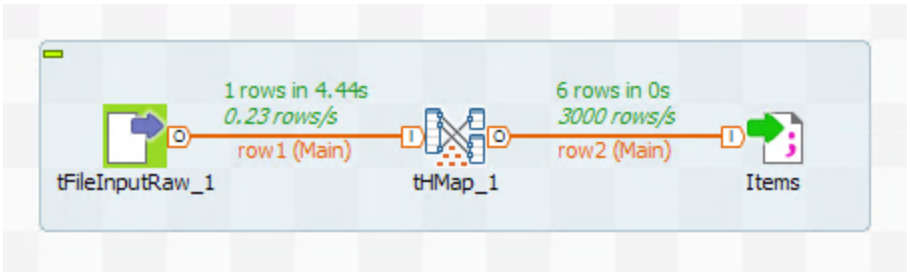


Note that you cannot perform a Test Run when the Structure has been created by a **tHMap Wizard**.

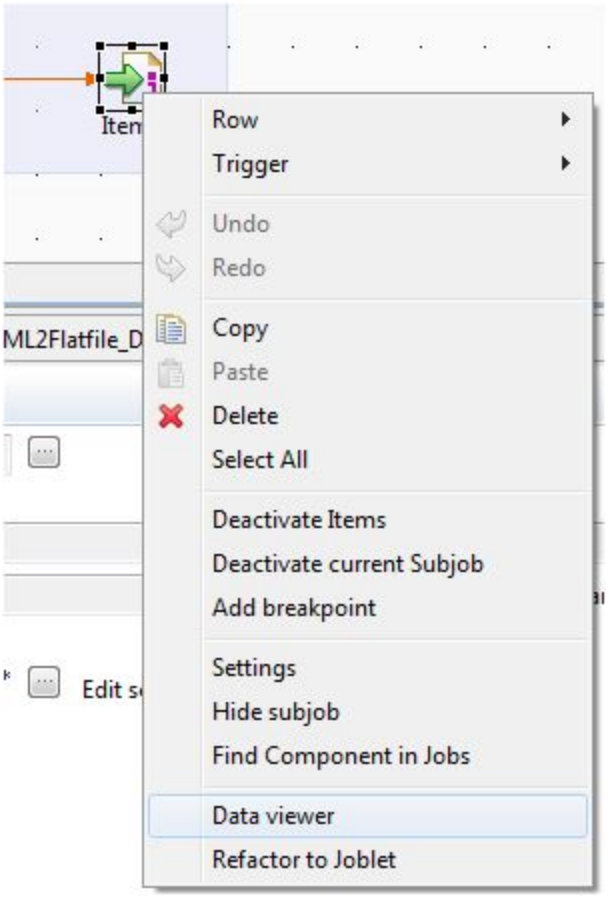
9. Save the **tHMap**.

Running the Job

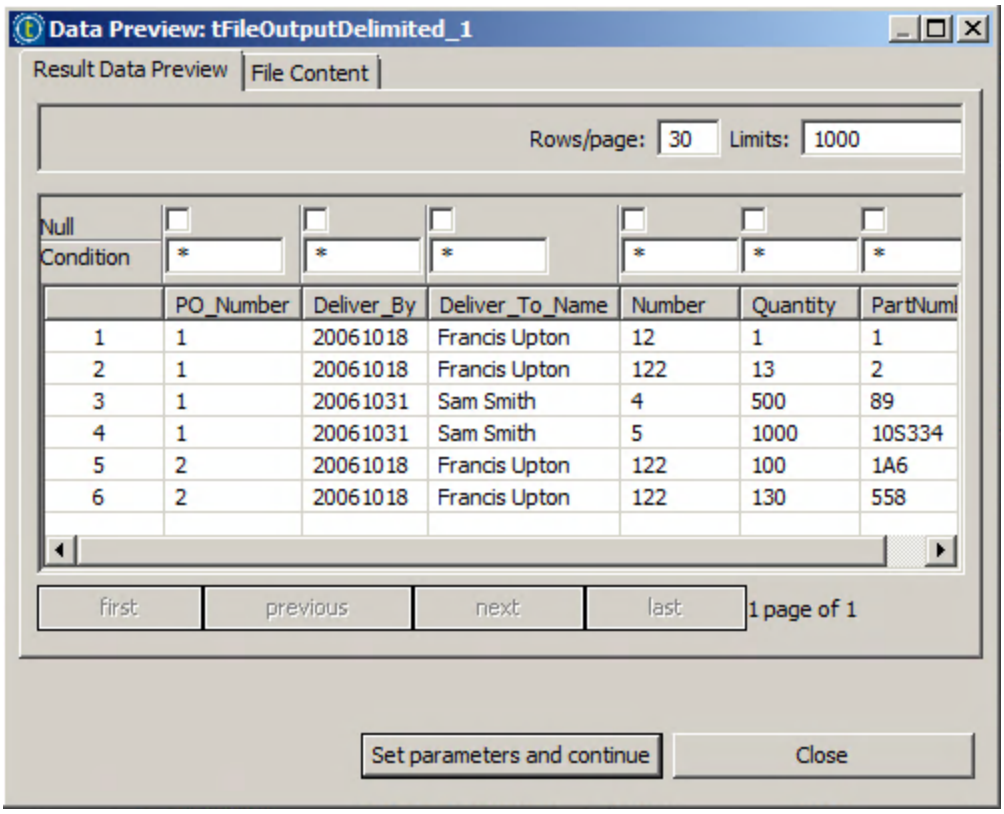
1. Switch back to the **Integration** perspective and run the job:

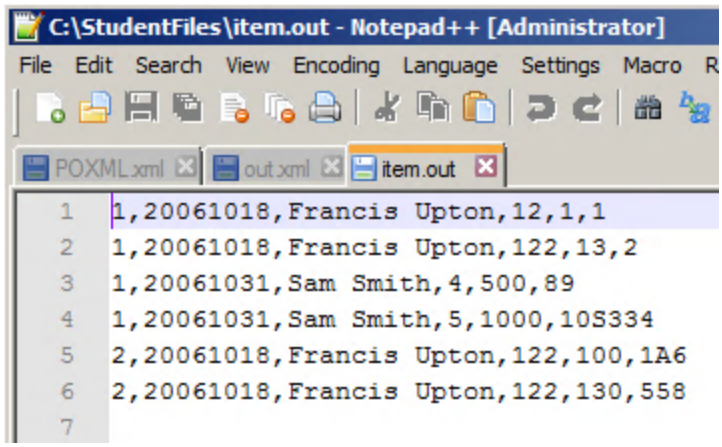


2. Right-click on the component **Items** and then click **Data Viewer**:



3. Use the **Data Viewer** to inspect the results. Alternatively, you can open *C:\StudentFiles\item.out* in Notepad++ and check the results are the same.





Next Step

This lesson is almost over. Head to the Wrap-Up section for a summary of the concepts reviewed in this lesson.

Wrap-Up

In this lesson, you learned how to:

- Use the tHMap Wizard to create a Map interactively, from Schemas and Structures already available or imported manually

Next Step

Congratulations, you successfully completed this lesson. Click the **Check your status with this unit** button below in order to save your progress. Then click **Completed. Let's continue >** on the next screen to jump to the next lesson.

LESSON 7

Using Context Variables

This chapter discusses the following.

Overview

Lesson Overview

In this exercise, you will use an existing Map that writes from a COBOL file to an XML file. This Map will be called from a Data Integration (DI) Job to demonstrate the integration between *Talend Data Mapper* and the **Integration** perspective of Talend Studio. Context variables will be used to pass values to the Job at runtime, so the output can take different values based on the context (a development or production environment, for example).

Objectives

After completing this lesson, you will be able to:

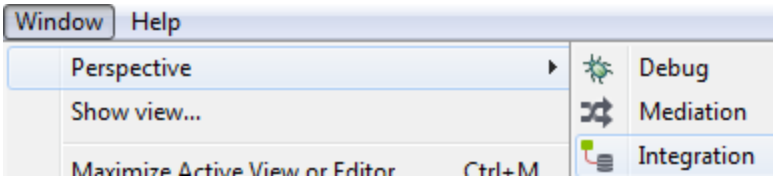
- Call a Talend Data Mapper Map from a DI Job using the **tHMap** Component
- Use **Contexts** and **Context Variables** to change the value of the output at runtime

Next Step

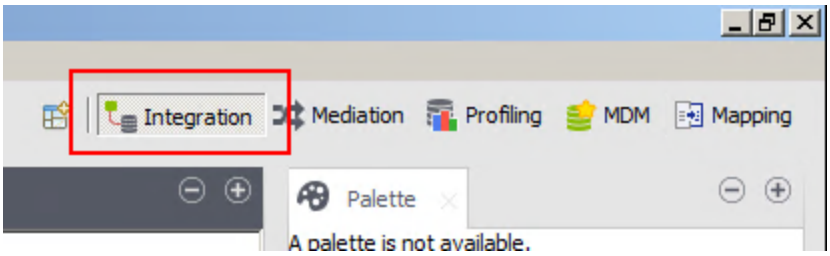
Let's create a DI Job in the Integration Perspective.

Creating the DI Job

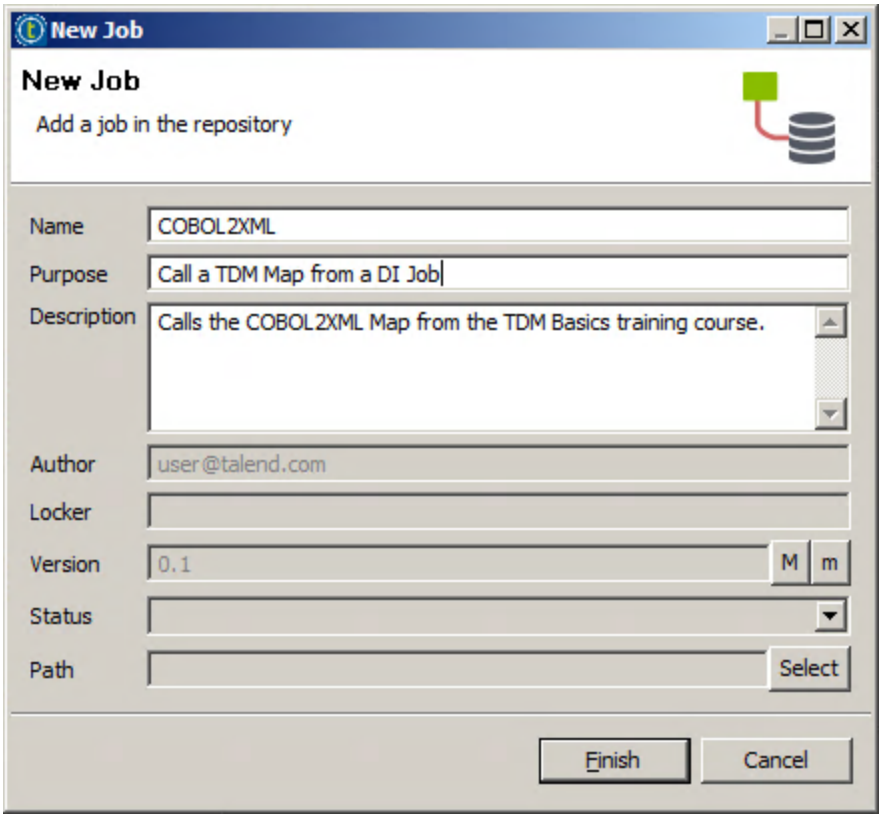
1. Click **Window > Perspective > Integration** to switch to the **Integration** perspective and be able to create a Data Integration Job.



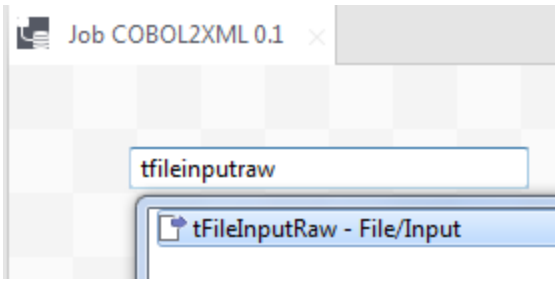
Note that you can also click the **Integration** button at the top right of the Studio to achieve the same result.



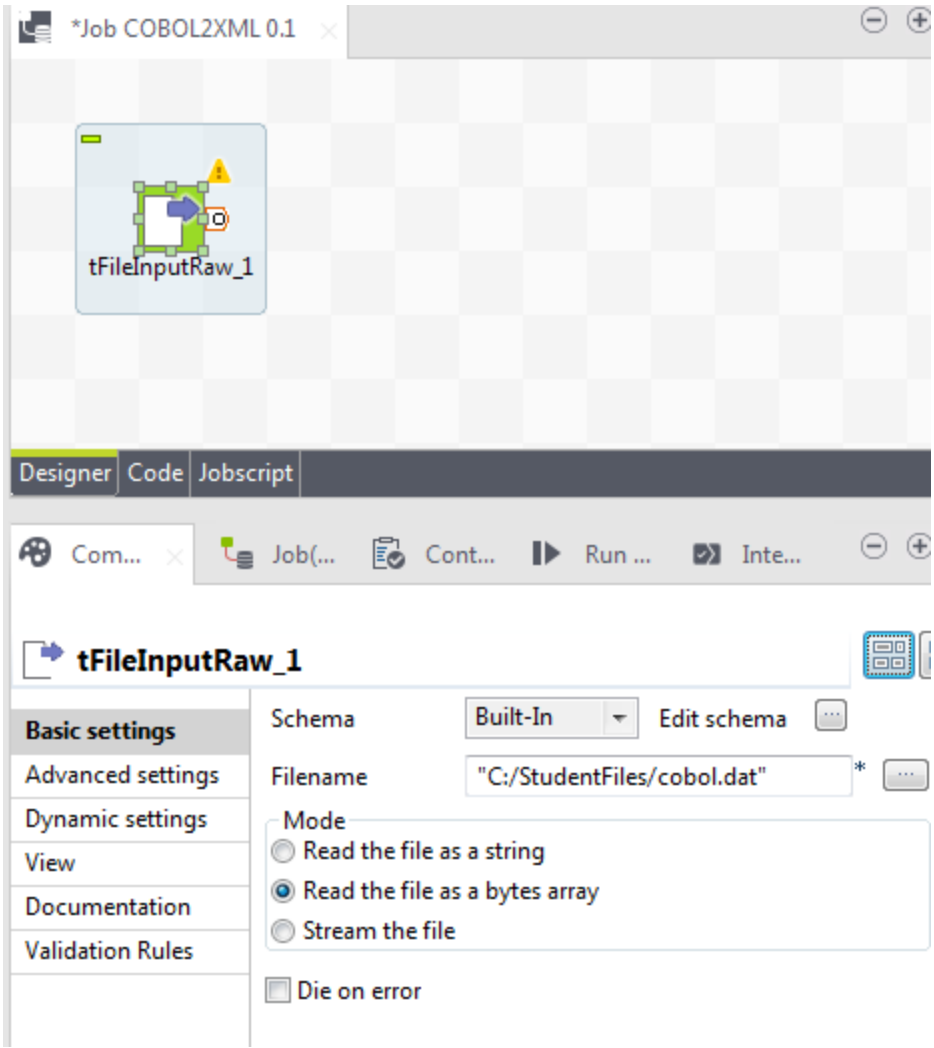
2. Right-click **Job Designs** in the **Repository** on the left, then select **Create Standard Job** (or **Create Job**, depending on the version of the Studio). Enter *COBOL2XML* in the **Name** field, fill in the **Purpose** and **Description** fields as illustrated below and click **Finish**.



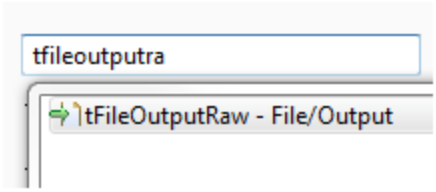
3. Place a **tFileInputRaw** component on the canvas:



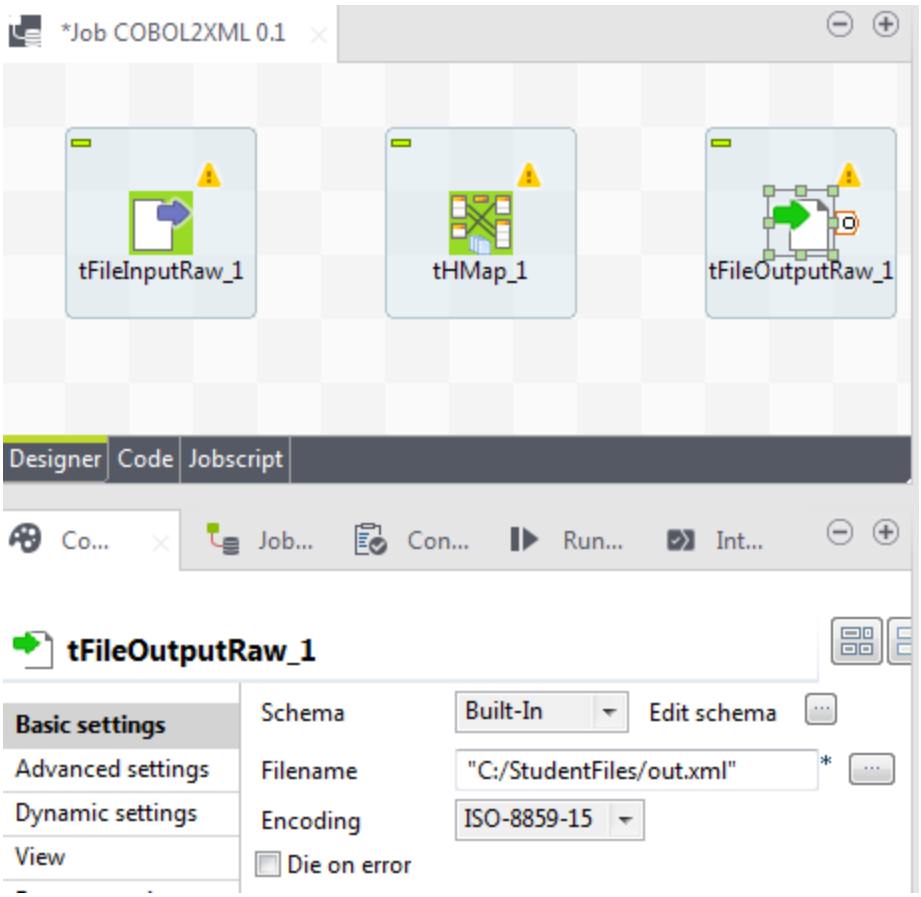
4. Double-click the **tFileInputRaw** component to display its **Component** view.
5. Click the **...** button next to the **Filename** field. Select the *C:\StudentFiles\cobol.dat* file and set the **Mode** field to *Read the file as a bytes array*.



6. Place a **tFileOutputRaw** component on the canvas:



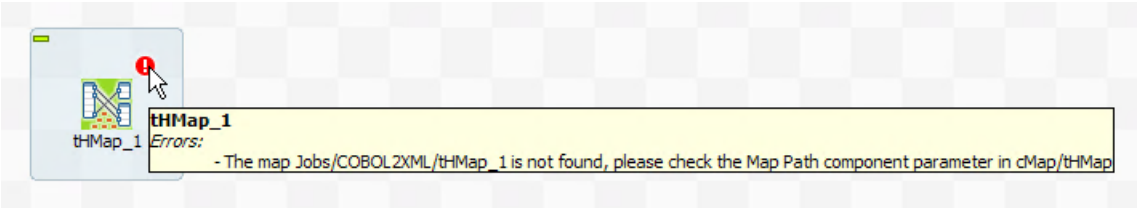
7. Double-click the **tFileOutputRaw** component to display its **Component** view.
8. Click the **...** button next to the **Filename** field. Browse to *C:\StudentFiles* and enter *out.xml* in the **File name:** field:



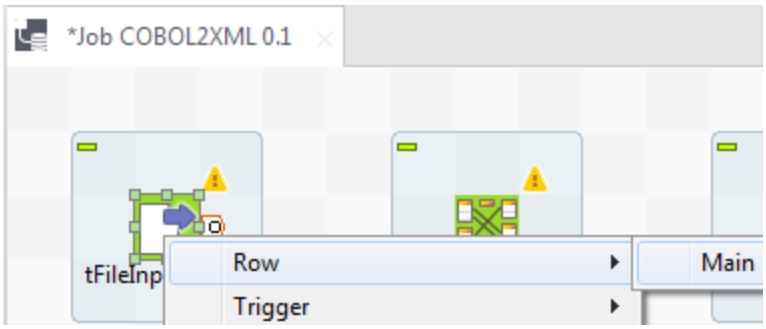
9. Finally, place a **tHMap** component on the canvas. This component is responsible for calling a *Talend Data Mapper* Map.

   Note that it needs to be connected to an input and an output before being configured.

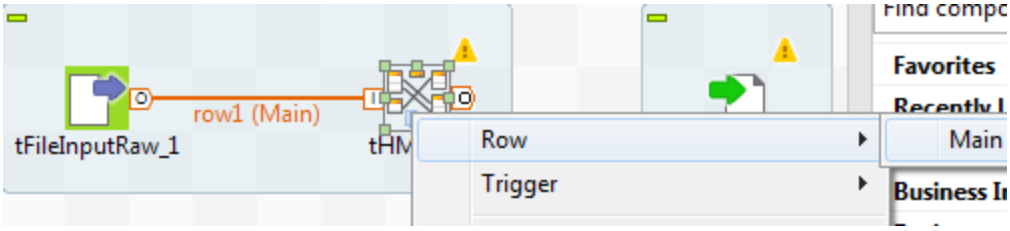
   Also note the red exclamation point indicating the Map is not found (yet). The actual Map to use will be specified later on.



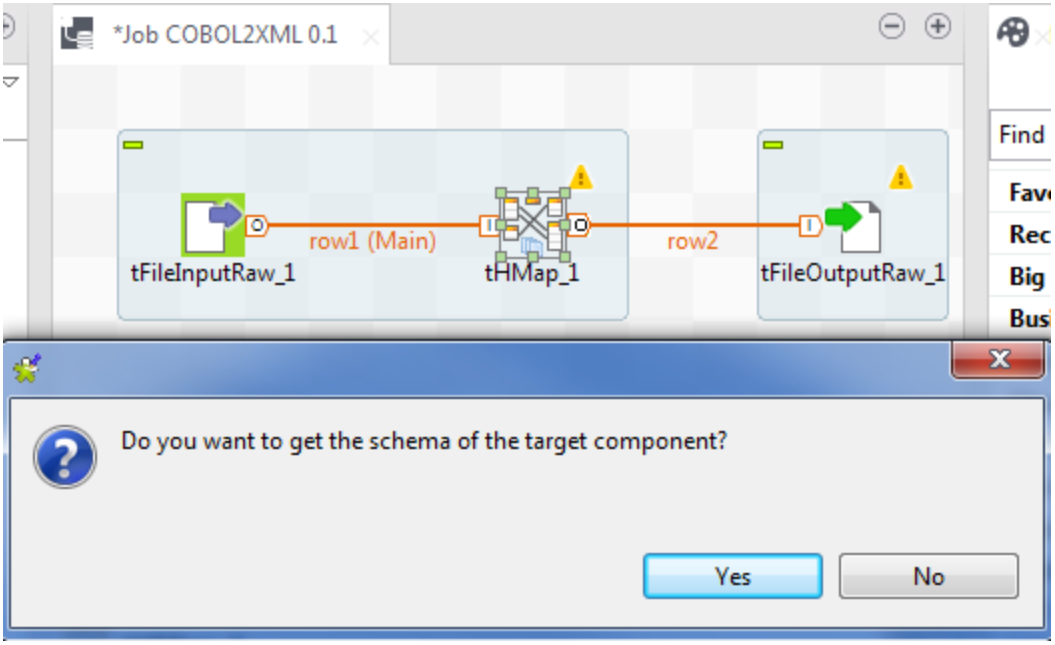
10. Right-click **tFileInputRaw**, select **Row > Main** and drop the link on the **tHMap** component:



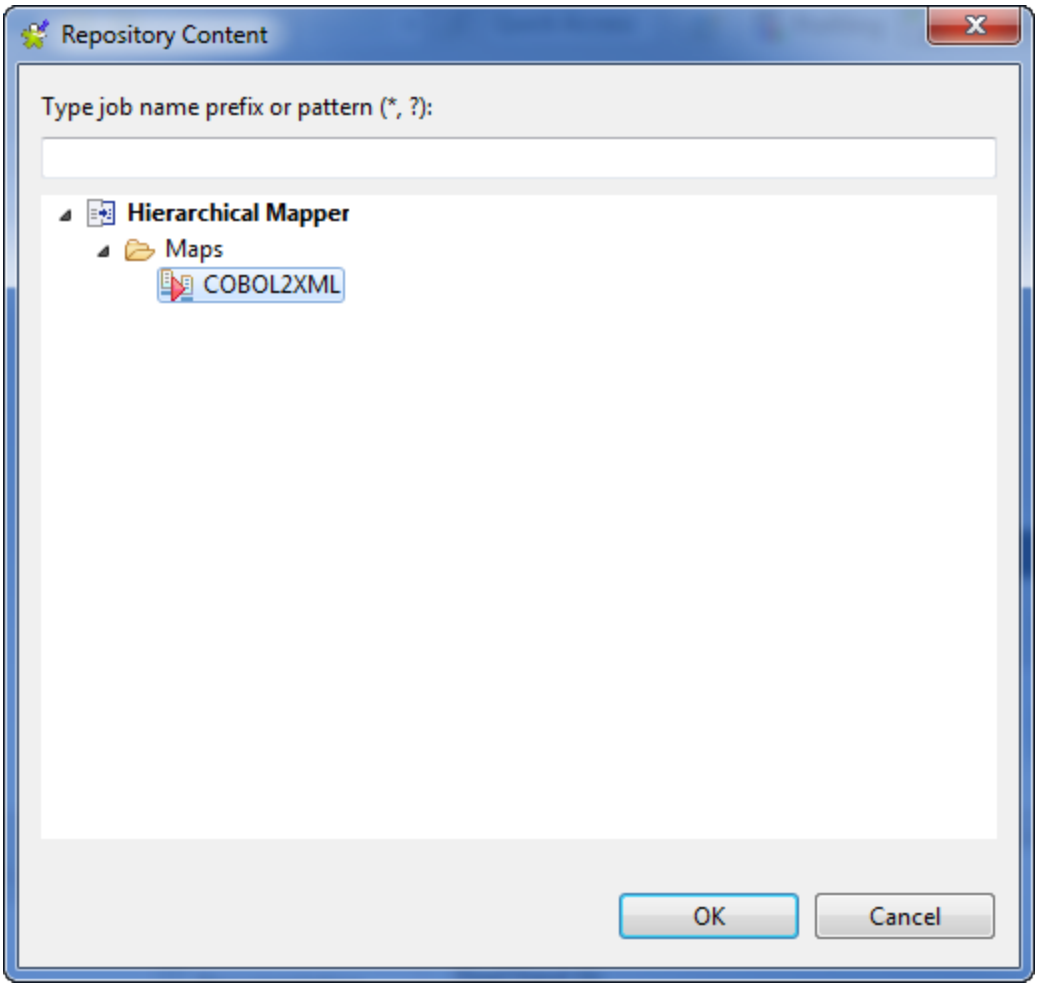
11. Right-click **tHMap**, click **Row > Main** and drop the link on the **tFileOutputRaw** component:



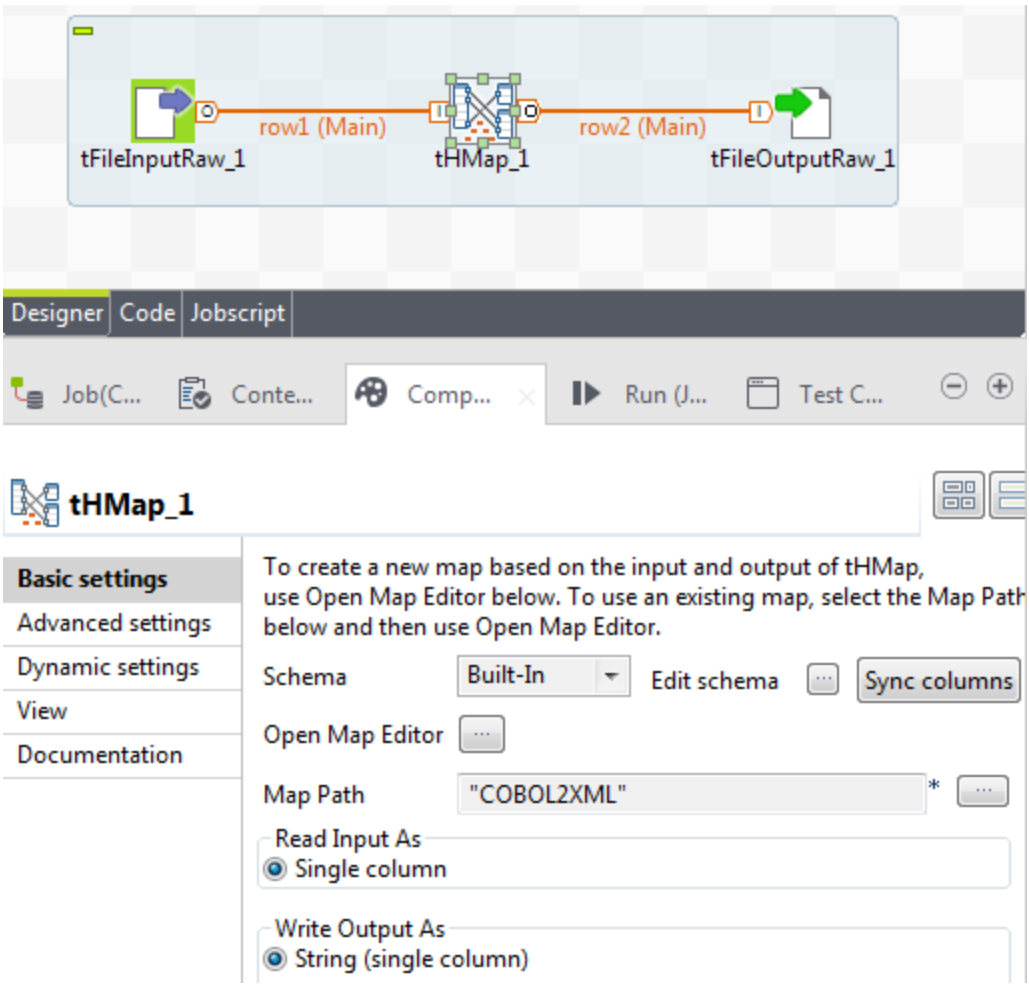
12. Click **Yes** when asked to propagate the schema from the **tHMap** Component to the **tFileOutputRaw** Component.



13. In the **tHMap Component** view, click the **…** button to the right of **Map Path**.
14. From the **Repository Content** pop-up window, select **Maps/COBOL2XML** and click **OK**:



15. Open the Component View of **tHMap** to check that **Read Input As** is set to *Single column*, and **Write Output As** is set to *String (single column)*.

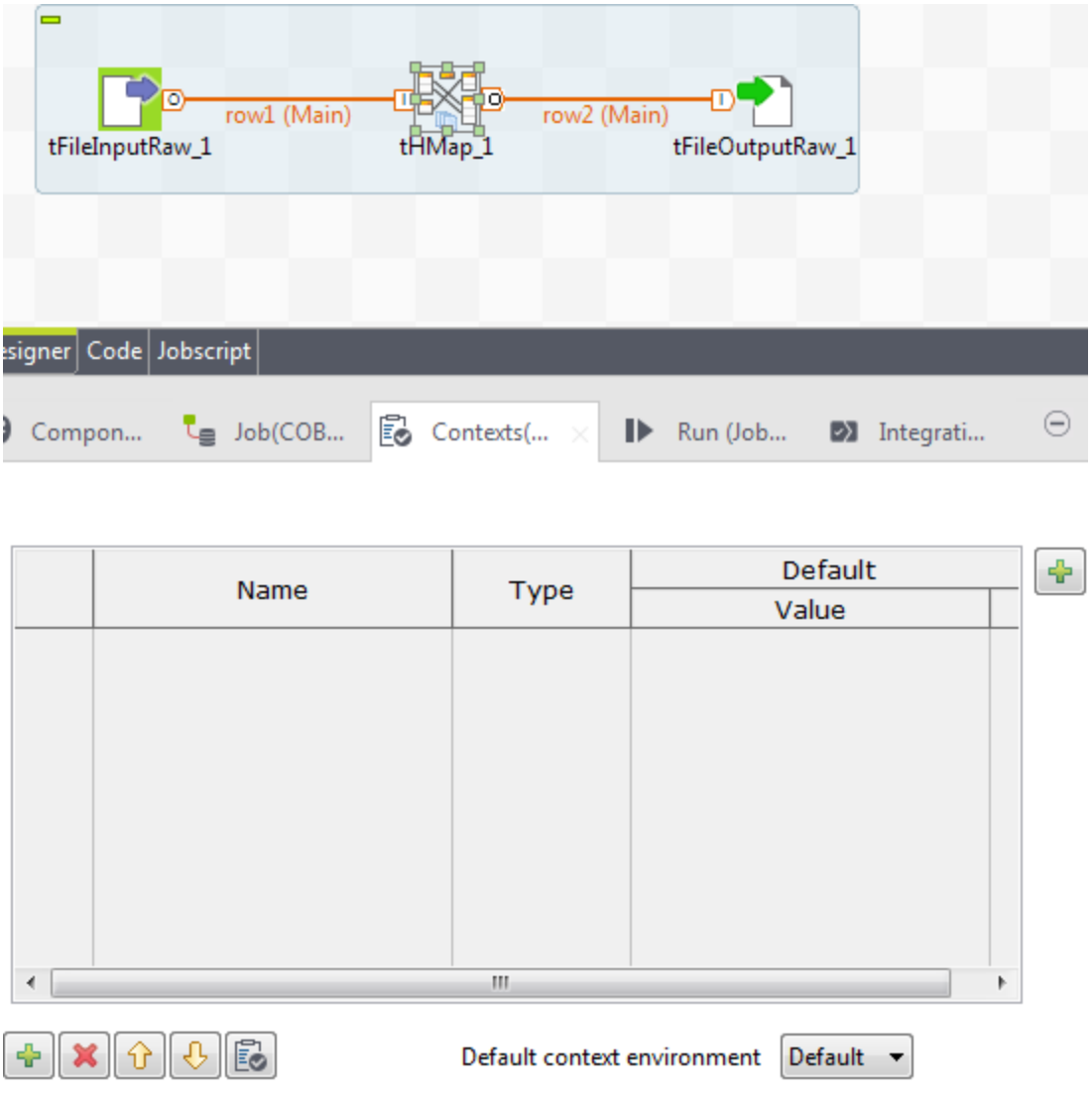


Next Step

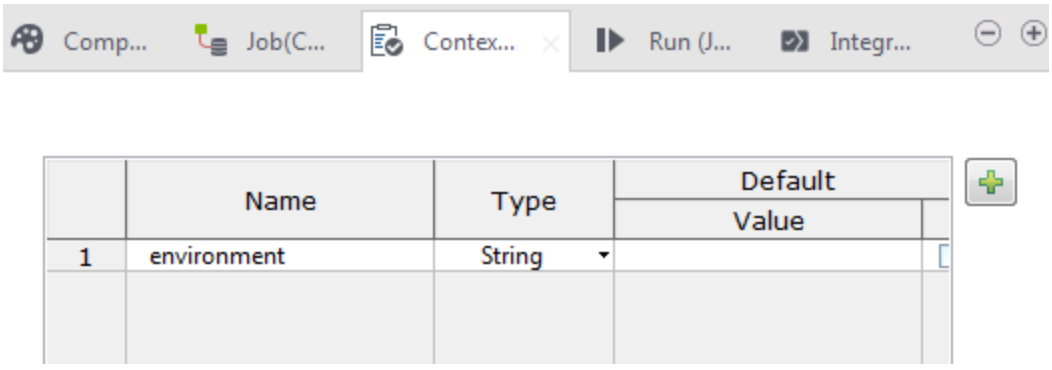
The DI Job is almost configured. Let's add some context variables.

Using Context Variables

1. Click the **Contexts(Job COBO2XML)** view, and check that there are no **Variables** yet:



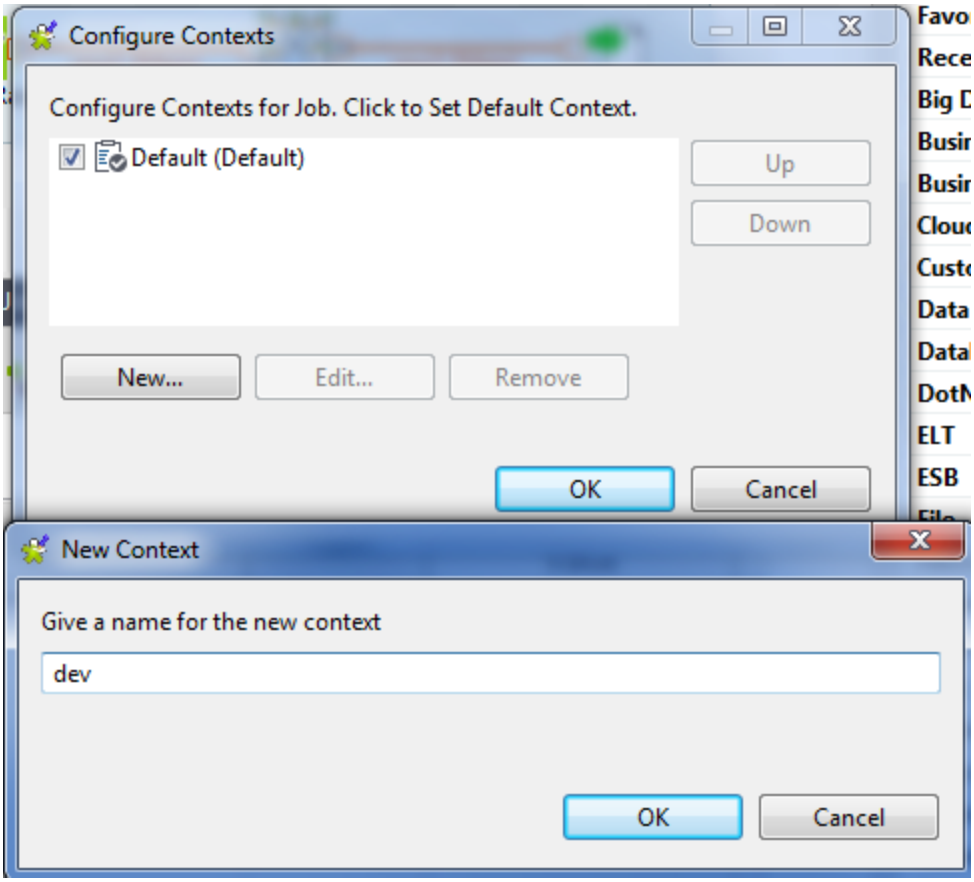
2. Click the green **+** icon at the **bottom left** of the table to add a Context Variable. Set the **Name** column to **environment**.



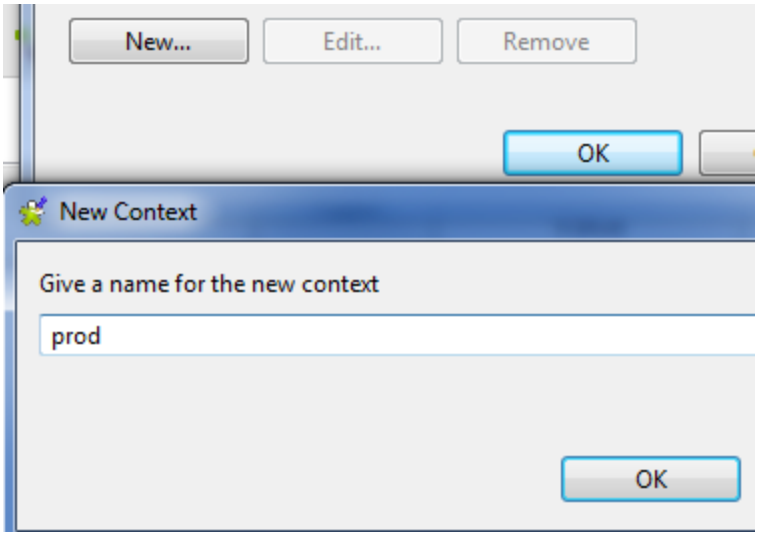
3. Click the green **+** icon at the **top right** of the table to configure the available Contexts for the current job. By default, there is

only one *Default* Context, so every Context Variable can have only one value. Adding additional Contexts will allow the same Variable to have multiple values. The runtime Context selected in the **Run Job** view will determine which value should be used. For example, let's add a *Development* Context and a *Production* Context.

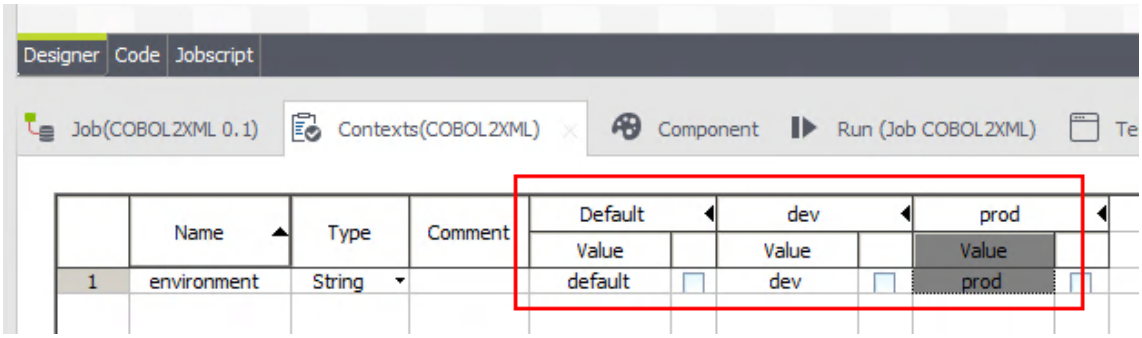
Click **New** to create a new Context (i.e. a group of Context Variables), enter the name **dev** then click **OK**.



4. Click **New** again to add a second Context. Name it **prod** then click **OK** and **OK** again to close the **Configure Contexts** window.



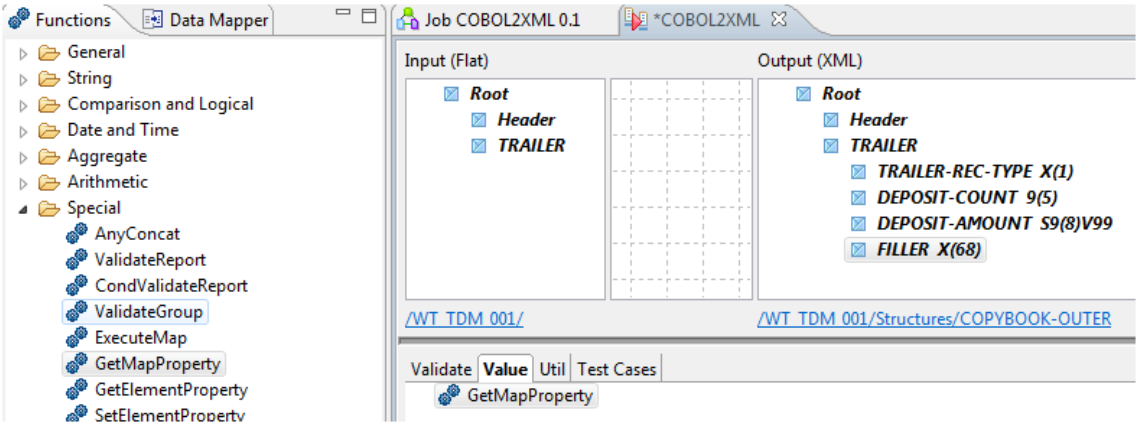
5. In the **Contexts** view, go to the column **Value** and enter **default** for **Default**, **dev** for **dev**, and **prod** for **prod**:



This way, the environment Context Variable can be used to determine the current development stage. It can be used to generate file names and paths that depend on the Context.

Mapping Perspective

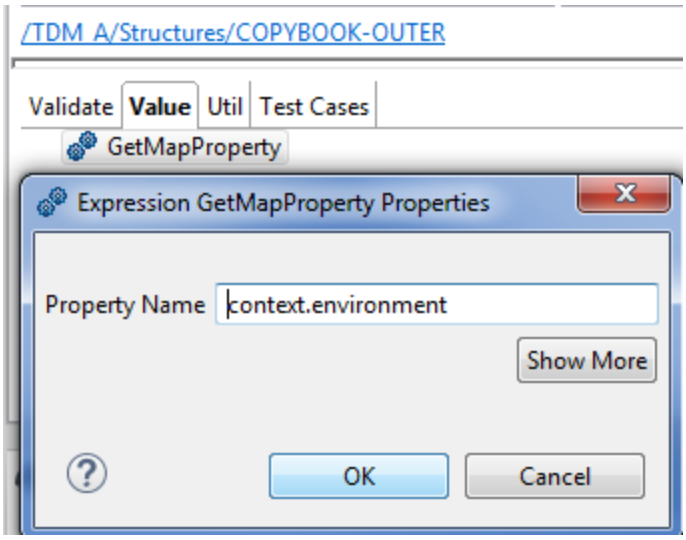
1. Switch back to the **Mapping** perspective, open the **COBOL2XML** Map and select the **TRAILER > FILLER** element in the Output area.
2. Remove the existing */Root/TRAILER/FILLER* value from the **Value** tab.
3. Drag a **Functions > Special > GetMapProperty** Function to the **Value** tab.



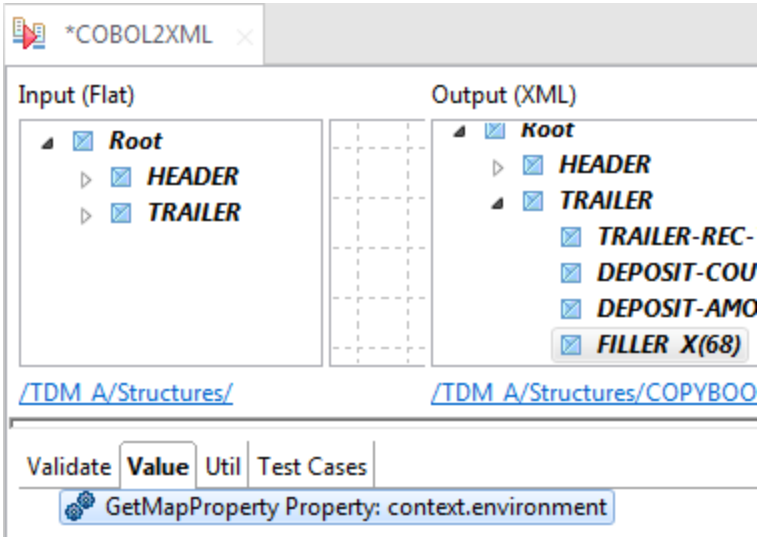
4. Double-click the **getMapProperty** Function and enter *context.environment* in the **Property Name** field.

   This means the *FILLER* field will contain the value of the *environment* Context Variable.

   Finally, click **OK** to save the changes.



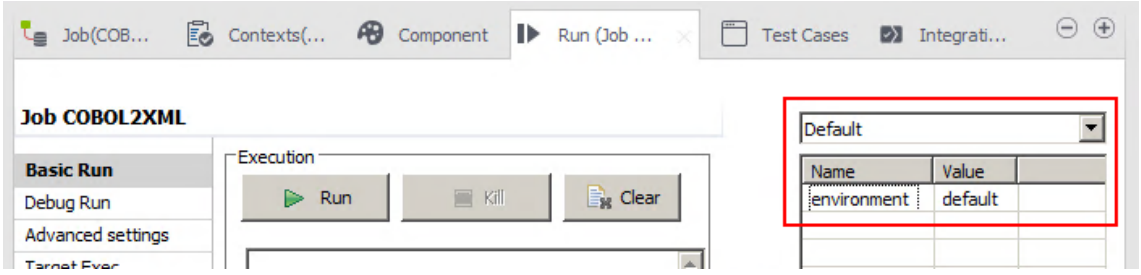
5. Check that the Value tab looks like the following.



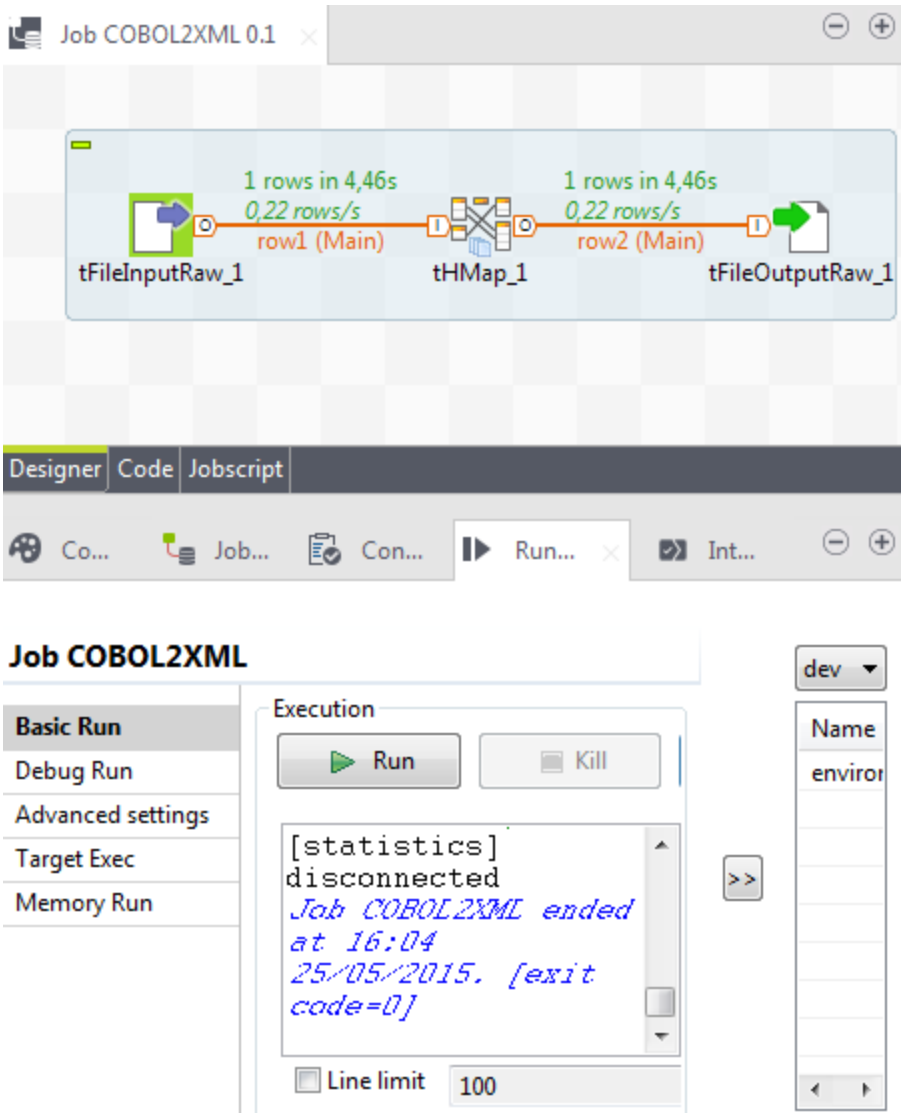
6. Save the changes to the *COBOL2XML* Map by hitting **Ctrl+S** or clicking **File > Save**.

Integration Perspective

1. Switch back to the Integration perspective, open the **COBOL2XML** Job and go to the **Run (Job COBOL2XML)** view.
2. Select one of the available Contexts from the drop-down menu on the right (**Default**, **dev**, or **prod**) before running the Job.



3. Run the Job by clicking the **Run** button, and wait for the Job to complete. A successful run will display the following message.



4. Open *C:\StudentFiles\out.xml* (in Internet Explorer or Notepad++, for example) and check that the value of the **FILLER** element matches the Context you selected before running the Job (here, *dev*)

```
<?xml version="1.0"?>
- <Root>
  - <HEADER>
      <HEADER-REC-TYPE>H</HEADER-REC-TYPE>
      <VENDOR-ID>TEST-VENDOR-TALEND</VENDOR-ID>
      <FILE-ID>169FFE8A-7FCD-404B-AAC5-21B9189EDA35</FILE-ID>
      <DATE-IN/>
    - <DETAIL>
      - <Record>
          <DETAIL-REC-TYPE>D</DETAIL-REC-TYPE>
          <DETAIL-ID>REM01</DETAIL-ID>
          <DETAIL-NAME>Baltimore</DETAIL-NAME>
          <FILLER/>
        </Record>
      - <Record>
          <DETAIL-REC-TYPE>D</DETAIL-REC-TYPE>
          <DETAIL-ID>REM02</DETAIL-ID>
          <DETAIL-NAME>Buffalo</DETAIL-NAME>
          <FILLER/>
        </Record>
      </DETAIL>
  </HEADER>
  - <TRAILER>
      <TRAILER-REC-TYPE>T</TRAILER-REC-TYPE>
      <DEPOSIT-COUNT>2</DEPOSIT-COUNT>
      <DEPOSIT-AMOUNT>400</DEPOSIT-AMOUNT>
      <FILLER>dev</FILLER>
  </TRAILER>
</Root>
```

Next Step

This lesson is almost over. Head to the Wrap-Up section for a summary of the concepts reviewed in this lesson.

Wrap-Up

In this lesson, you learned how to:

- Call a Talend Data Mapper Map from a DI Job using the tHMap Component
- Use Contexts and Context Variables to change the value of the output at runtime

Next Step

Congratulations, you successfully completed this lesson. Click the **Check your status with this unit** button below in order to save your progress. Then click **Completed. Let's continue >** on the next screen to jump to the next lesson.

This page intentionally left blank to ensure new chapters start on right (odd number) pages.

LESSON 8

Using the cMap Component

This chapter discusses the following.

Overview

Lesson Overview

In this exercise, you will configure a Route to perform an XML to JSON transformation using a **cMap** Component. This Component is only available in the **Mediation** Perspective. The Route will listen for changes in an XML file containing a list of accounts, and will produce a JSON file containing a list of contacts for each account.

Objectives

After completing this lesson, you will be able to:

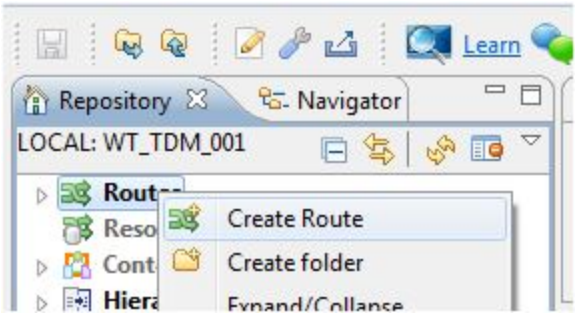
- Create a JSON Structure by adding and configuring nodes manually. The same method can be applied to XML, COBOL...
- Use the **cMap** Component in Routes in the **Mediation** Perspective. It demonstrates the integration between the various Talend products. A Map can be reused in other perspectives/products.

Next Step

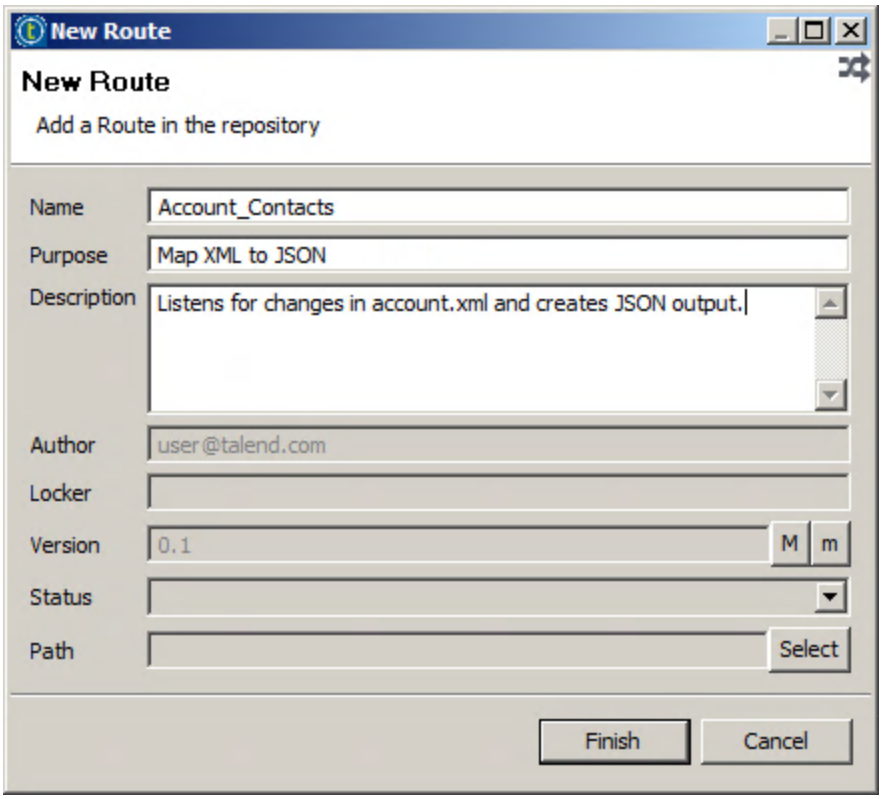
Switch to the **Mediation** Perspective and create a Route.

Creating a Route

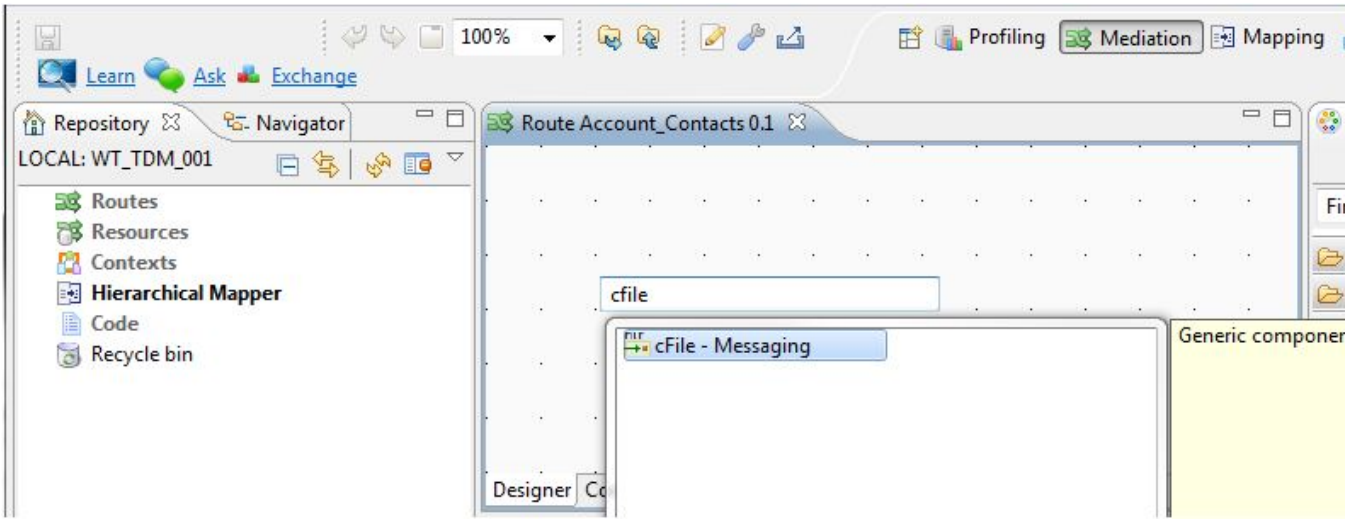
1. Switch to the **Mediation** perspective. Create a new Route by right-clicking **Routes** and selecting **Create Route**.



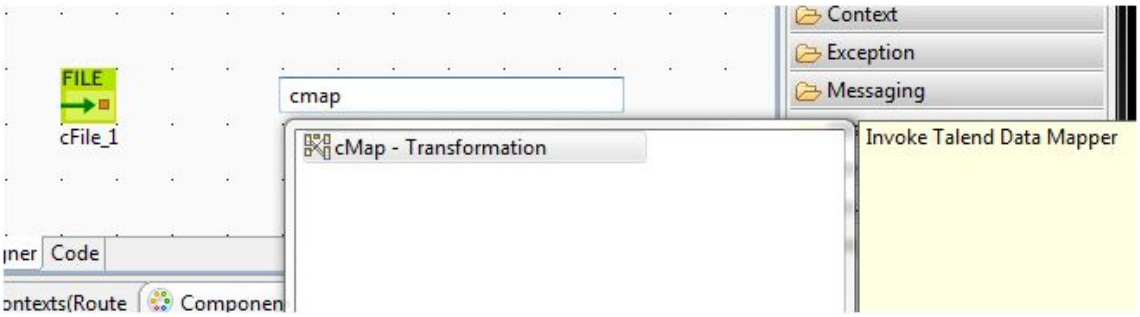
2. Configure the new Route as illustrated below then click **Finish**.



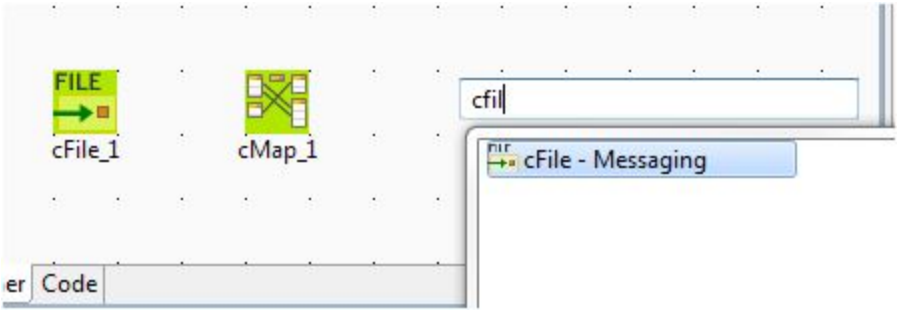
3. Place a **cFile** Component on the canvas:



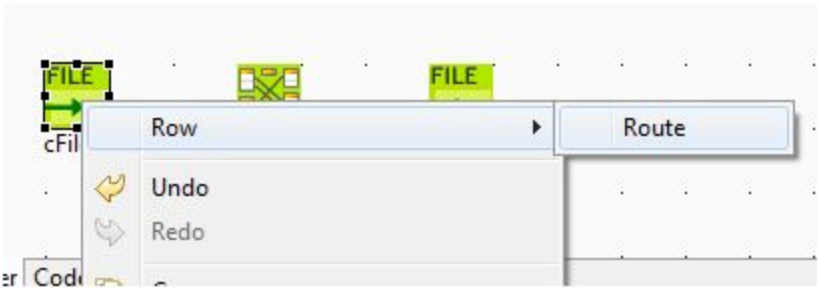
4. Place a **cMap** Component to the right of the **cFile**.



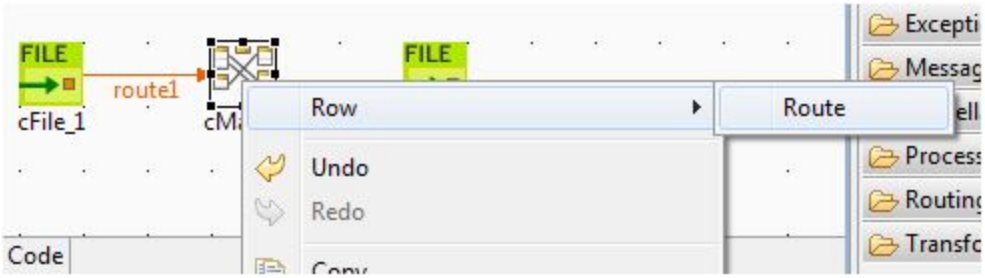
5. Copy *cFile_1* and paste it to the right of the **cMap** Component to create *cFile_2*.



6. Right-click *cFile_1*, select **Row > Route** and drop the route on top of *cMap*.

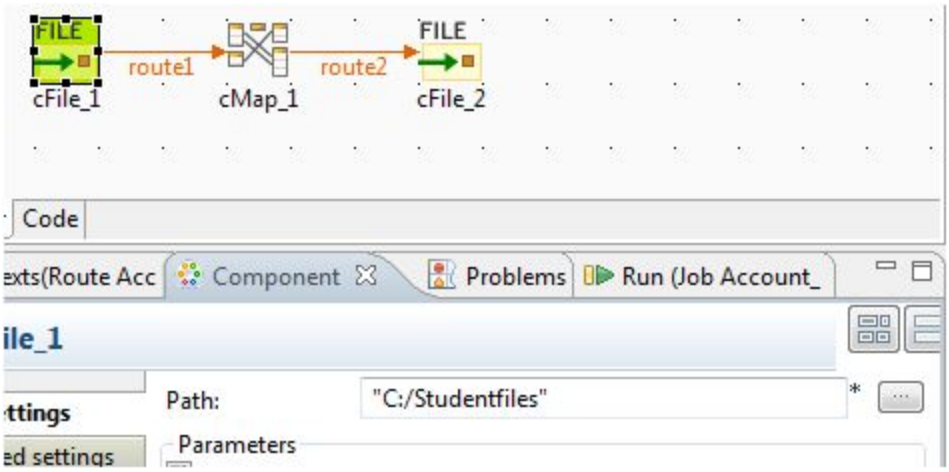


7. Right-click *cMap*, select **Row > Route** and drop the route on top of *cFile_2*.

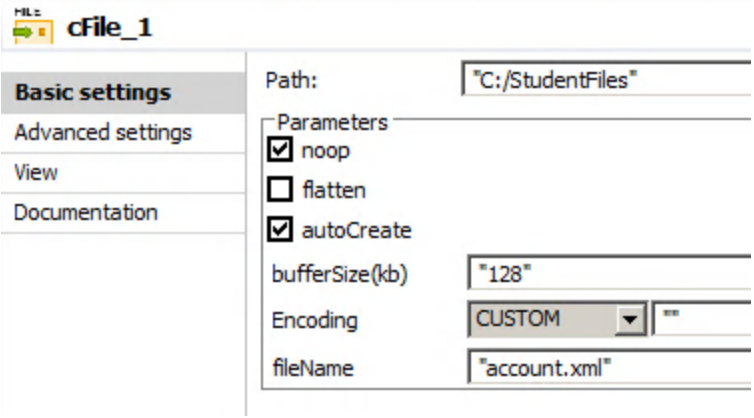


Configuring the cFile Components

1. In the **Component** view of *cFile_1*, click the **...**button to the right of the Path field and select *C:/StudentFiles*:

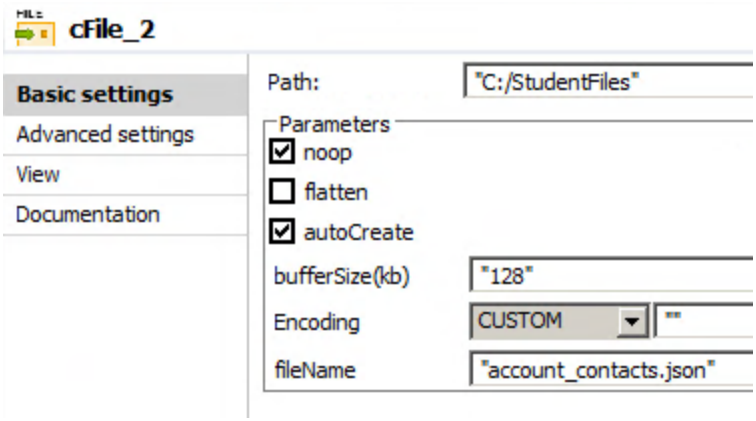


2. Enter *"account.xml"* in the **fileName** field.



3. Configure *cFile2* to drop a JSON file in the same **Path** (*"C:/StudentFiles"*) with a **fileName** equal to *"account_*

contacts.json".

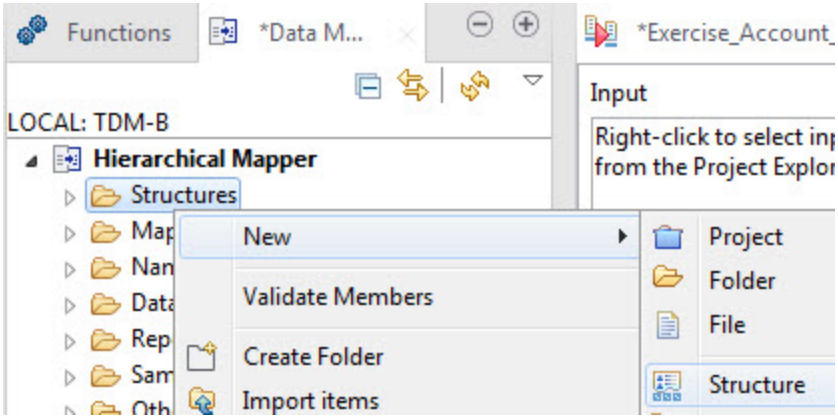


Next Step

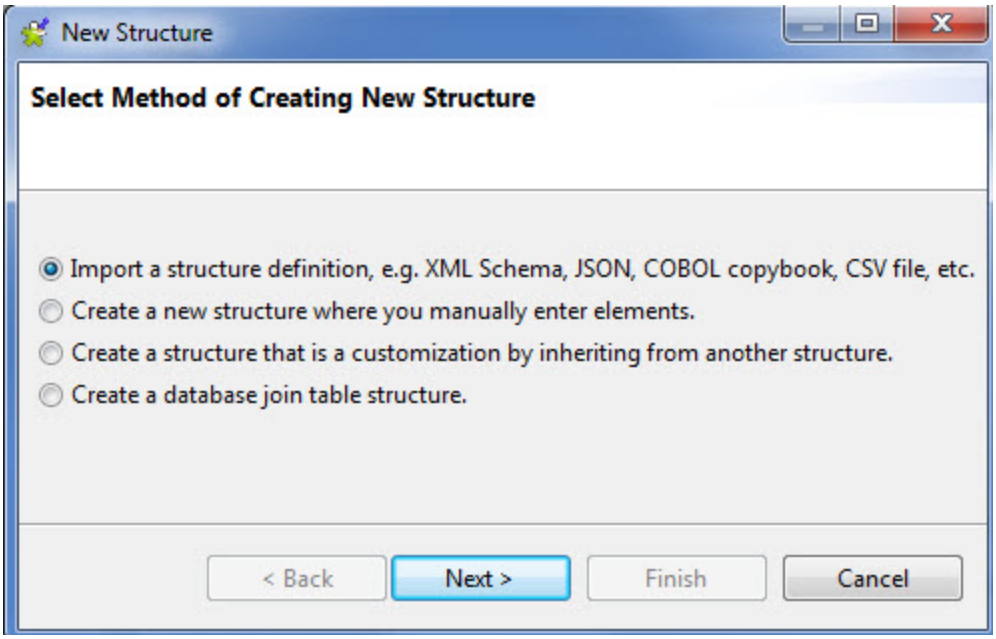
The **cMap** Component will be configured later in the exercise. Let's now define the Structures and Mappings using in *Talend Data Mapper*.

Importing the Input XML Structure

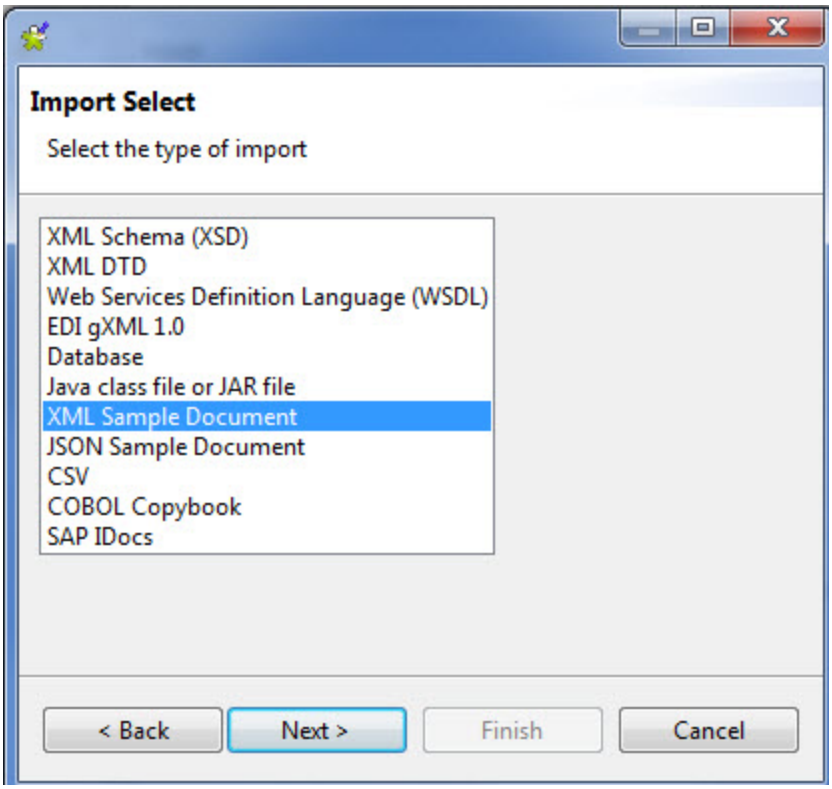
1. Switch to the **Mapping** Perspective and the **Data Mapper** tab. Right-click **Structures** and select **New > Structure**:



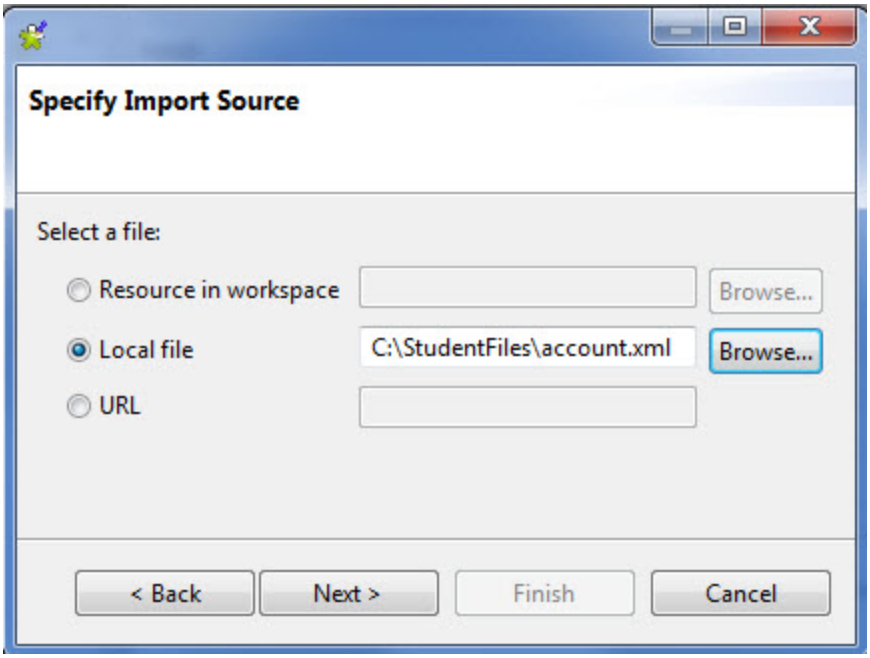
2. Select the **Import a structure definition, e.g. XML Schema, JSON, COBOL copybook, CSV file, etc.** option and click **Next**:



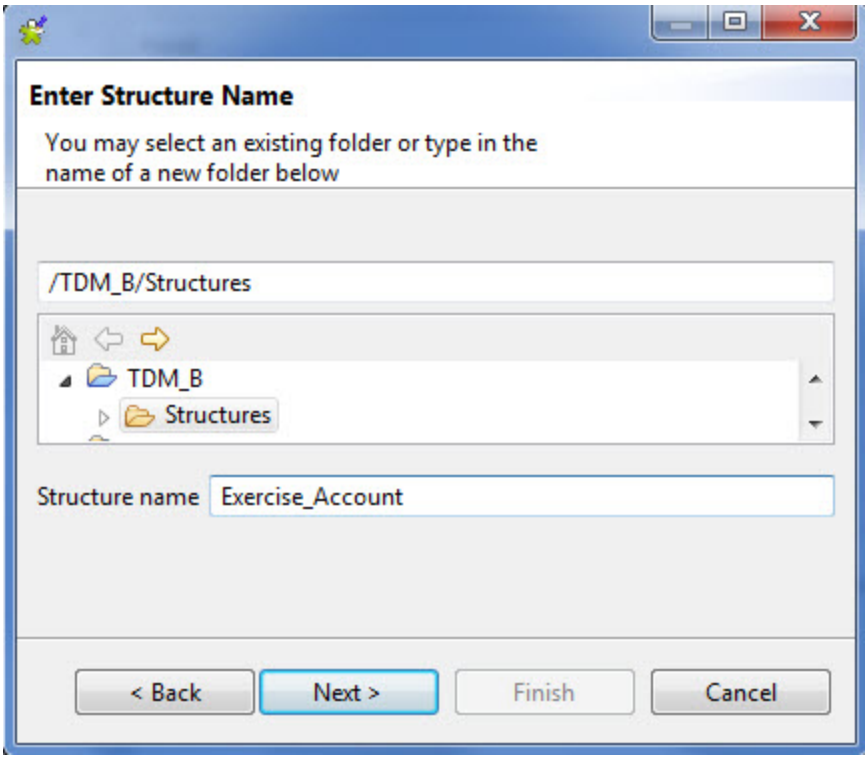
3. Select **XML Sample Document** and click **Next**.



4. Check **Local file**, click **Browse...**to select **C:\StudentFiles\account.xml** then click **Next**:



5. Name the new Structure **Exercise_Account**, click **Next** and then **Finish**:

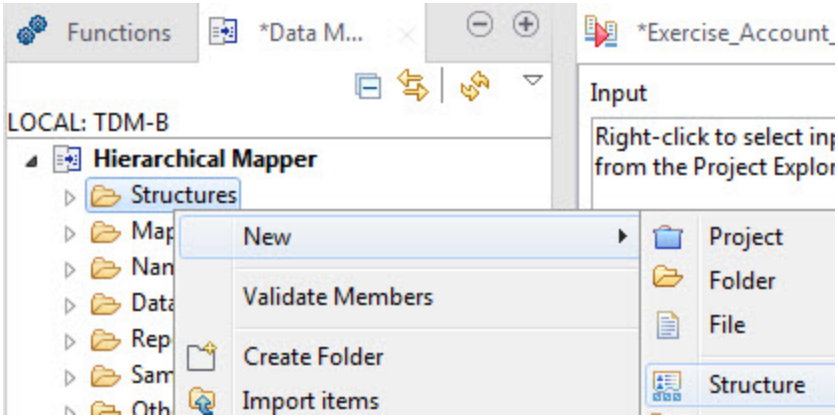


Next Step

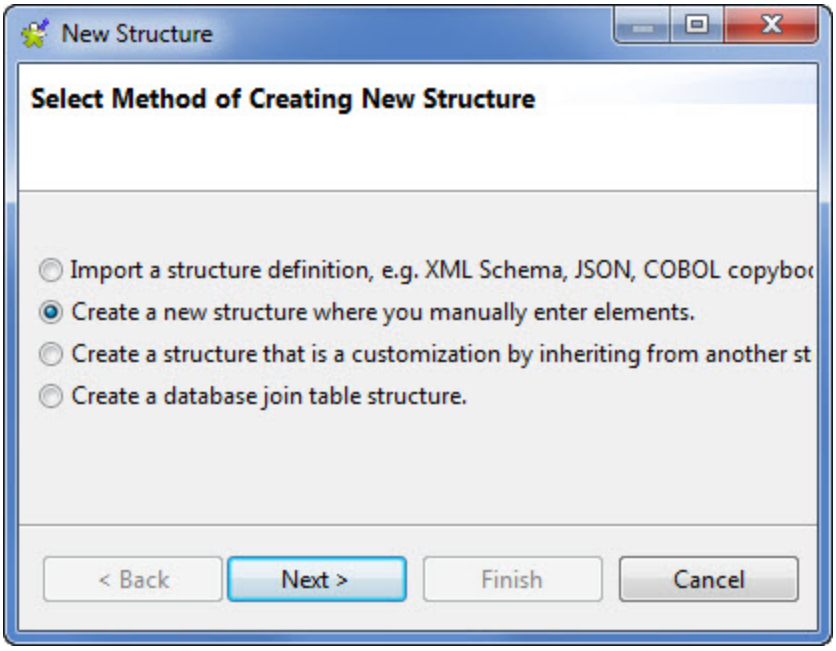
With the XML Structure imported, let's create the JSON Structure.

Creating the Output JSON Structure

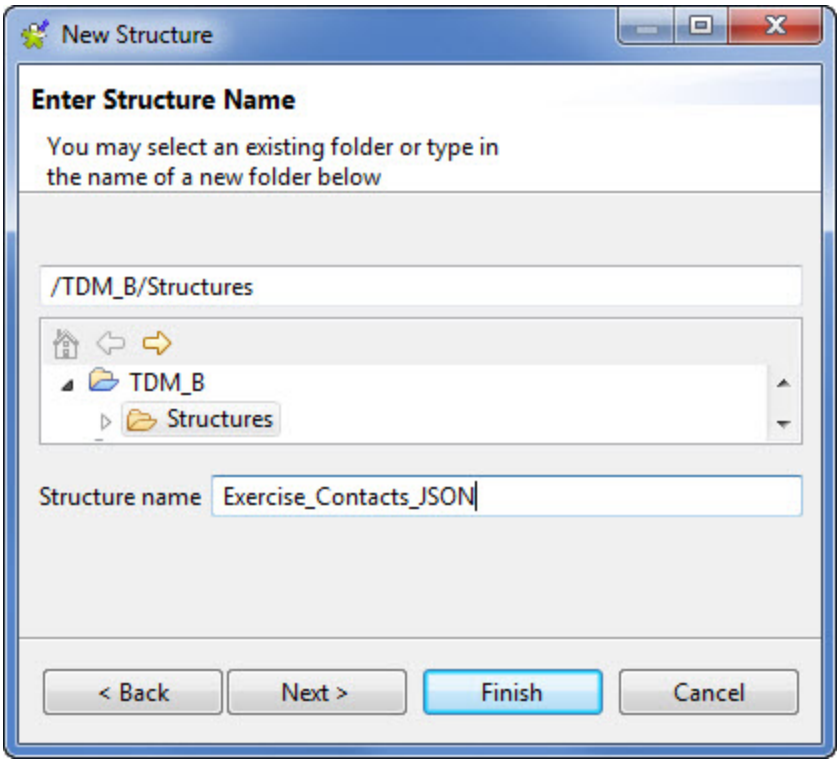
1. Right-click **Hierarchical Mapper > Structures** and select **New > Structure**:



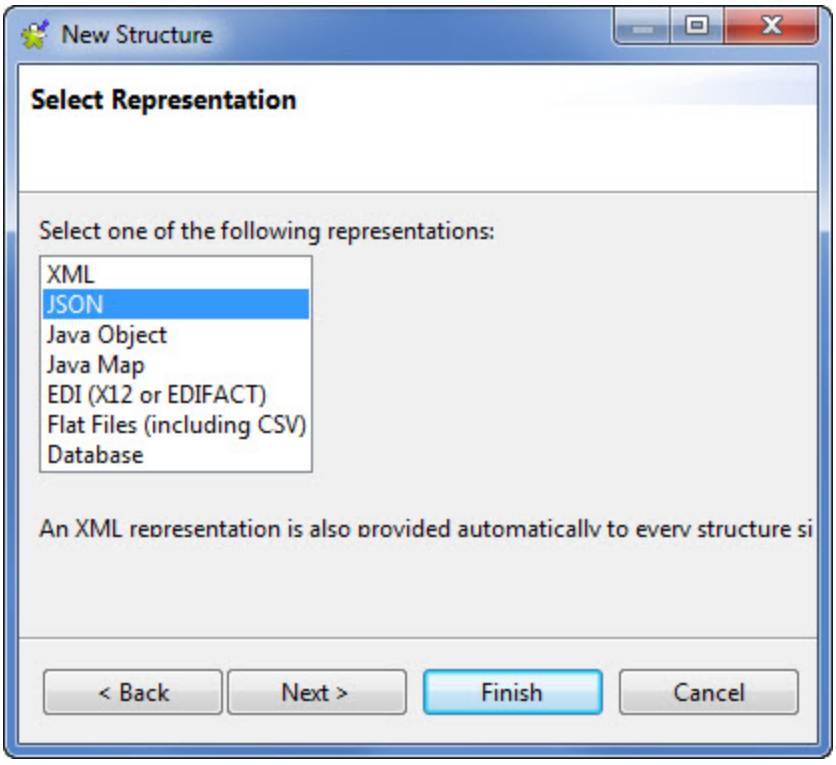
2. Check the **Create a new structure where you manually enter elements** option and click **Next**.



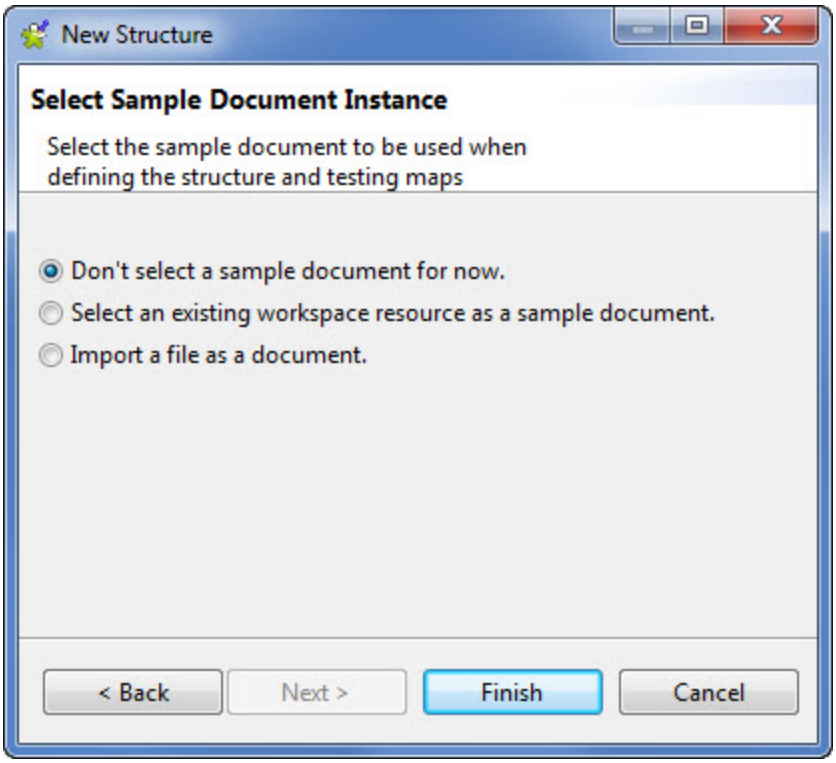
3. Select the **TDM-Essentials > Structures** folder and name the Structure **Exercise_Contacts_JSON** before clicking **Next**:



4. Select a **JSON** representation and click **Next**.

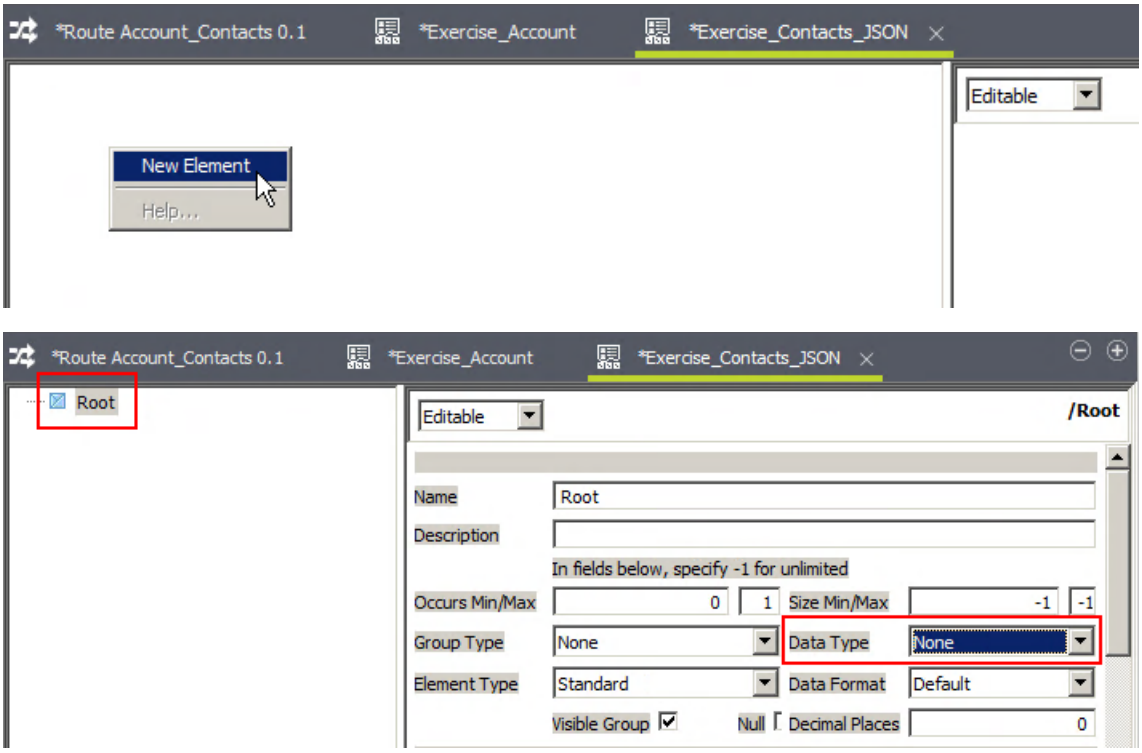


5. Select **Don't select a sample document for now** and click **Finish**:



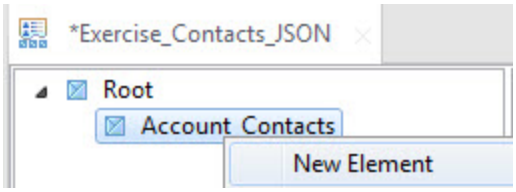
Adding Elements

1. Right-click the design area and select **New Element**. Name it *Root* and set the **Data Type** to *None* (last item in the list):



2. Right-click the *Root* element and select **New Element**to create a child node. Name it *Account_Contacts*.

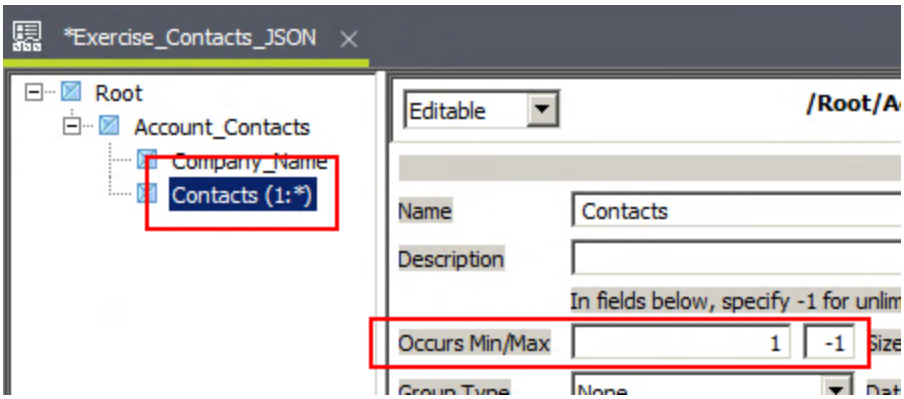
3. Right-click *Account_Contacts*, select **New Element** and name the new child *Company_Name*.



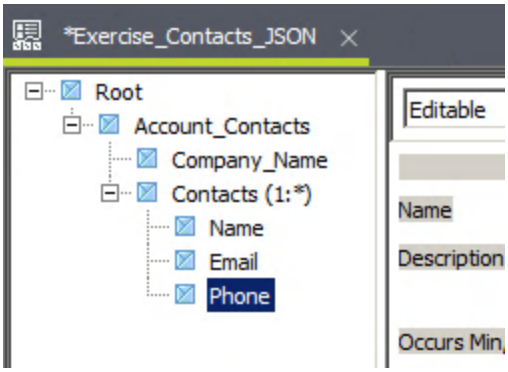
4. Repeat the previous step to create a *Contacts* node under *Account_Contacts*.

   This node will contain a list of contacts, so let's make it a looping element by entering *1* and *-1* in the **Occurs Min/Max** fields.

   Notice the node name changes from *Contacts* to *Contacts (1:*)* as soon as these parameters are entered.



5. Create three additional nodes under *Contacts (1:*)*: *Name*, *Email*, and *Phone*.



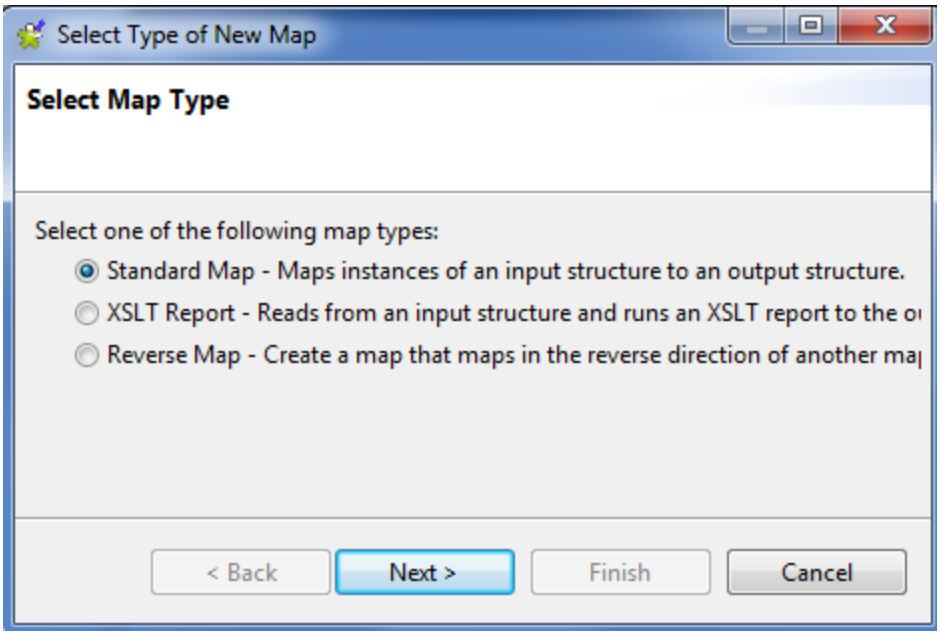
6. Save the Structure.

Next Step

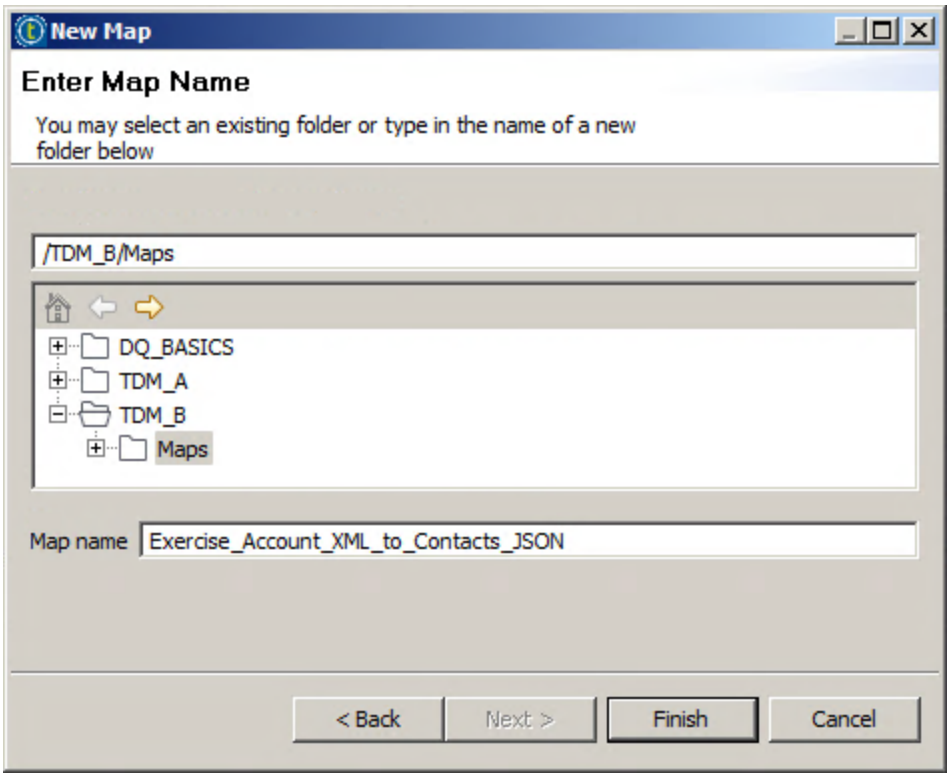
With the input and output Structures created, let's create the Map between them.

Creating the XML to JSON Mapping

1. Right-click **Hierarchical Mapper > Maps** and select **New > Map**.
2. Select a **Standard Map** and click **Next >**:

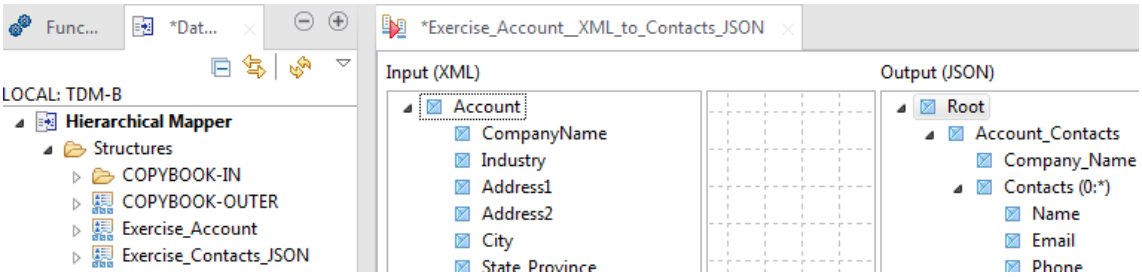


3. Name the new Map *Exercise_Account_XML_to_Contacts_JSON* and click **Finish**.

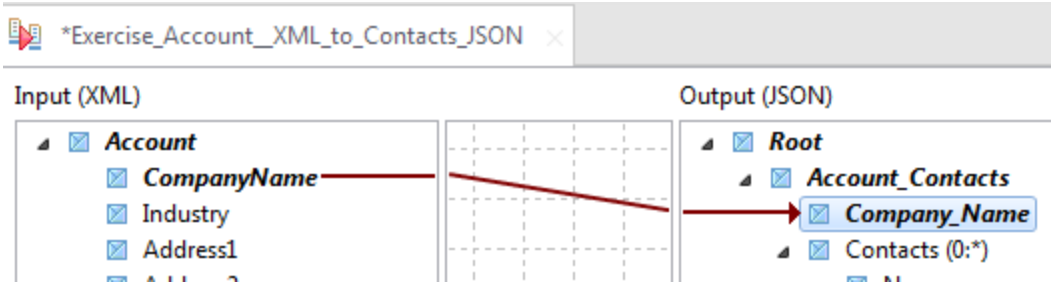


4. Drag and drop the **Exercise_Account** Structure to the Input area.

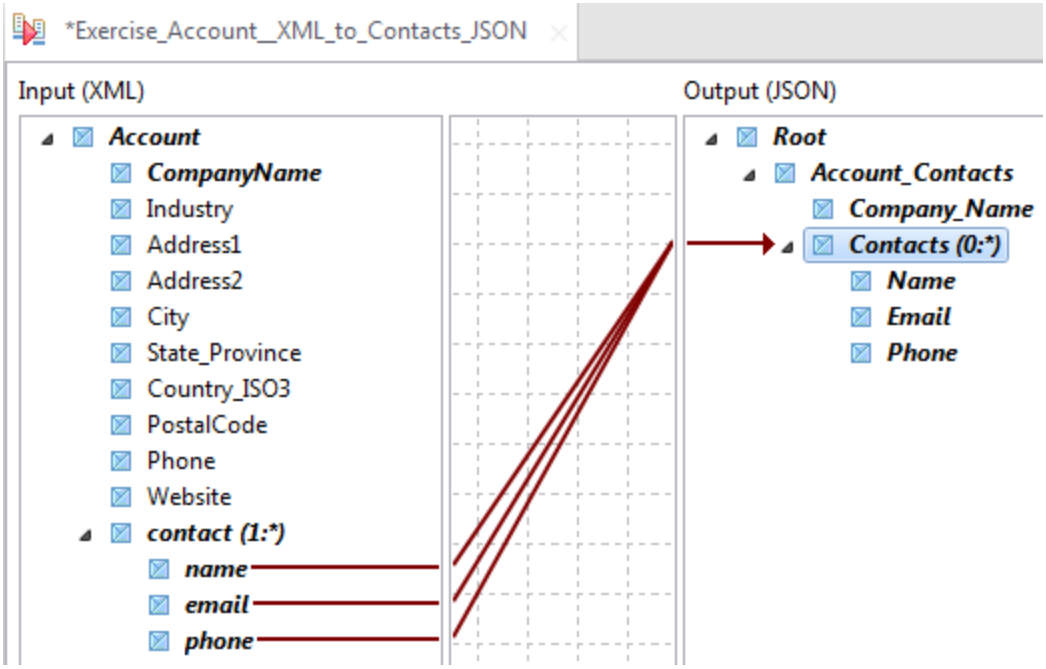
Drag and drop the **Exercise_Contacts_JSON** Structure to the Output area.



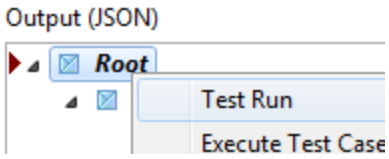
5. Map **Account > CompanyName** to **Root > Account_Contacts > Company_Name**.



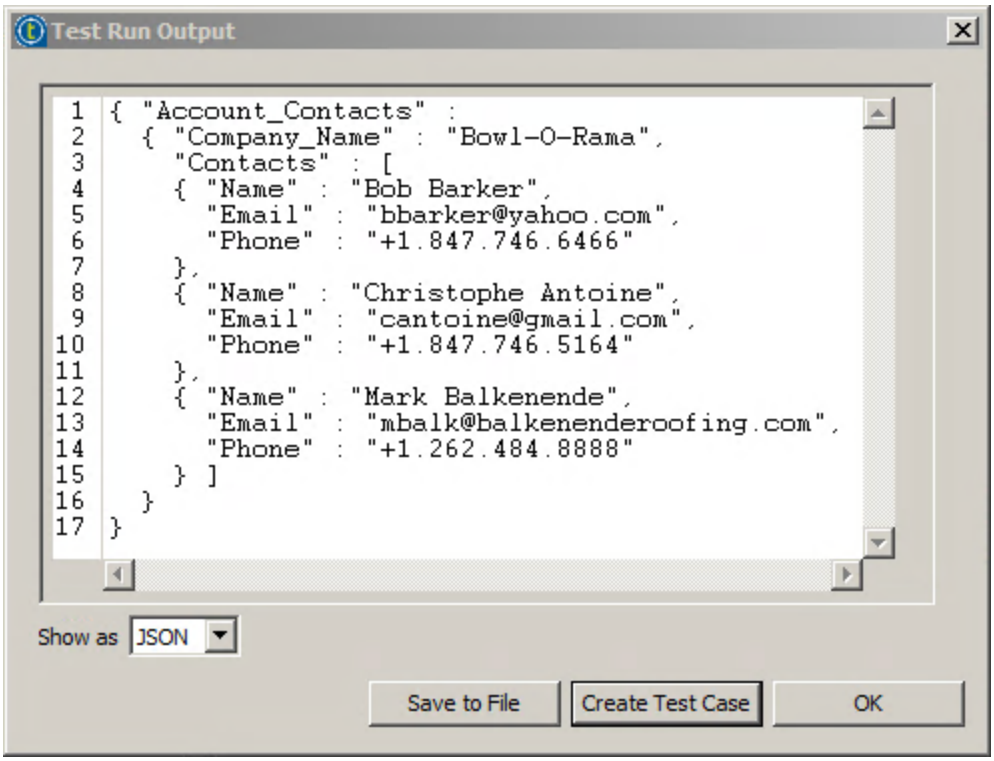
6. Map **Account > contact** to **Root > Account_Contacts > Contacts**. Note that the mapping is done automatically for nodes whose names match.



7. Save the Map.
8. Right-click *Root* in the **Output** area and select **Test Run**.



9. The following JSON document should appear in the output window.

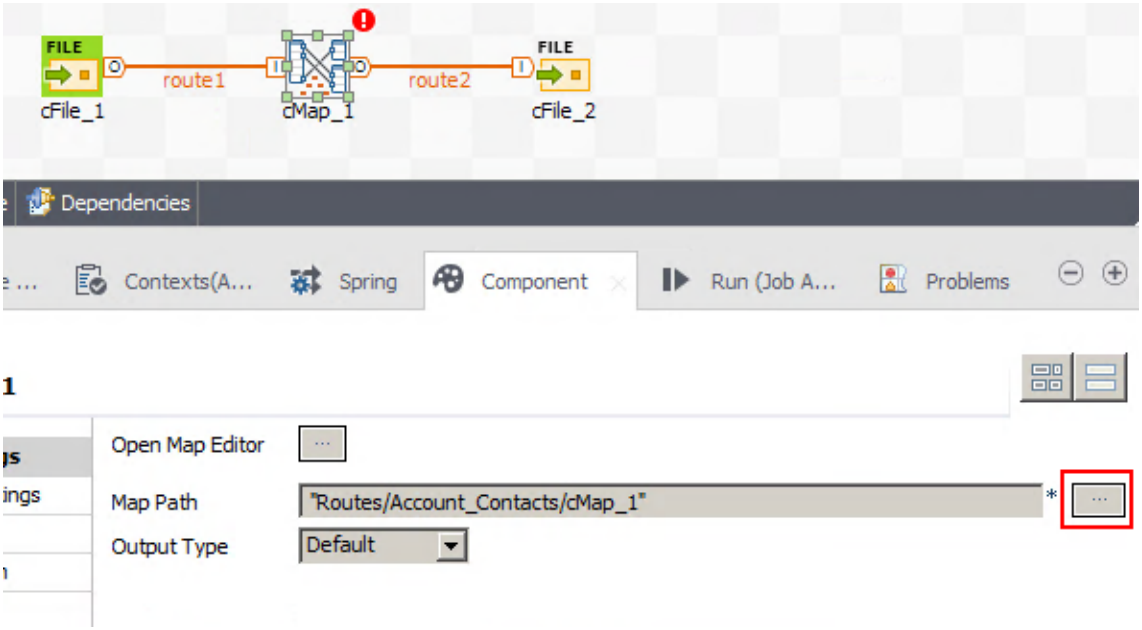


Next Step

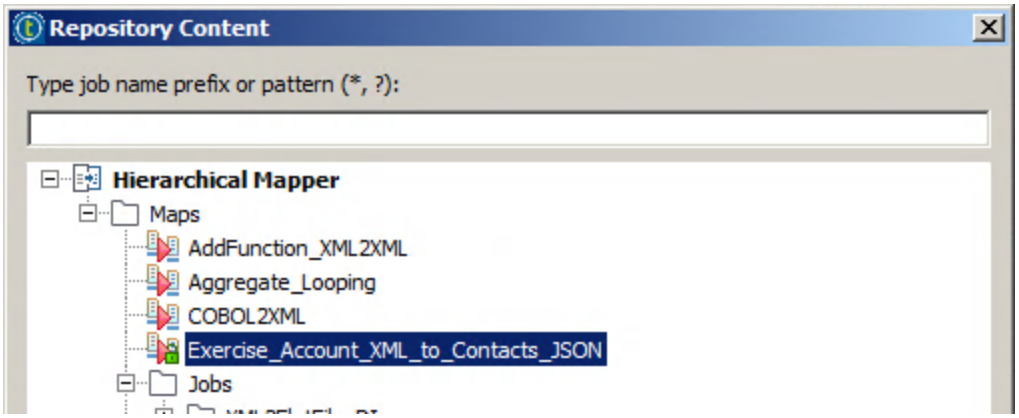
With the Map completed, let's configure the **cMap** Component in the Route created in the first step.

Configure cMap and Testing the Route

1. Switch back to the **Mediation** perspective and the *Account_Contacts* Route in **Repository > Routes**.
2. Double-click the **cMap** Component to display its **Component** View.
3. Click the **...** button to the right of **Map Path** to select the Map that will be used to perform the actual transformation.

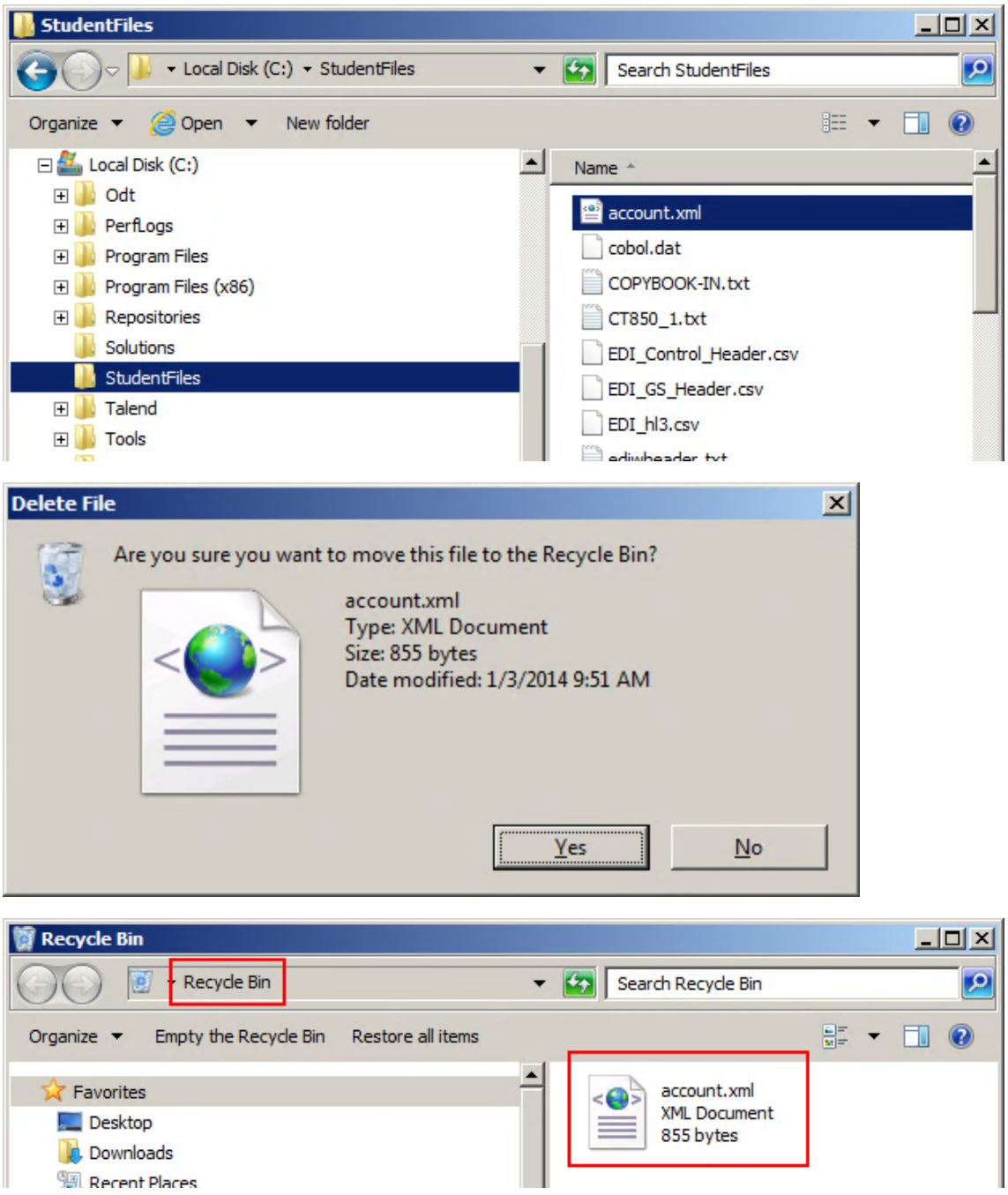


4. Select the **Exercise_Account_XML_to_Contacts_JSON** Map created in a previous step and click **OK**.

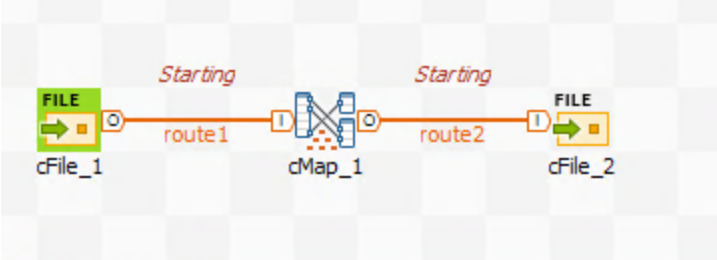


5. It's time to test the "polling" effect of the *cFile_1* Component. This Component is responsible for listening to changes in the *account.xml* file, so let's remove it temporarily and restore it later on from the Recycle Bin to simulate a new file appearing all of a sudden. Imagine this file is created by another application.

   Delete *C:\StudentFiles\account.xml* to send it to the Recycle Bin.



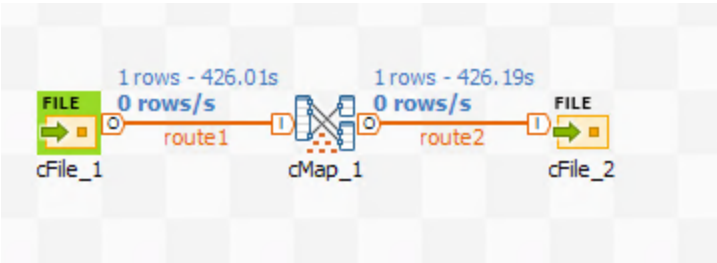
6. Click **Run** to run the Route and examine the canvas.

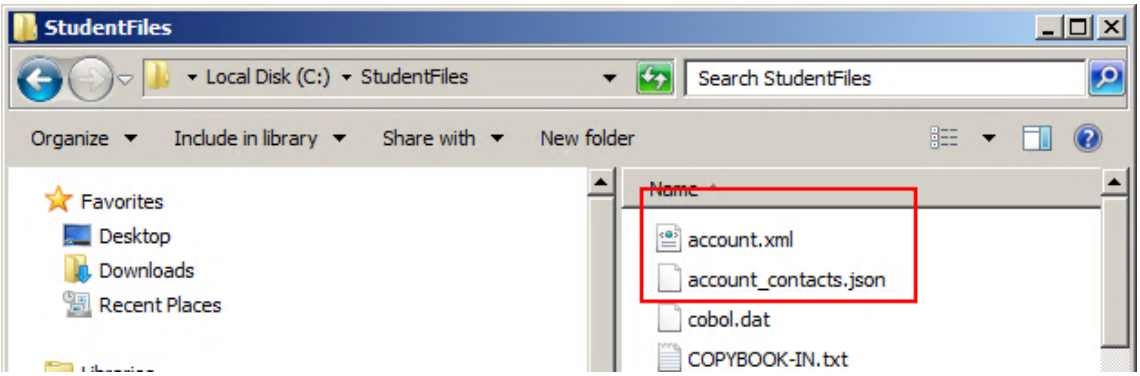


Note that nothing happens as long as *account.xml* is still in the Recycle Bin and does not exist physically in the *C:/StudentFiles* folder. The Route will remain in the **Starting** state until this specific file appears on disk, and *account_contacts.json* is not created either. The Route continuously polls the *C:/StudentFiles* folder, looking for changes in *account.xml* and processing it just once as soon as it appears.

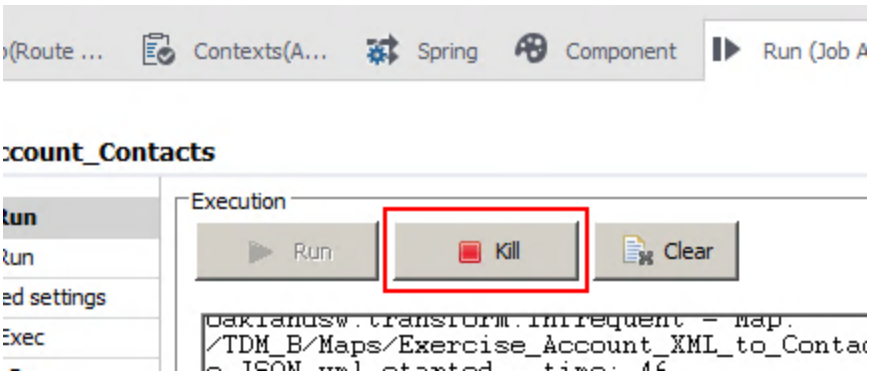
7. Now recover the input file from the Recycle Bin to simulate a file creation. Open the **Recycle Bin** on the Desktop, select *account.xml* and click **Restore this item** in the toolbar:

8. Examine the canvas again and note that the Route is no longer in the **Starting** state. It processed *account.xml* and created the output JSON file.

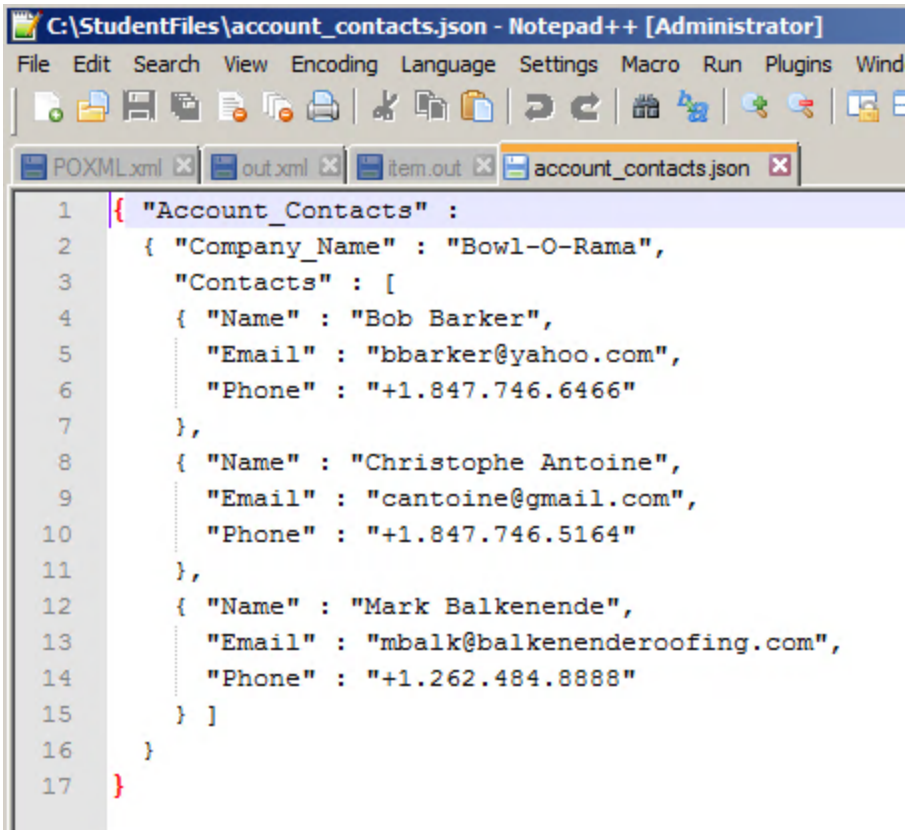




9. Click **Kill** to stop the execution.



10. Finally, open **C:\StudentFiles\account_contacts.json** in Notepad++ to examine its contents.



Next Step

This lesson is almost over. Head to the Wrap-Up section for a summary of the concepts reviewed in this lesson.

Wrap-Up

In this lesson, you learned how to:

- Create a JSON Structure by adding and configuring nodes manually. The same method can be applied to XML, COBOL...
- Use the **cMap** Component in Routes in the **Mediation** Perspective. It demonstrates the integration between the various Talend products. A Map can be reused in other perspectives/products.

Next Step

Congratulations, you successfully completed this lesson. Click the **Check your status with this unit** button below in order to save your progress. Then click **Completed. Let's continue >** on the next screen to jump to the next lesson.

This page intentionally left blank to ensure new chapters start on right (odd number) pages.

LESSON 9

Refactoring

This chapter discusses the following.

Overview

Lesson Overview

This lesson focuses on refactoring best practices. Suppose you need to upgrade the Structure of an element to a new version of a standard. The new standard is very similar tot the old one, but you still need to map all the elements again. What are the best practices for this kind of refactoring? This lab provides tips and tricks on how to perform this operation for two different Maps and the associated Structures.

Objectives

After completing this lesson, you will be able to:

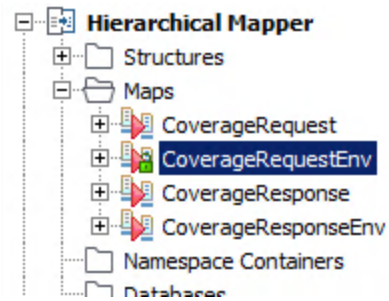
- Change the inheritance and architecture of an existing Structure
- Move the old mapping expressions to the new elements

Next Step

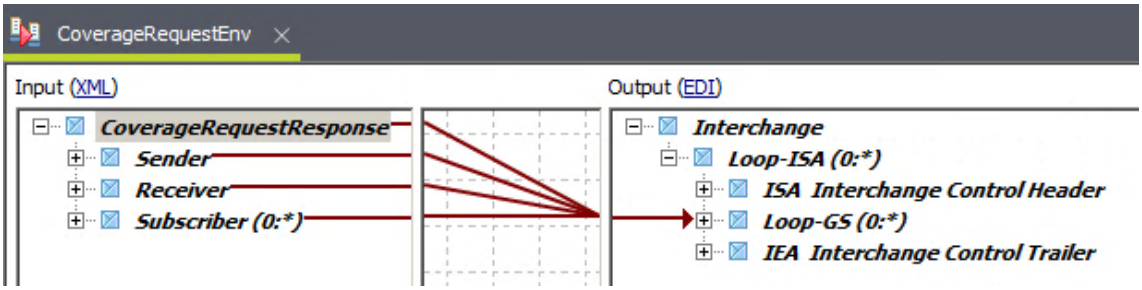
First, let's focus on upgrading the Output inheritance.

Upgrading the Output Inheritance

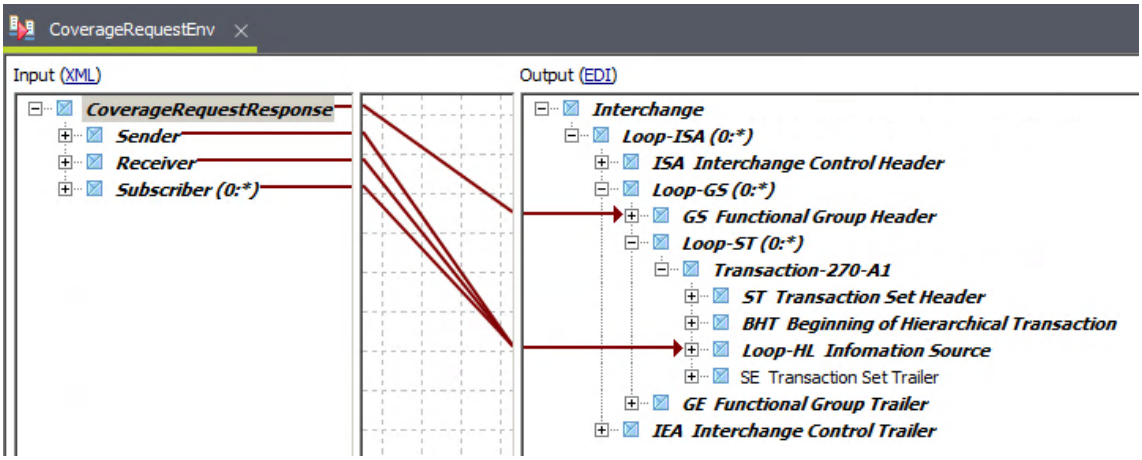
1. Double-click the existing **Hierarchical Mapper > Maps > CoverageRequestEnv** Map to open it.



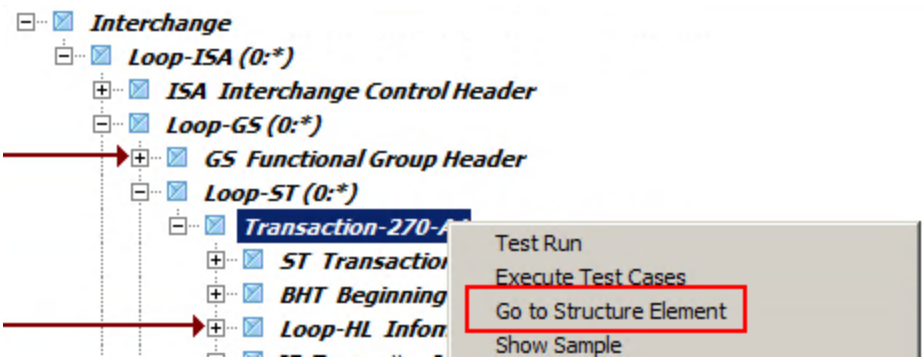
2. Expand the output Structure until the contents of the **Loop-ISA (0:*)**element are visible.



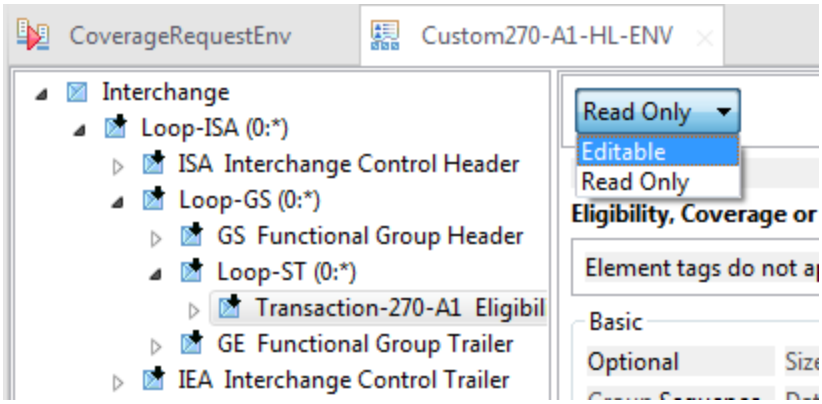
3. Further expand **Loop-GS (0:*) > Loop-ST (0:*) > Transaction-270-A1** to reveal the **Loop-HL Information Source** element.



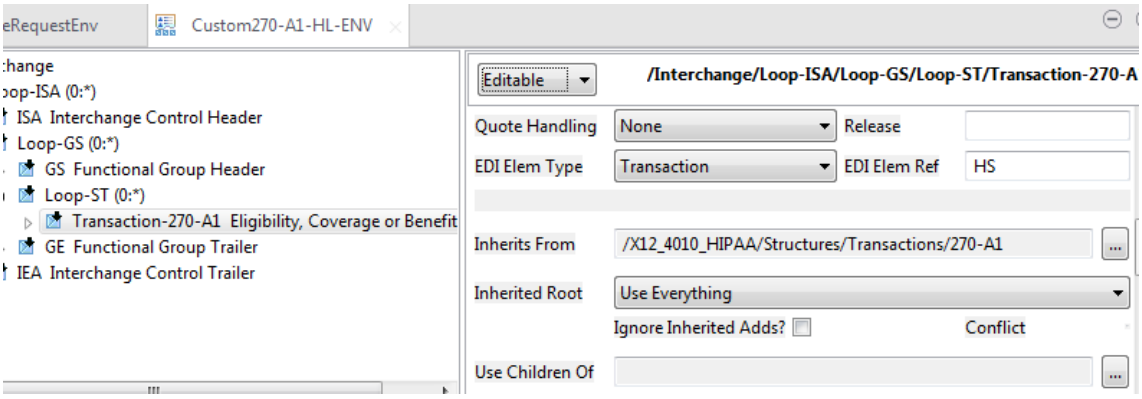
4. Right-click **Transaction-270-A1** and select **Go to Structure Element** to jump to the element definition.



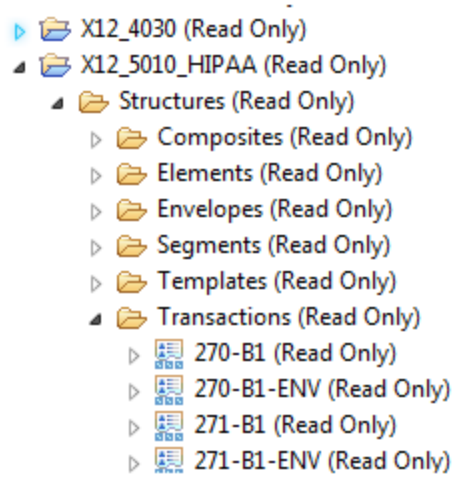
5. Change **Read Only** to **Editable** to be able to modify the definition of the element (type, size...).



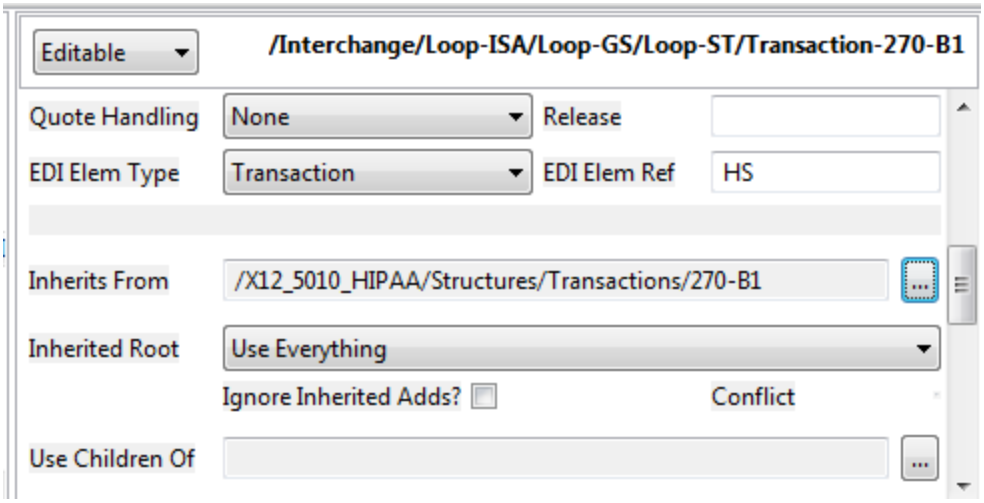
6. Scroll down to the **Inherits From** field and click the **...** button next to it to modify the inheritance of this specific node. Note that the node currently inherits its architecture from the */X12_4010_HIPAA/Structures/Transactions/270-A1* Structure.

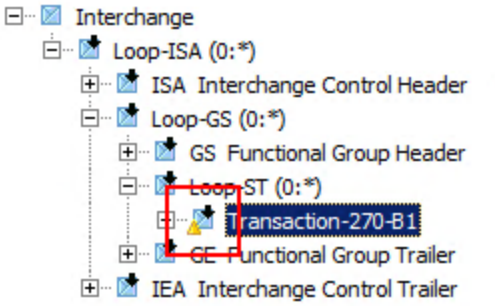


7. Select the **X12_5010_HIPAA > Structures > Transactions > 270-B1** Structure and click **OK** to change the inheritance of the current node.



8. Check that the Structure has actually changed, then save it. Also notice the yellow badge on top of the new *Transaction-270-B1* element. It means some warnings appeared due to the new inheritance. The next step is to fix these warnings by remapping the elements so they match the new architecture.



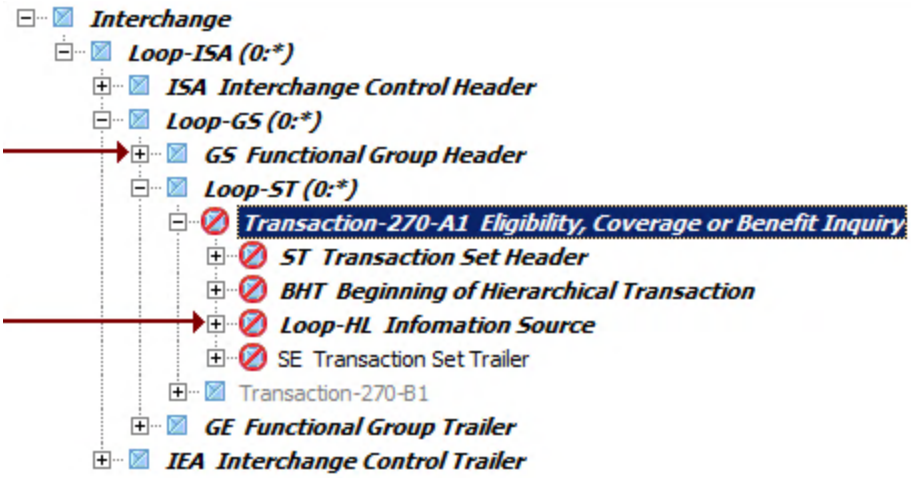


Next Step

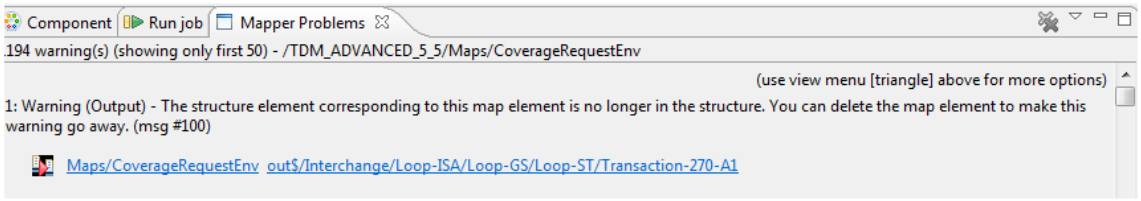
Now that the output Structure changed, let's also modify the mapping so it matches the new architecture.

Upgrading the Output Mapping

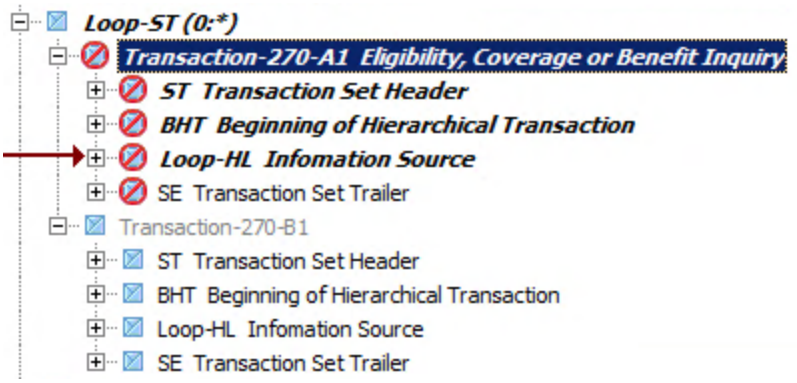
1. Display the Map again and spend some time examining the changes and warnings introduced by the new inheritance.



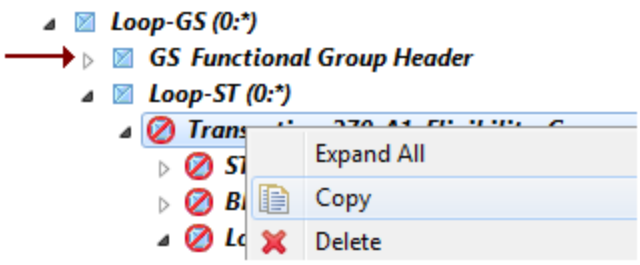
2. Also read the warnings in the **Mapper Problems** tab at the bottom of the Studio:



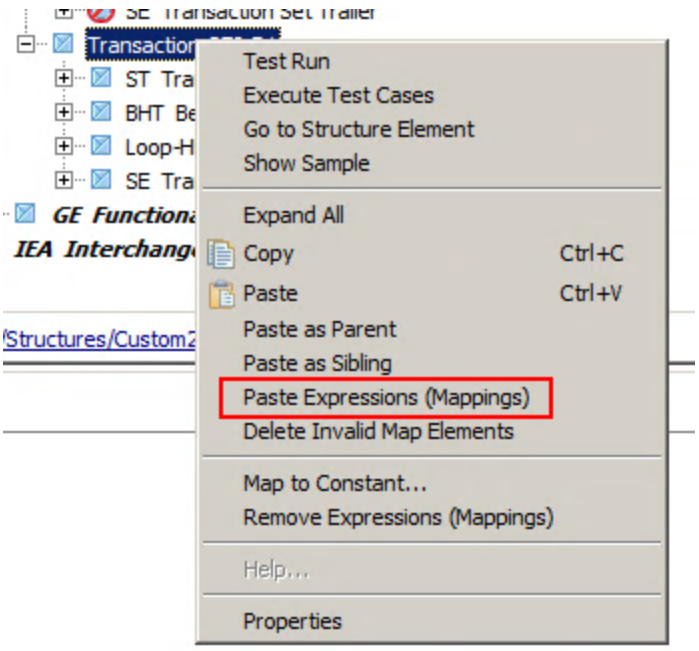
3. Finally, notice a output element called *Transaction-270-B1*. Expand it and examine the new elements which do not have a mapping associated with them (yet). Let's copy the old mappings to the new elements.



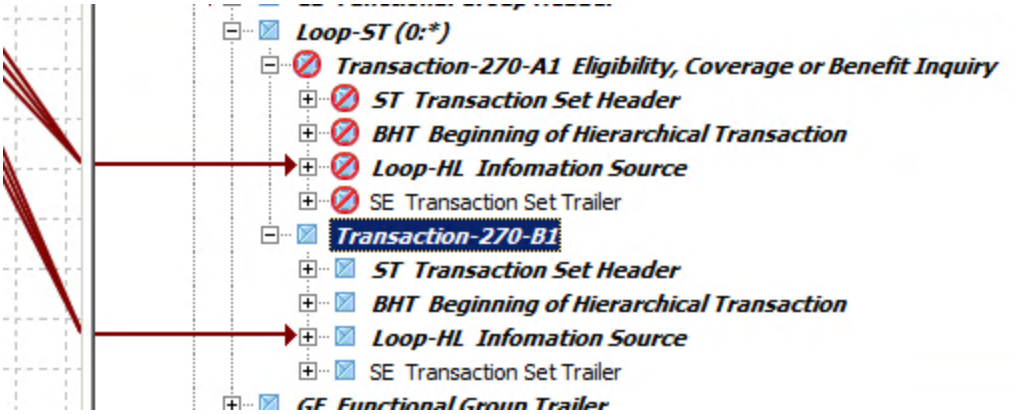
4. Right-click **Transaction-270-A1** and select **Copy** to copy the mapping expressions.



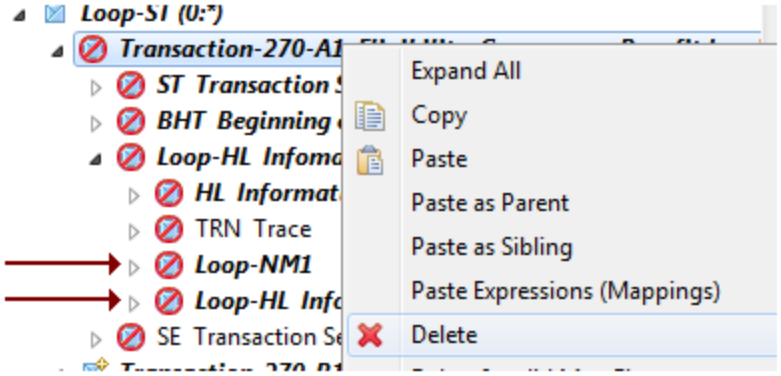
5. Right-click **Transaction-270-B1** and select **Paste Expressions (Mappings)**:



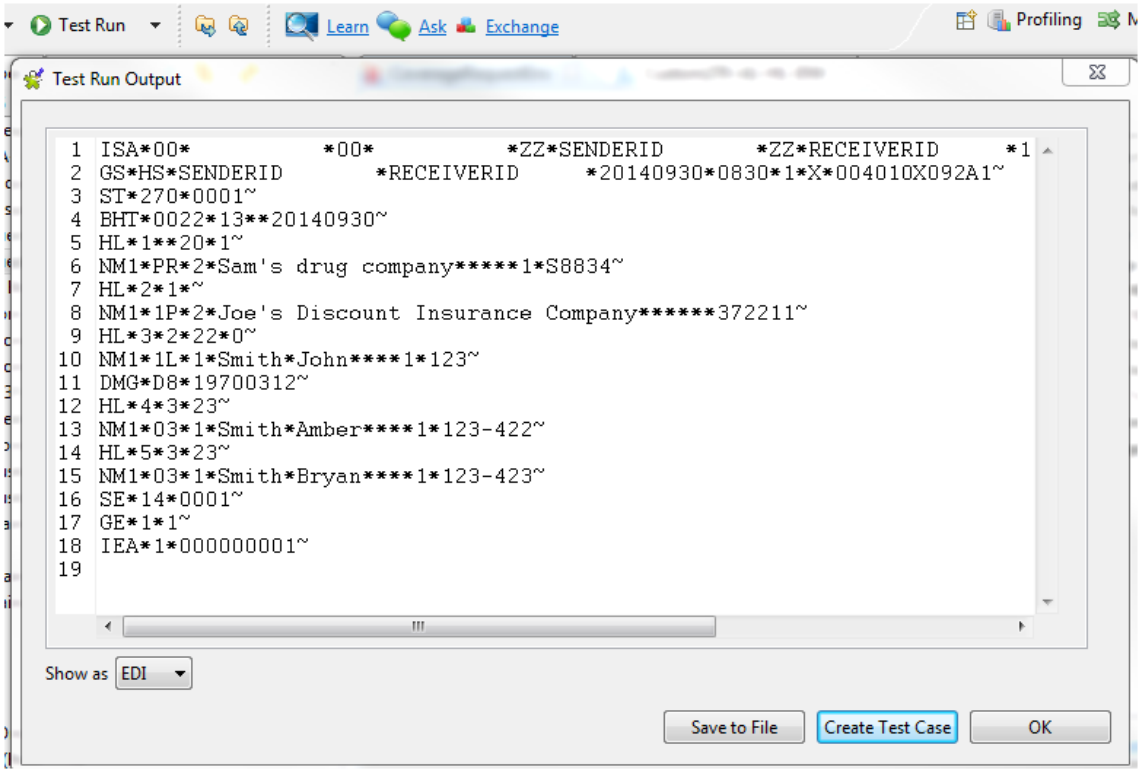
6. Notice that both the old and new elements are mapped.



7. Right-click on **Transaction-270-A1** and select **Delete** to remove the node itself and all associated mappings. Then save the Map.



8. Finally, click **Test Run** to check the output.

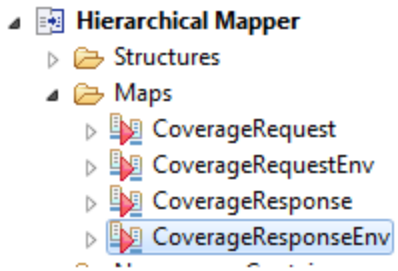


Next Step

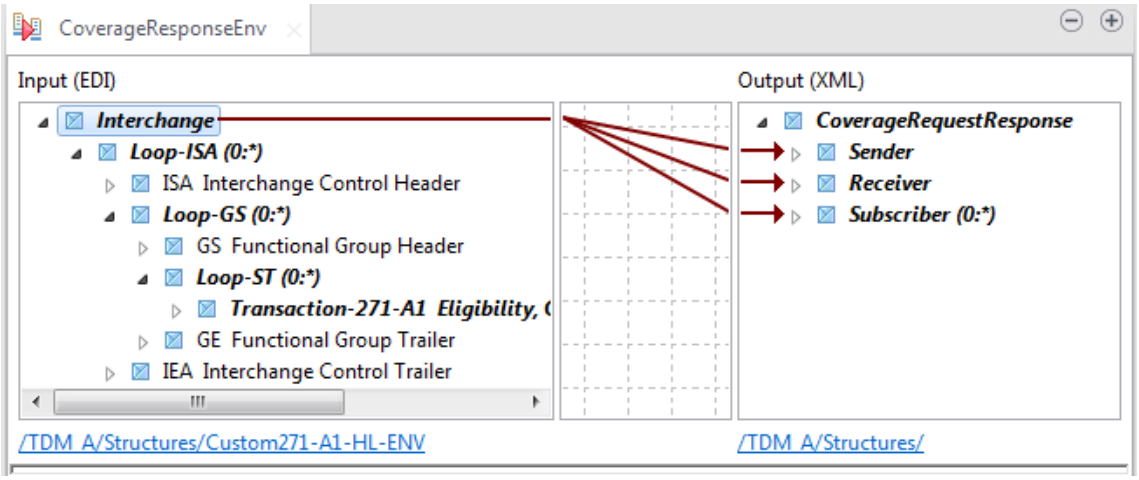
With the new mapping completed on the output side, let's modify the inheritance of the input Structure.

Upgrading the Input Inheritance

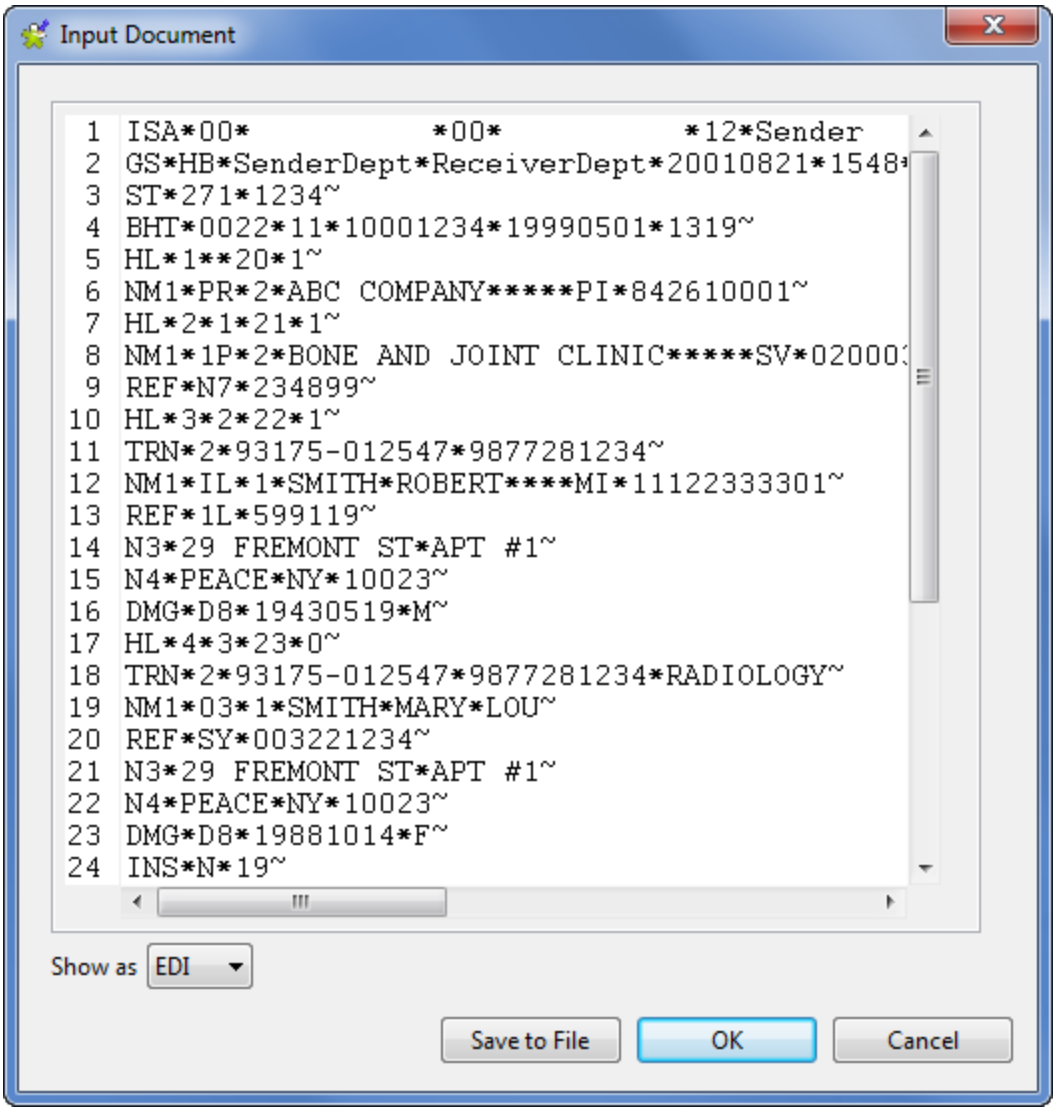
1. Double-click the **CoverageResponseEnv** Map to open it.



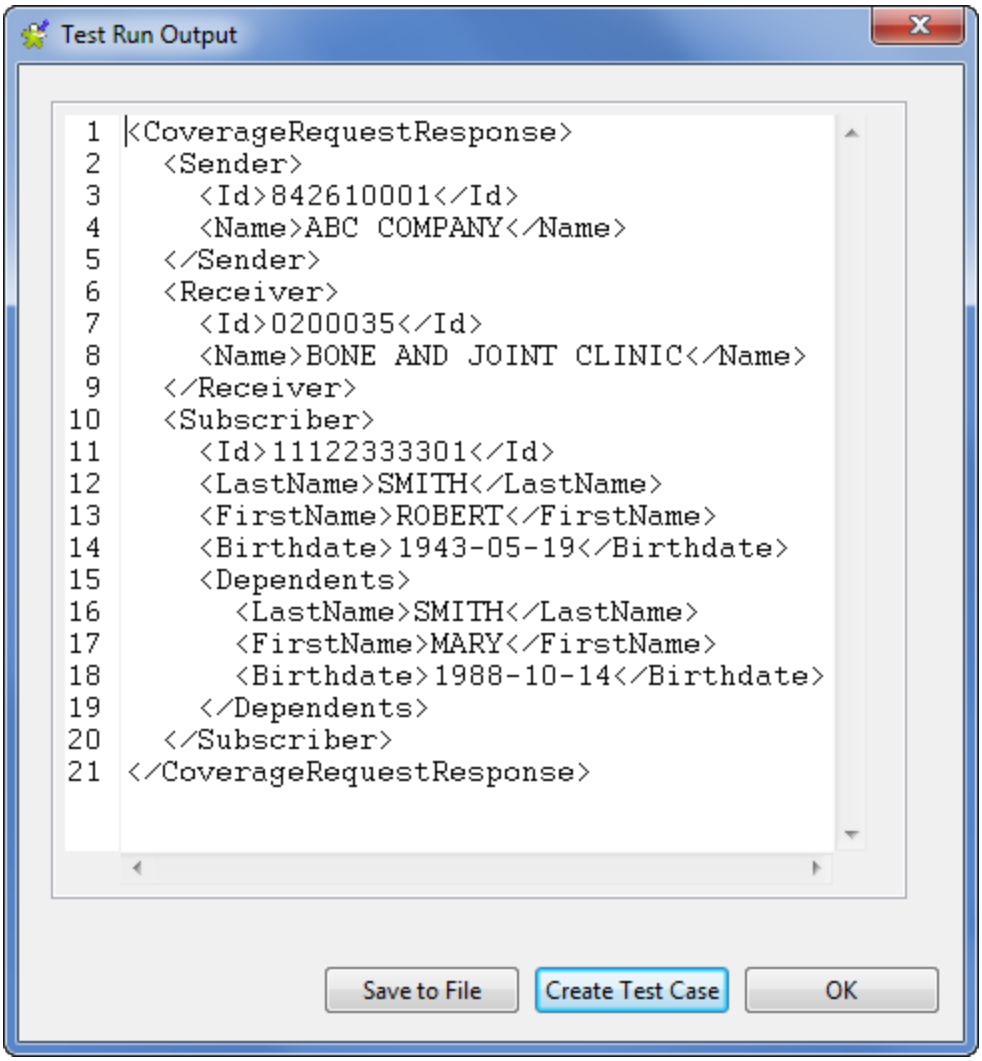
1. Expand the **Loop-ISA > Loop-GS > Loop-ST** element in the **Input** area until the *Transaction-271-A1* node is visible.



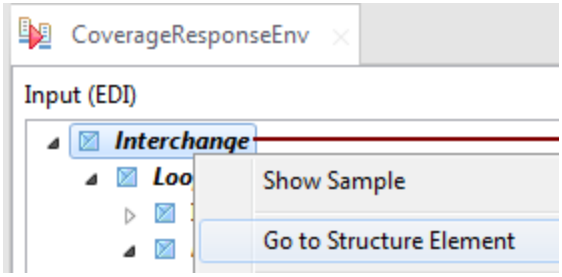
2. Click **Show Document** to display some sample data and click **OK** to close the dialog.



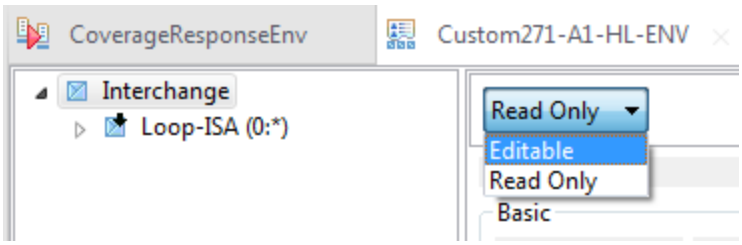
3. Click **Test Run** to generate the associated output and click **OK** again to close the dialog.



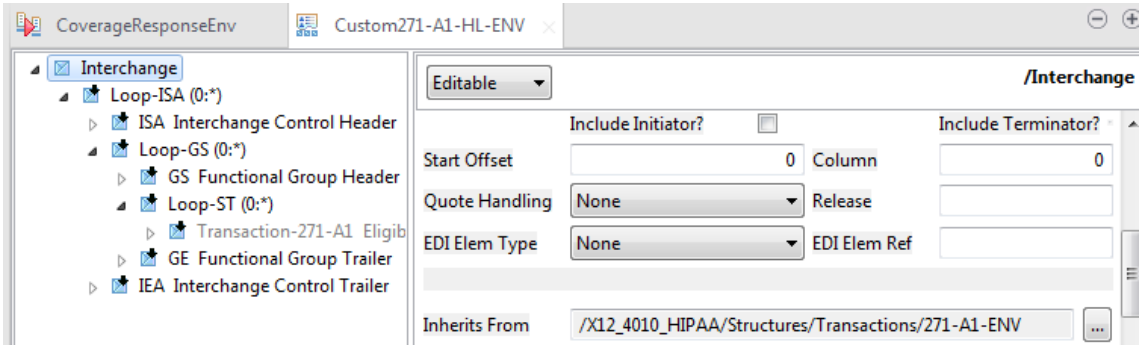
4. Right-click the *Interchange* element in the **Input** area and select **Go to Structure Element**.



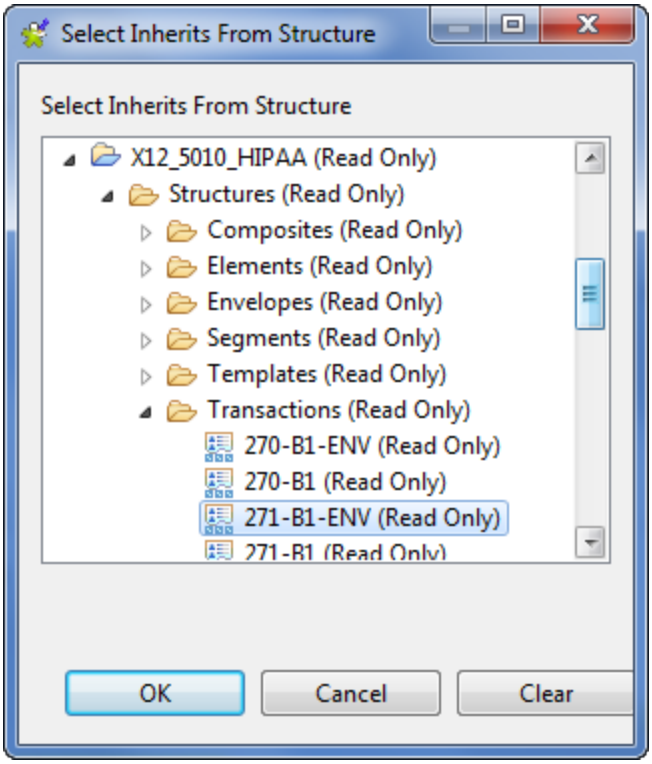
5. Change **Read Only** to **Editable**.



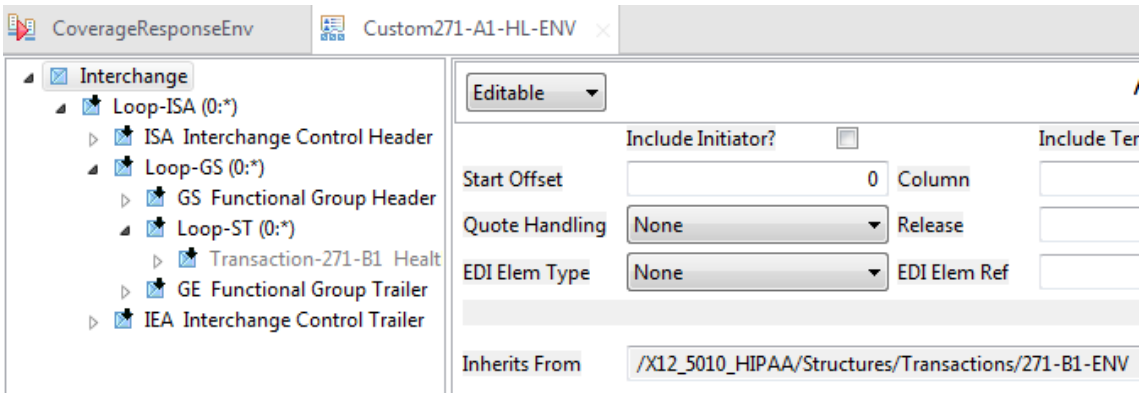
6. Scroll down to the *Inherits From* field and notice the element currently inherits from the */X12_4010_HIPAA/Structures/Transactions/271-A1-ENV* Structure.



7. Click the **...** button next to the *Inherits From* field, select **X12_5010_HIPAA > Structures > Transactions > 271-B1-ENV** Structure then click **OK** to change the inheritance of the *Interchange* element.



8. Save the updated Structure.

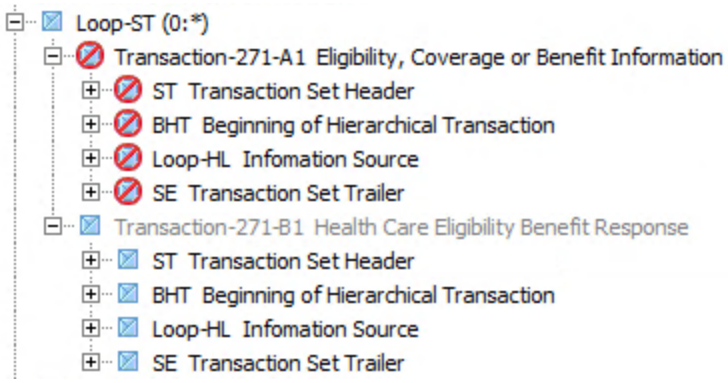


Next Step

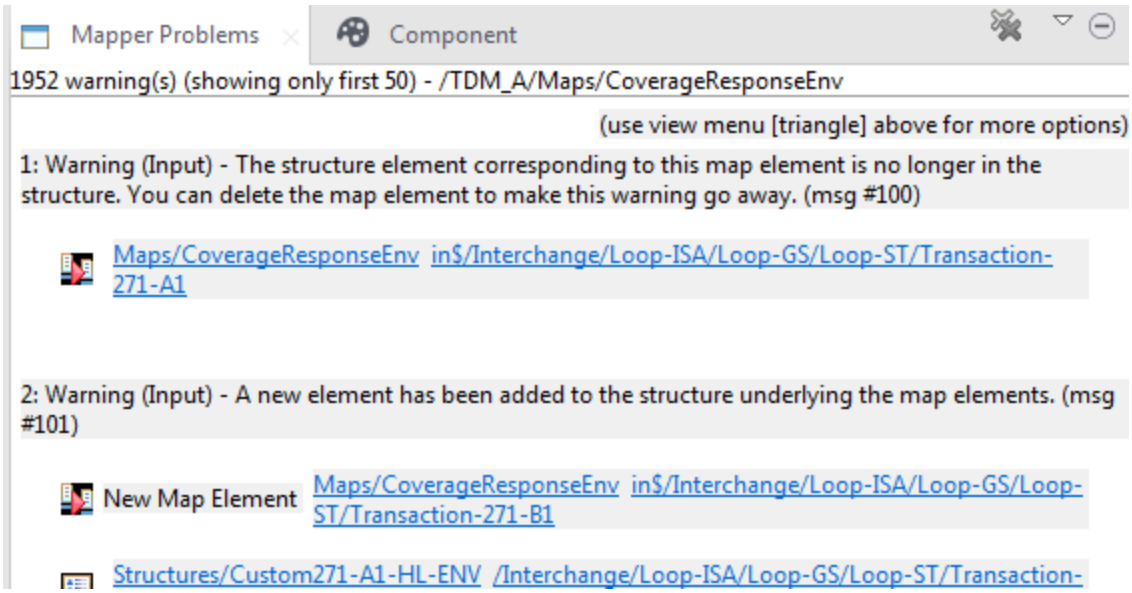
Now that the input Structure changed, let's also modify the mapping so it corresponds to the new architecture.

Upgrading the Input Mapping

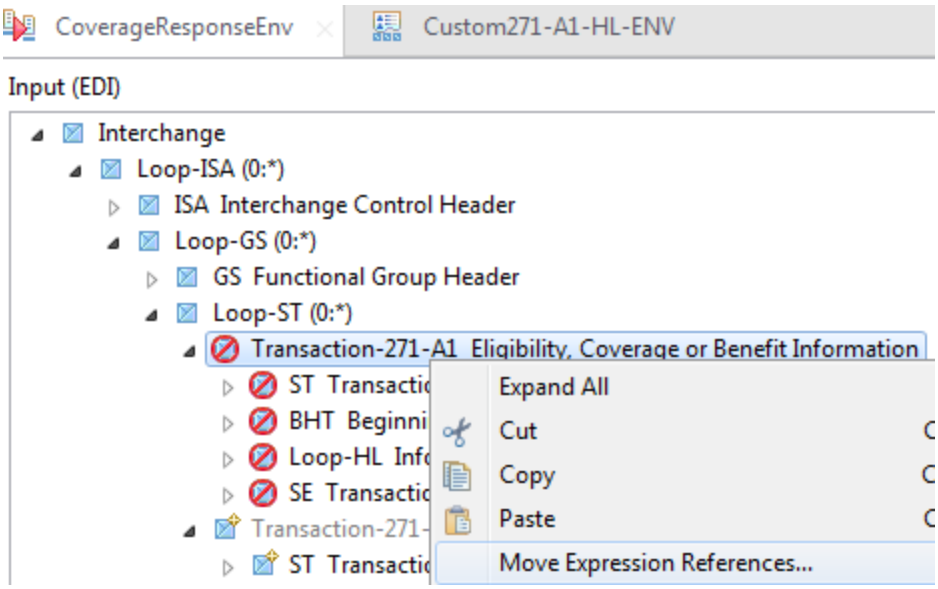
1. Display the Map again and notice the new *Transaction-271-B1* element, with no mappings associated yet.



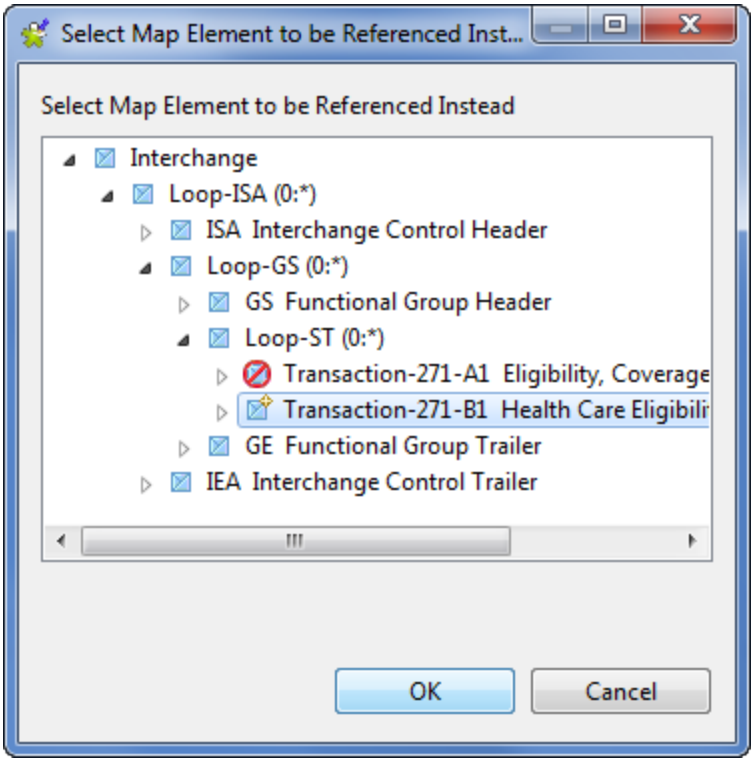
2. Examine the warnings in the **Mapper Problems** tab. Let's fix these warnings by moving the mappings from *Transaction-271-A1* to *Transaction-271-B1*.



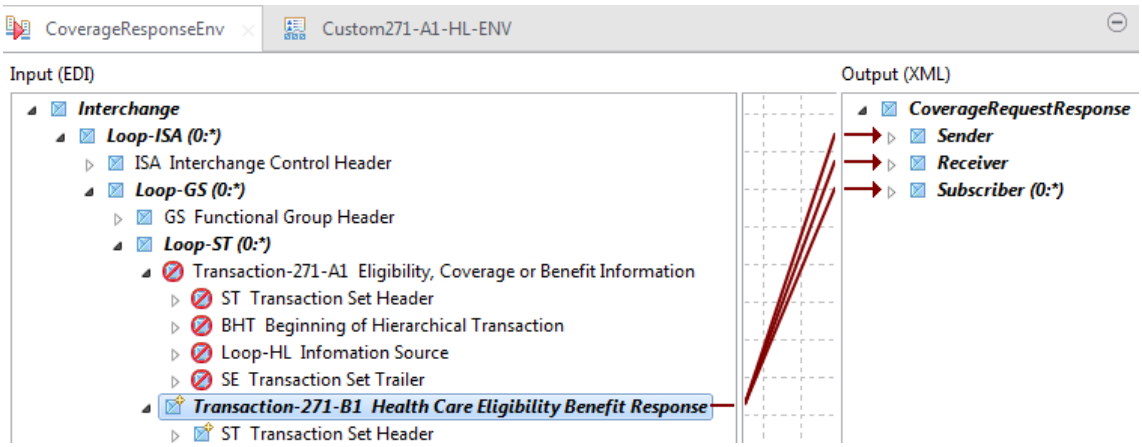
3. Right-click on *Transaction-271-A1* and select **Move Expression References**.



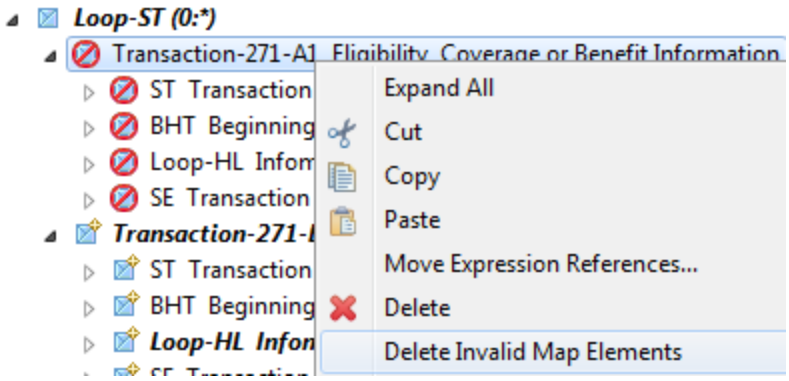
4. Select **Transaction-271-B1** as the destination and click **OK**.



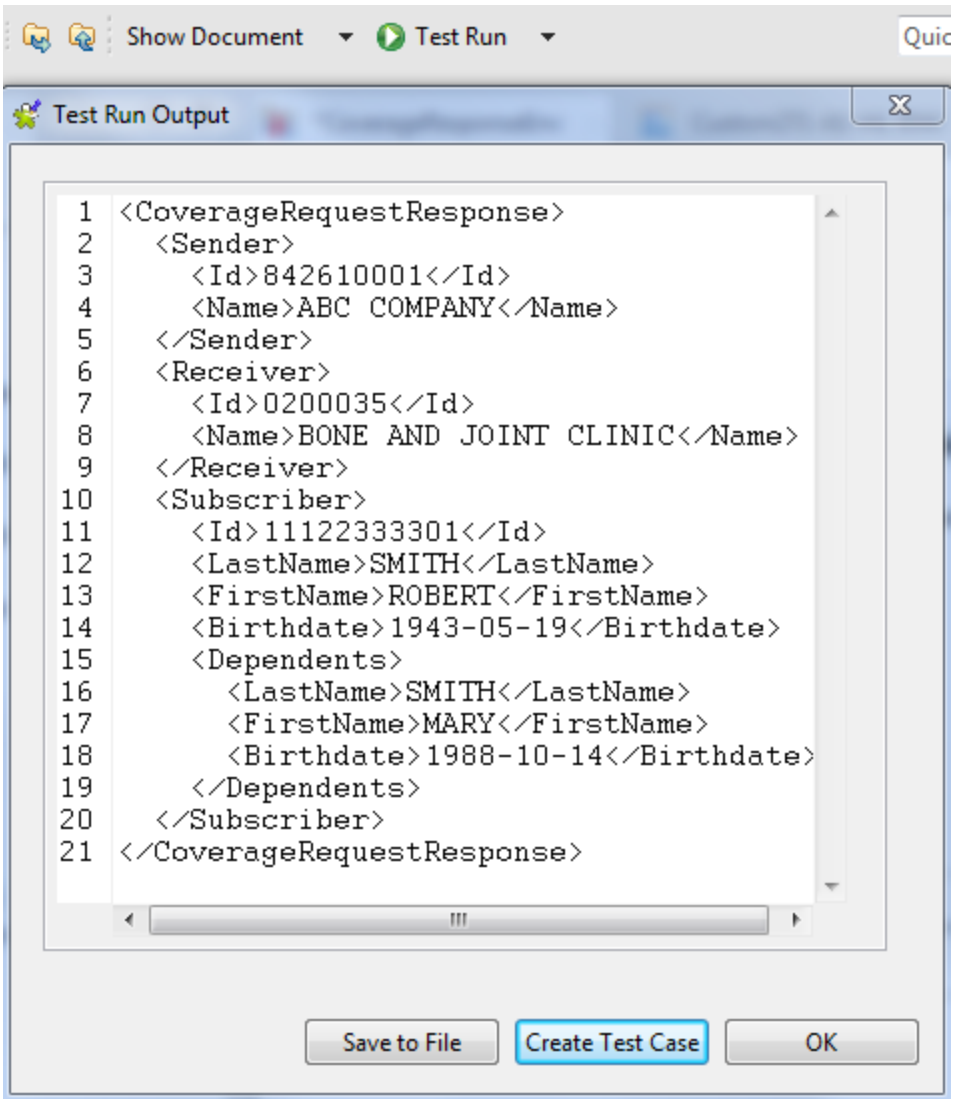
5. Notice the new element is mapped correctly. Also note that the **Move Expression References** operation was much faster than the copy/paste/delete sequence used on the output mapping.



6. Right-click **Transaction-271-A1** and select **Delete Invalid Map Elements** to remove all elements with wrong mappings.



7. Finally, click **Test Run** to check the output.



Next Step

This lesson is almost over. Head to the Wrap-Up section for a summary of the concepts reviewed in this lesson.

Wrap-Up

In this lesson, you learned how to:

- Change the inheritance and architecture of an existing Structure
- Move the old mapping expressions to the new elements

Next Step

Congratulations, you successfully completed this lesson. Click the **Check your status with this unit** button below in order to save your progress. Then click **Completed. Let's continue >** on the next screen to jump to the next lesson.

LESSON 10

Defining and Running Test Cases

This chapter discusses the following.

Overview

Lesson Overview

You will create a test case for the existing *COBOL2XML* Map, based on the default output. Then you will introduce a bug intentionally, check that the test case no longer succeeds, fix the bug and check that everything is back on track.

Objectives

After completing this lesson, you will be able to:

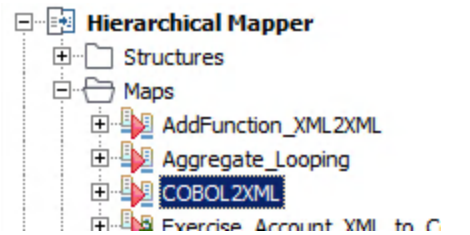
- Use the output of a given Mapping to define a reference test case
- Run test cases for a given node, to make sure no feature was broken by accident

Next Step

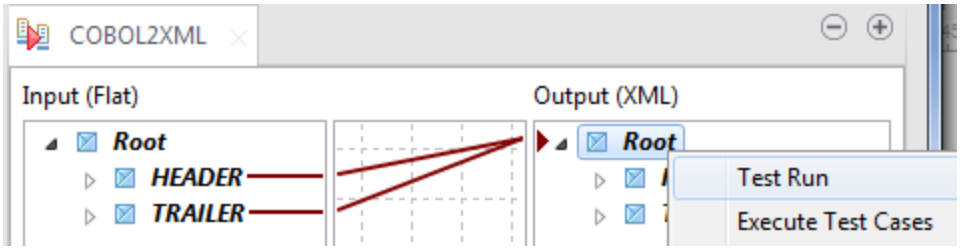
Let's use an existing Map and create a test case from a reference output.

Setting Up a Test Case

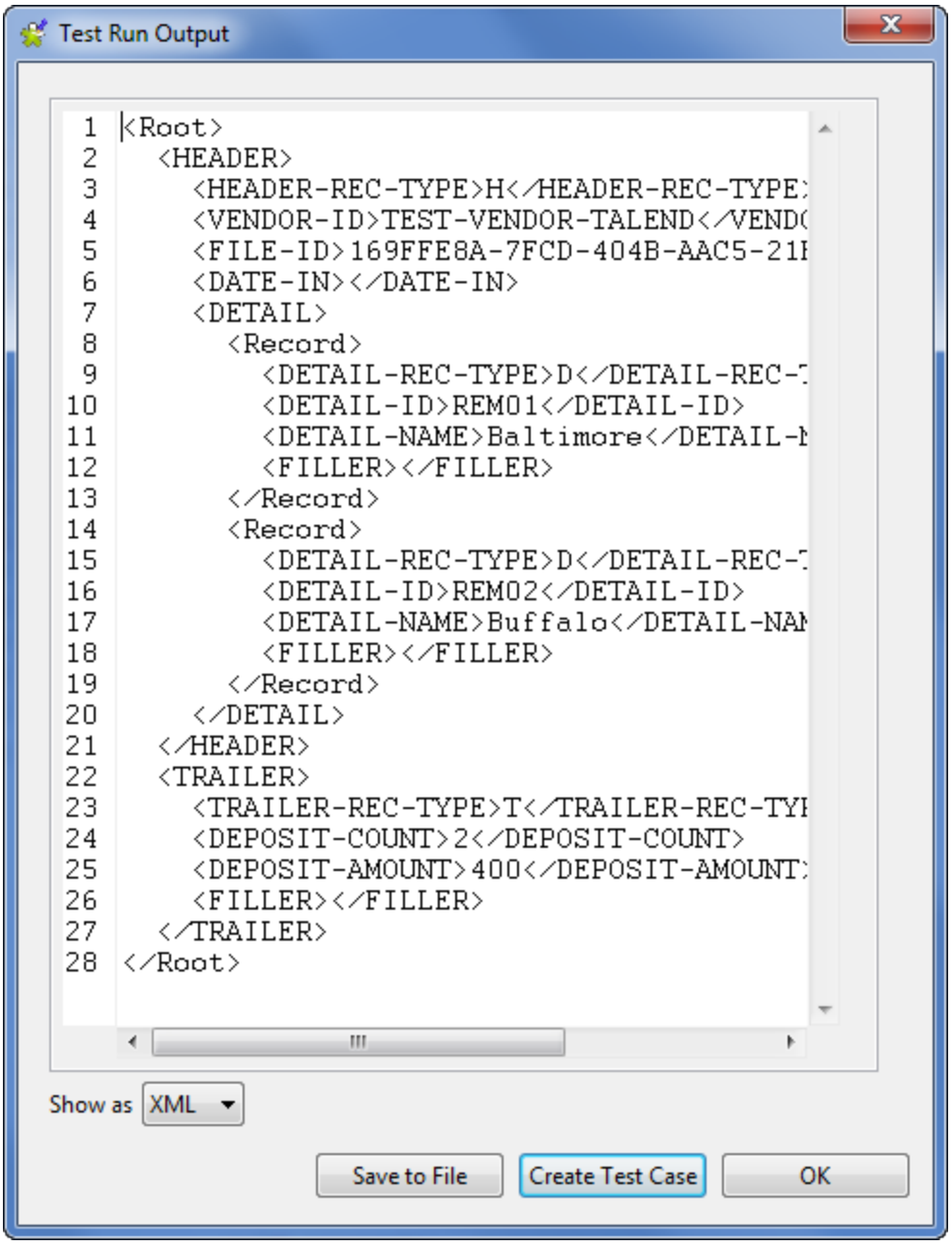
1. Double-click the **Hierarchical Mapper > Maps > COBOL2XML** Map to open it.



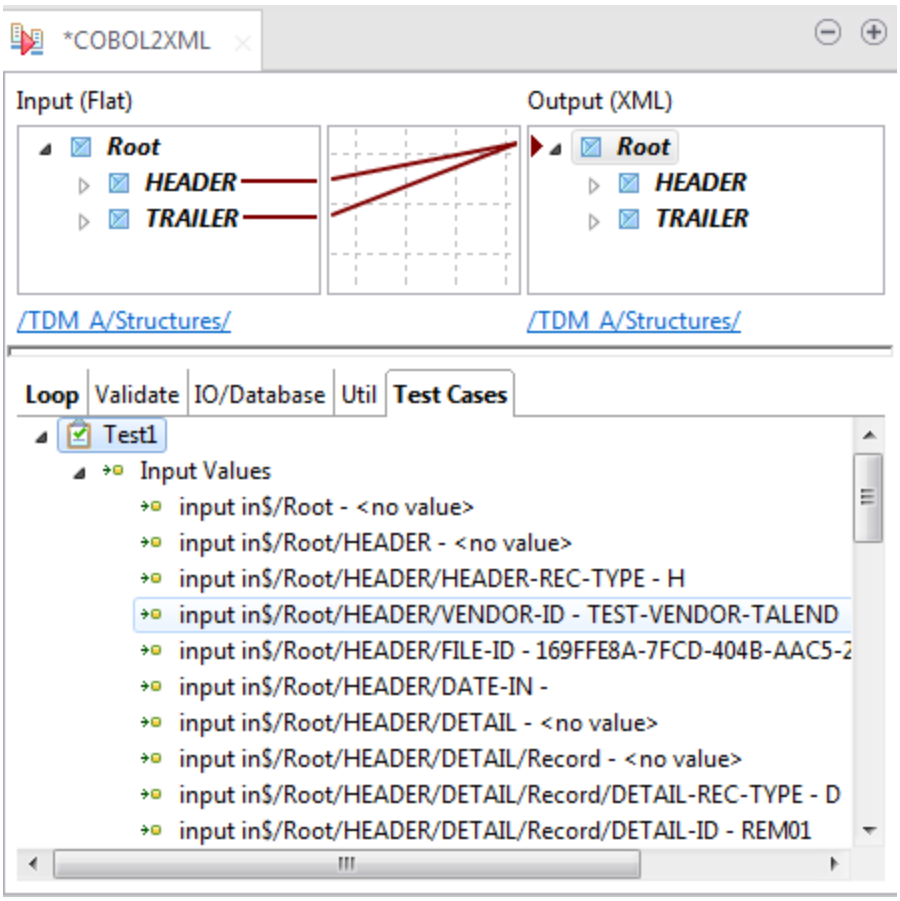
2. Right-click the **Root** node in the **Output** area and select **Test Run**.



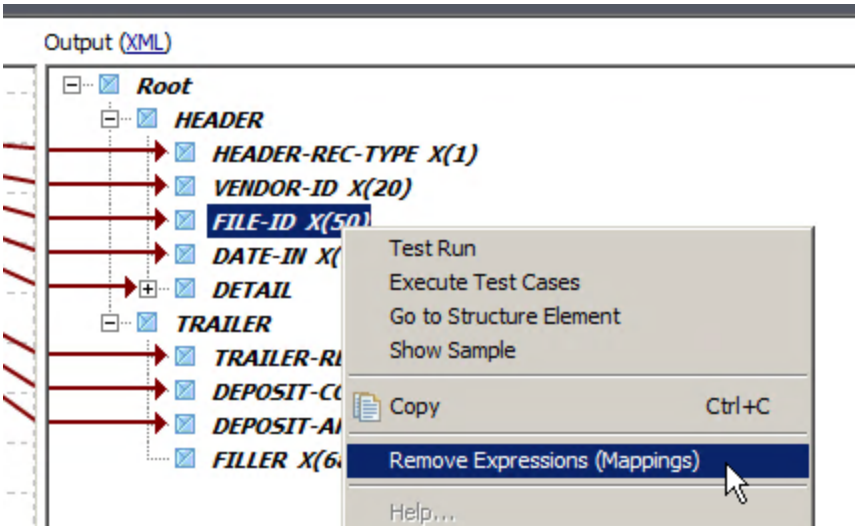
3. The usual output window is displayed. Now, instead of clicking **OK** to close it, click **Create Test Case** in order to use the current output as the output of a future unit test. Future outputs will be compared to this reference to check whether the transformations performed successfully or not.



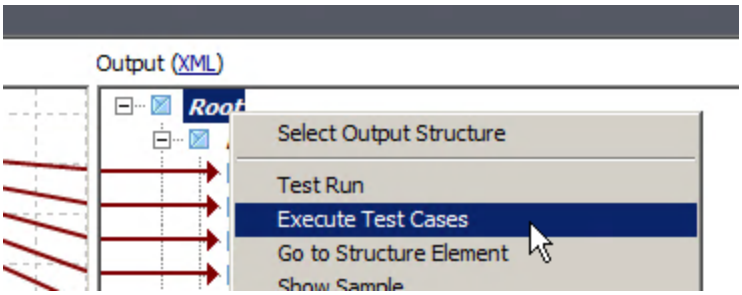
4. The **Test Cases** tab shows up and displays a list of all saved Test cases.



5. Right-click the **FILE-ID X(50)** element in the **Output** area and select **Remove Expressions (Mappings)**.

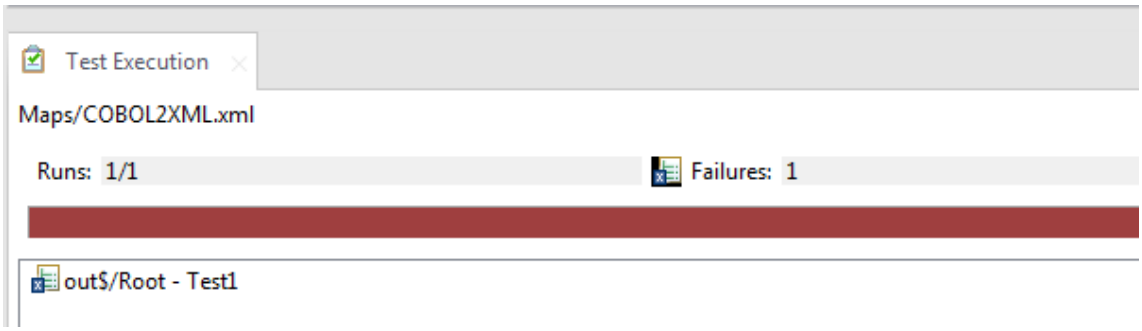


6. Right-click **Root** again and select **Execute Test Cases** to run all test cases registered with the Root node.

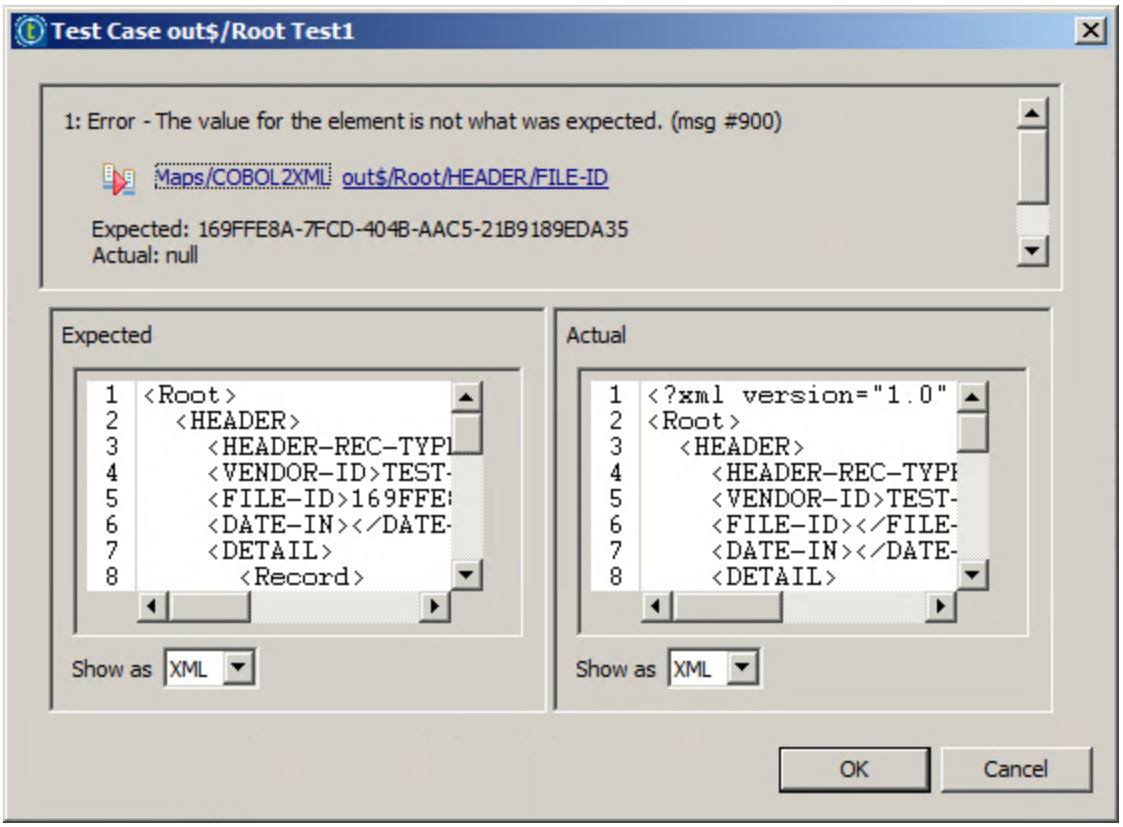


7. Examine the **Test Execution** tab at the bottom and note the progress bar is red, indicating a failure. It is confirmed by the **Failures: 1** message at the top right. It is the expected behavior, as we removed some of the mappings from the output. Some nodes must be missing compared to the reference output saved earlier in the exercise.

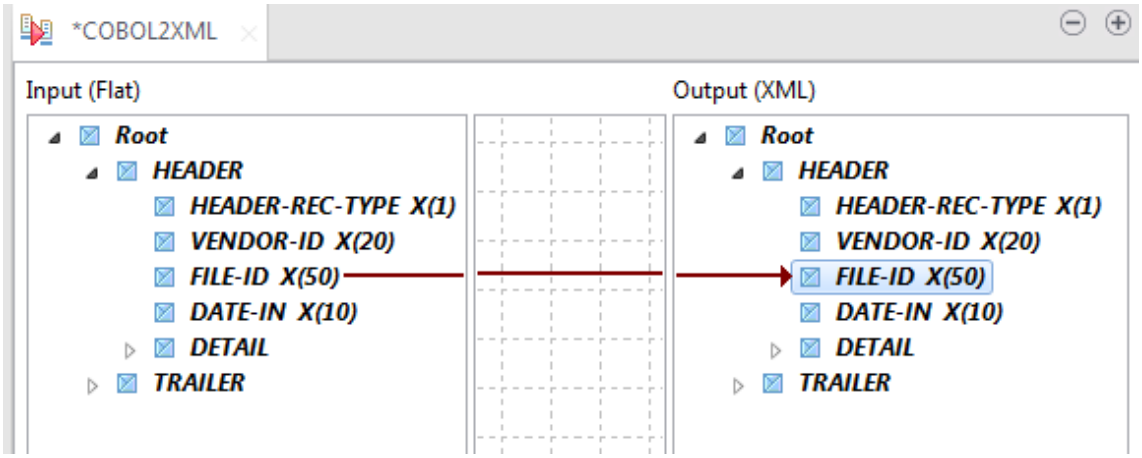
   Double-click **out$/Root - Test1** to get some details about the failure.



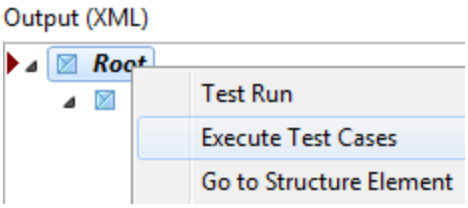
8. The **Expected** and **Actual** results are displayed so it is possible to check exactly where the error occurred. As mentioned in the error message at the top, the failure occurred in the *COBOL2XML* Map, in the *Root/HEADER/FILE-ID* output node. The reference output has a value associated to this node, while the current output has a null value (indicating the node does not exist). Let's fix this error and run the test cases again.



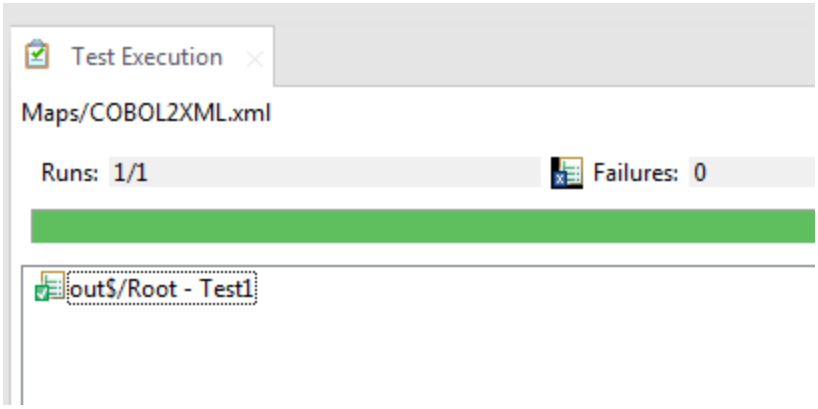
9. Map the input element **FILE-ID X(50)** to the output element **FILE-ID X(50)**.



10. Right-click **Root** again and select **Execute Test Cases** to run all test cases registered with the Root node.



11. Examine the **Test Execution** tab at the bottom and note the progress bar is now green, indicating the absence of failure. It is confirmed by the **Failures: 0** message at the top right.



Of course, it is possible to register several test cases for a given output. Having multiple unit tests covering different use cases buys peace of mind for the developer. It makes it possible to tinker with the application or fix bugs while maintaining the integrity of the application. For example, it is common for a bug fix to break another feature (unintentionally). Having tests covering every use case is very interesting from a quality engineering point of view. It makes sure a new feature or a bug fix did not break another feature.

Next Step

This lesson is almost over. Head to the Wrap-Up section for a summary of the concepts reviewed in this lesson.

Wrap-Up

In this lesson, you learned how to:

- Use the output of a given Mapping to define a reference test case
- Run test cases for a given node, to make sure no feature was broken by accident

Next Step

Congratulations, you successfully completed this lesson. Click the **Check your status with this unit** button below in order to save your progress. Then click **Completed. Let's continue >** on the next screen to jump to the next lesson.

This page intentionally left blank to ensure new chapters start on right (odd number) pages.