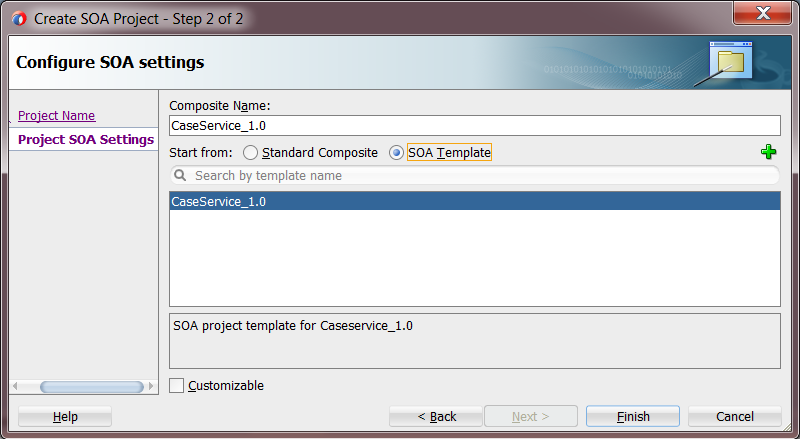
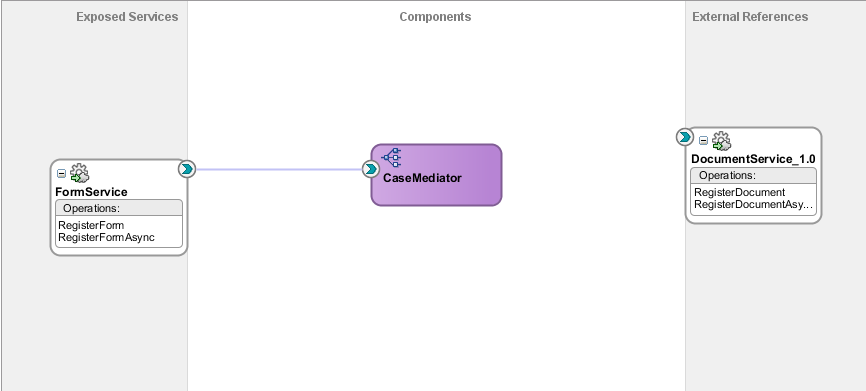
Storyboard X-masterclass 12c

# 1. Kick-start a SOA project using templates

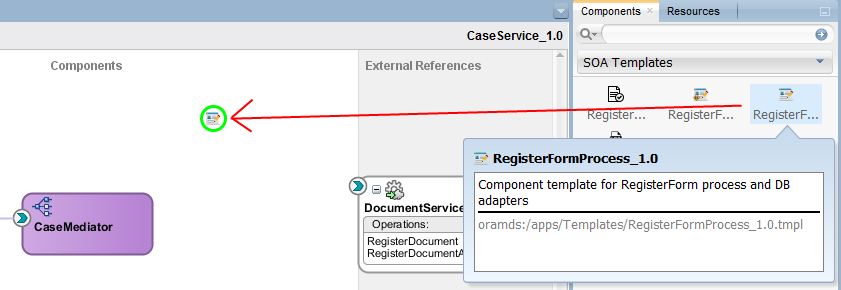
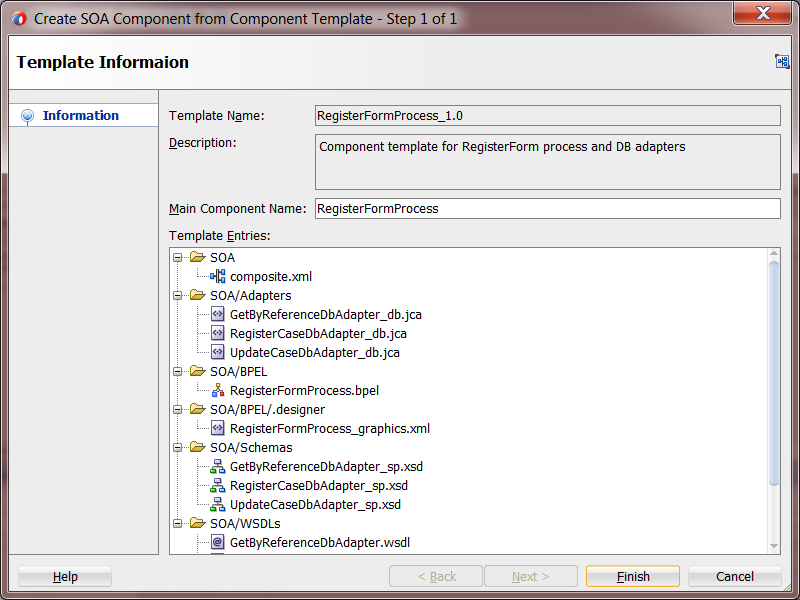
Create the CaseService composite using SOA templates (result is CaseServiceBasedOnTemplates).

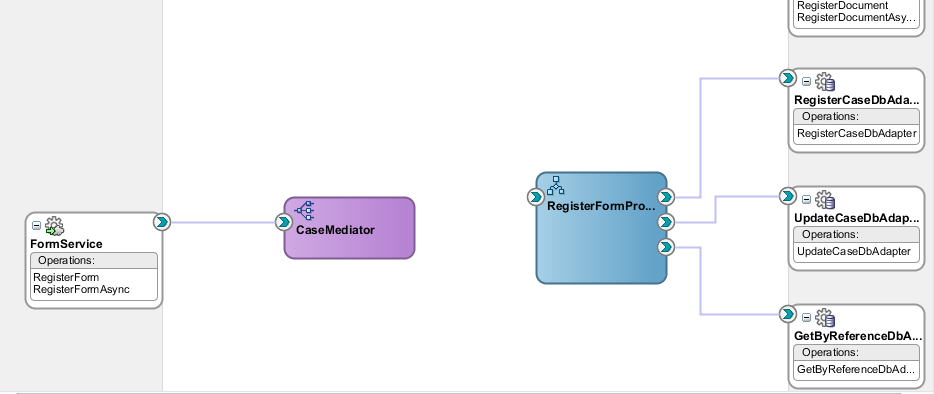
1. Create a new SOA project (CaseService\_1.0) using the corresponding project template. This template includes a SOAP Service Binding with WSDL from the MDS, a Mediator, Schematron validations and SOAP Reference Binding to DocumentService.



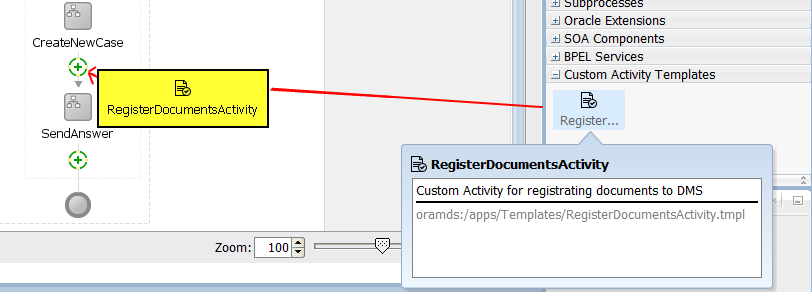


1. Create a new BPEL process (RegisterFormProcess) using the component corresponding template. Templates can found in the Component palette, under SOA templates, and can be dragged and dropped onto the composite editor. This template includes a BPEL process skeleton with Adapters and Fault Handling for the initial registration/creation of new cases.

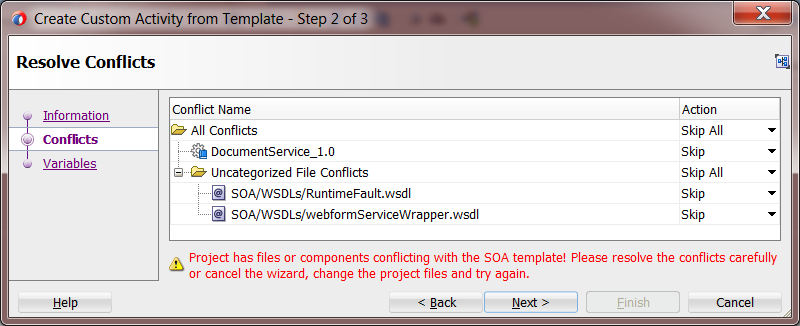
  




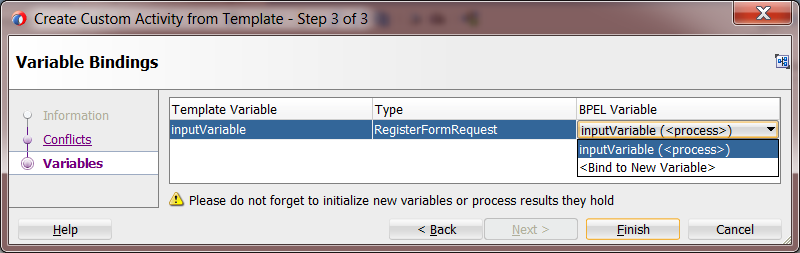
1. Create a new Scope (RegisterDocument) using custom activity template. Templates can found in the Component palette, under Custom Activity Templates, and can be dragged and dropped onto the BPEL editor. This template includes transformation(s), an Invoke of DocumentService and the needed Partner Link.

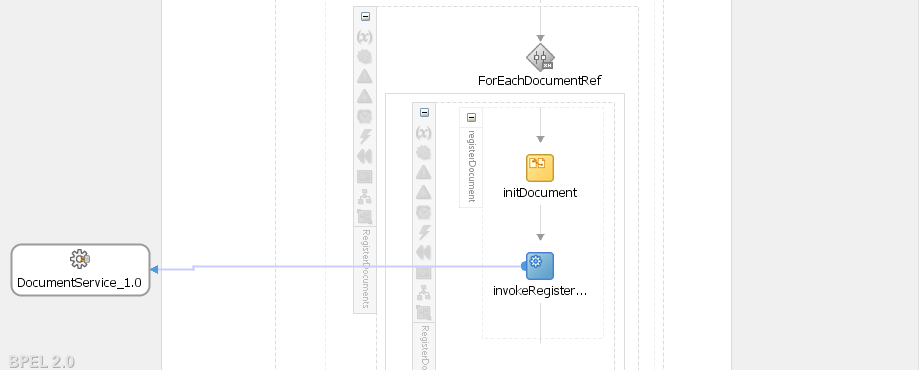


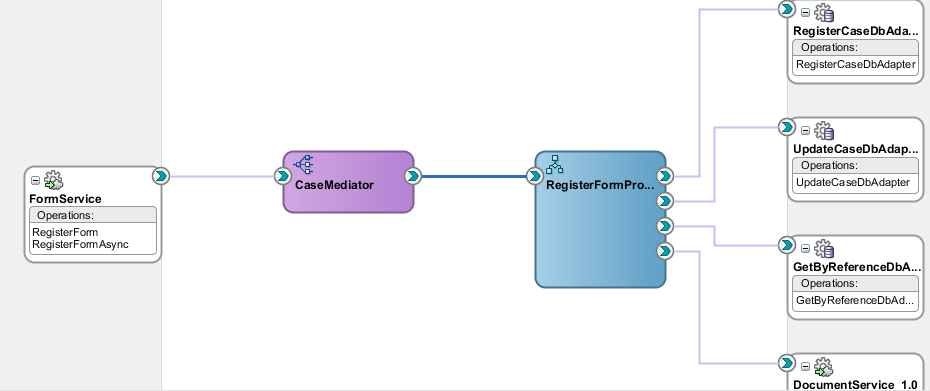
Skip or override conflicts were needed, but for now skip all conflicts.



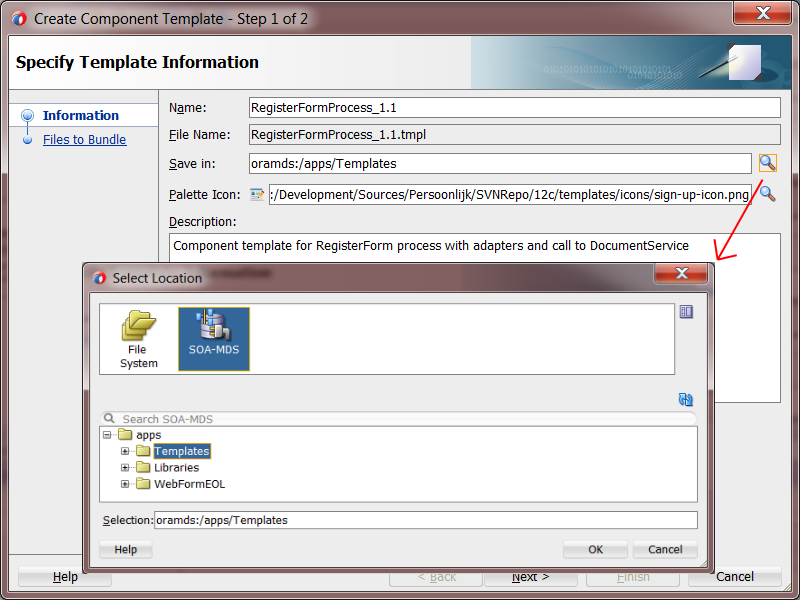
Assign the existing inputVariable to use in the transformations. You can also bind a new Variable, but then you need to initialize it yourself.



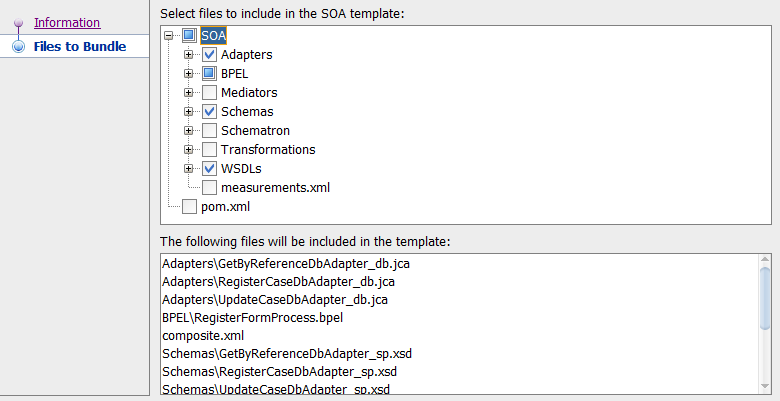




1. Create a new Component template based on the current RegisterForm. Right-click on the BPEL process in the composite overview and choose the Create Component Template option.



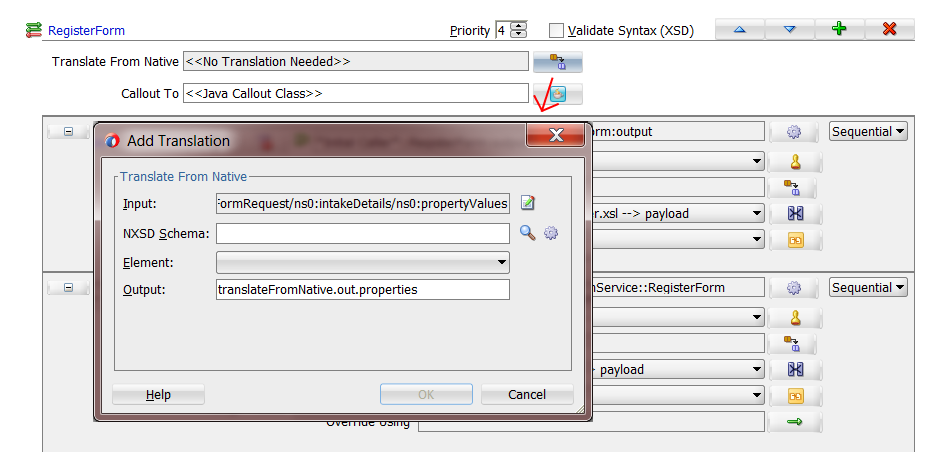
Choose which file to include in template. Depending files can’t be excluded / deselected.



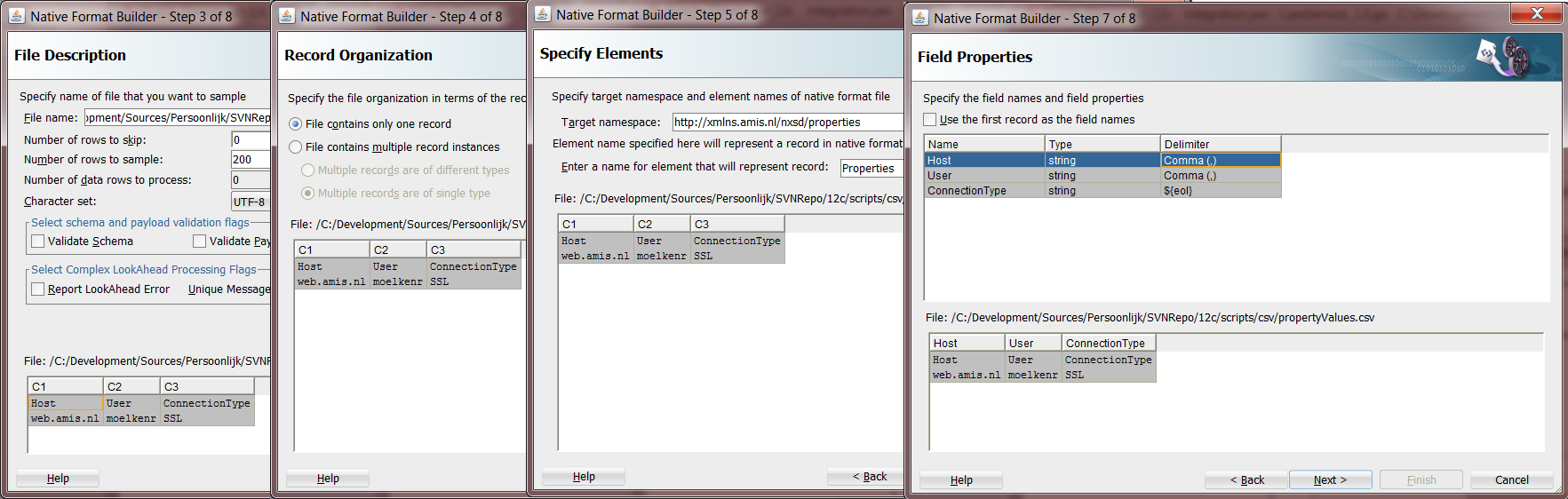
# 2. Updated Mediator Component

In the mediator an extra action Translate from Native is added. With this action you can transform a CSV/JSON/String element of the request to XML using a **NXSD transformation**. The transformed output can then be used in the routing rules of the Mediator. This action can be done on **operation** level, but also on a specific **routing rule** for that operation (result is CaseServiceMediatorUpdates).

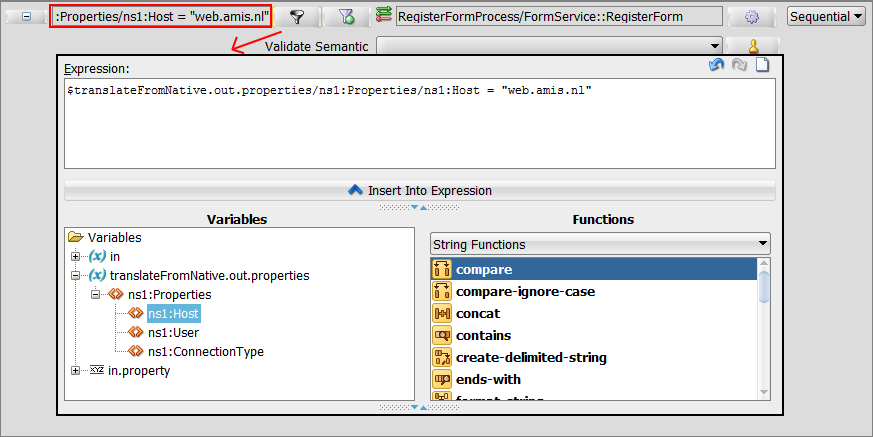
1. In the created project (CaseService\_1.0) open de Mediator (CaseMediator) component. For the RegisterForm operation create a new “Translate to Native” action.
2. Select the propertiyValues element from the input payload:  
   $in.payload/fmr:FormRequest/ns0:intakeDetails/ns0:propertyValues



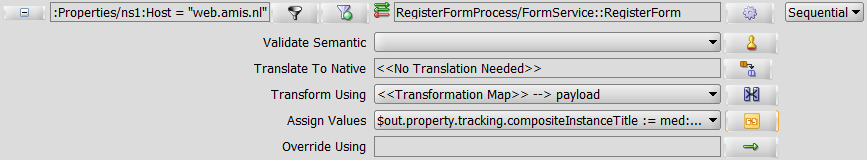
1. Create a new NXSD schema (propertyValues.xsd) based on the propertyValues.csv file (under scripts). De element consist of a **single-row** delimited values string, so it is not needed to skip any header rows.

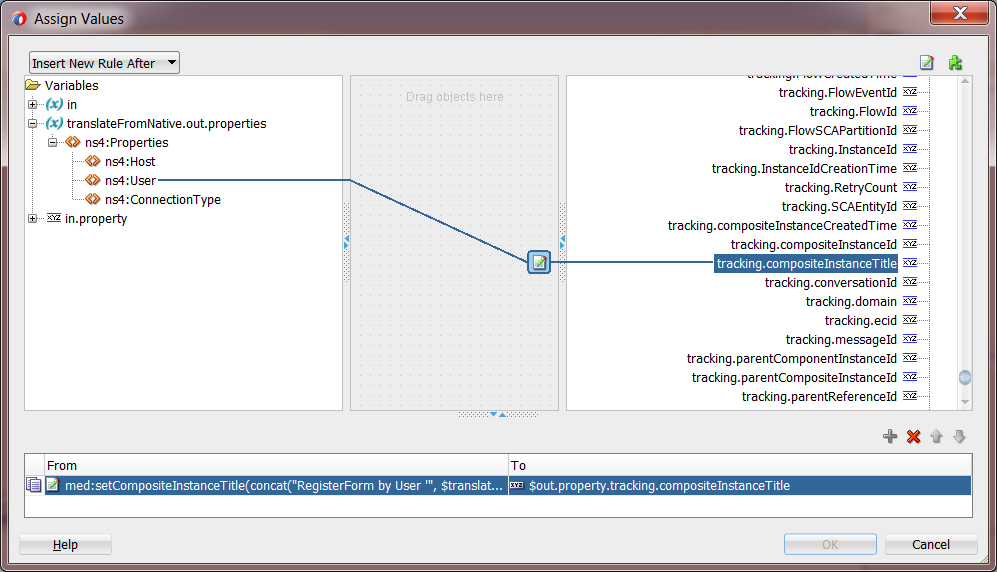


1. The newly created translation and output variable can be used within routing rules, transformations and assigns. Add a filter to the routing rule to filter on the Host element.

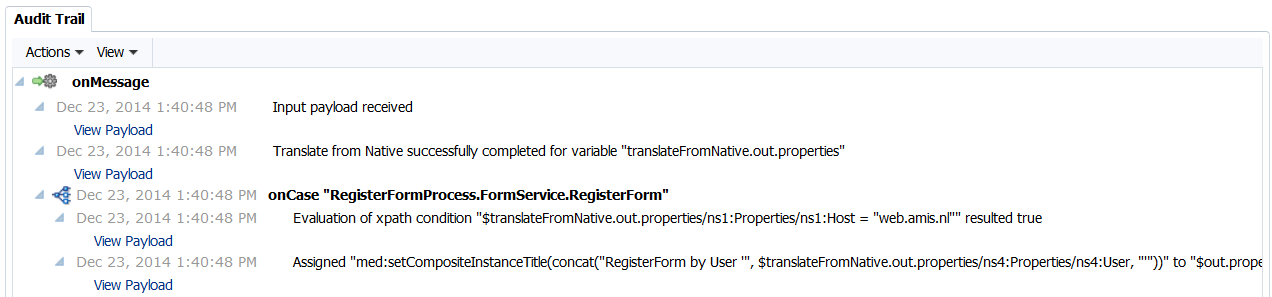


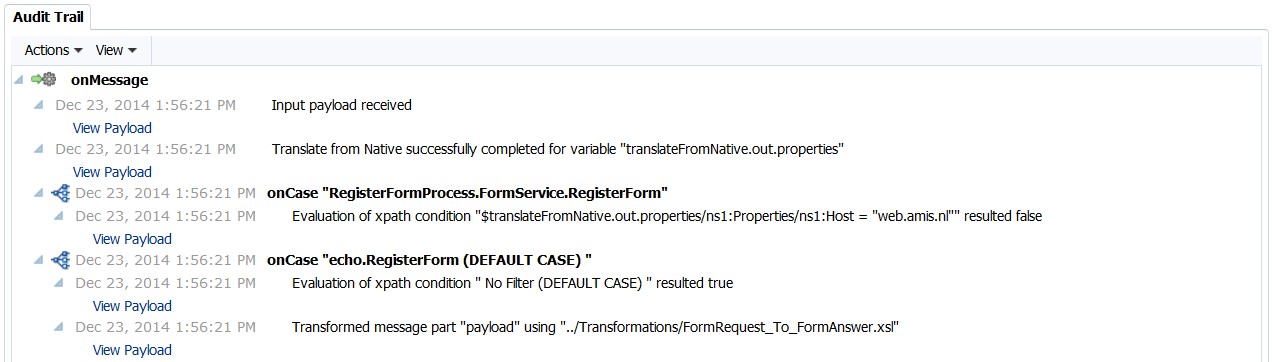
1. The assign of single elements has gotten a new editor much like the editor used in BPEL. Investigate this by adding a assign of the Composite Title.





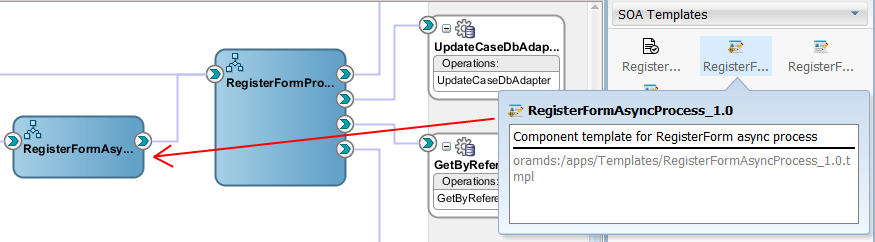
When a new instance is initiated the translation is executed. The values are used on runtime in the filter of the routing rule. If the value is correct the BPEL process is called else a reply is echoed.



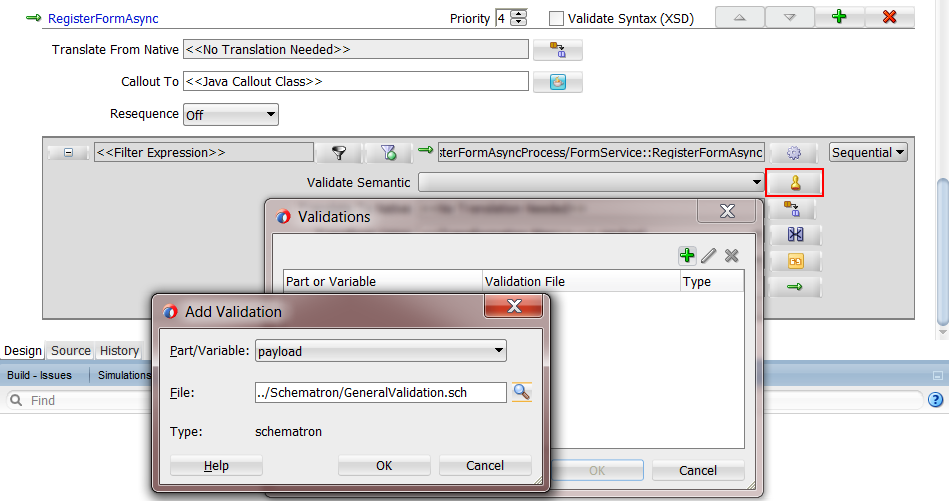


A second new action is Override Using. With this action you can override routing rule options using a DVM or Business Rules. Based on an element value of the request you can change the routing rule option dynamically. This action can only be added to async / one-way operations. With the key value you have the possibility to override the XSLT transformation file,  Filter expression, Execution Type, Syntax Validation (XSD), Semantic Validation (Schematron) file instead of creating a bunch of static routing rule.

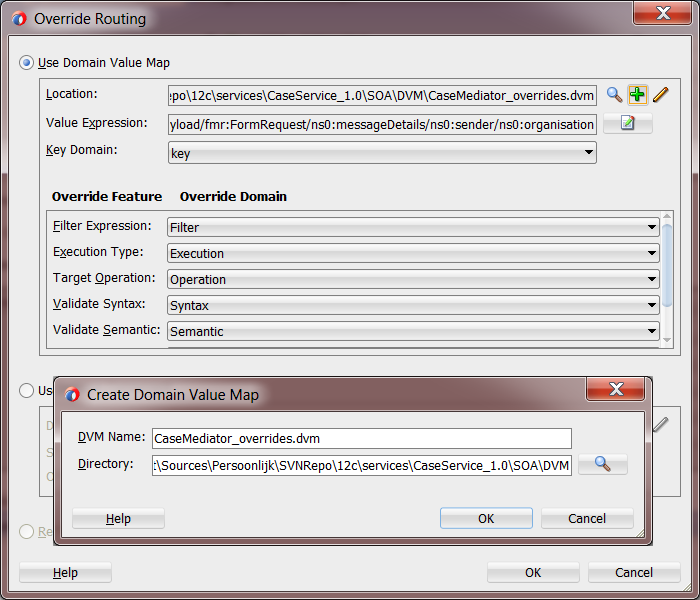
1. Prepare the composite by dragging and dropping the RegisterFormAsyncProcess component template and wire it to the CaseMediator and RegisterFormProcess component.



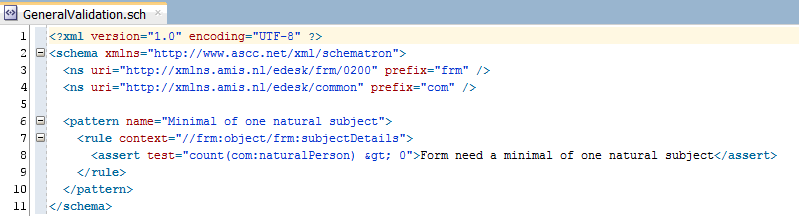
1. Add the GeneralValidation schematron schema to the routing rule. The schematron can be added under the Validate Schematic action.

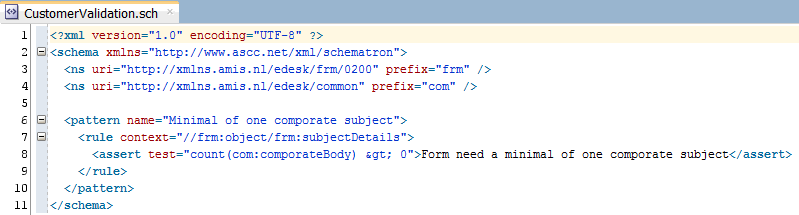


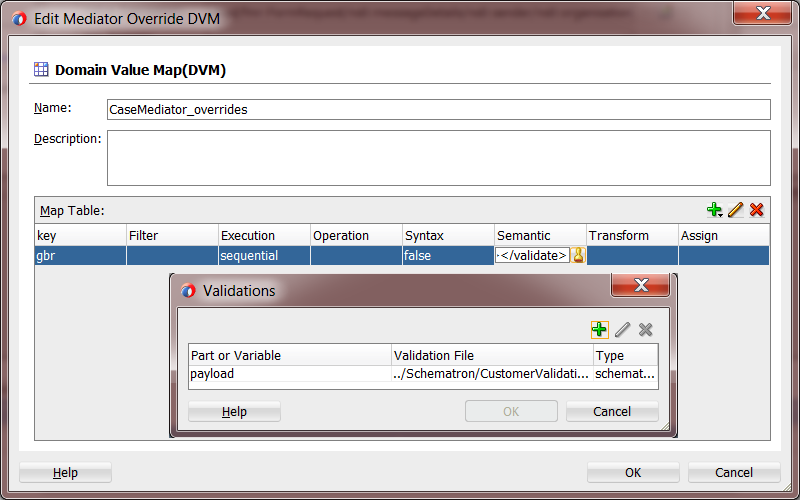
1. Create a new Override Using a DVM file and switch schematron validation for a specific sender/organisation value: xp20:lower-case ($in.payload/fmr:FormRequest/ns0:messageDetails/ns0:sender/ns0:organisation)

`

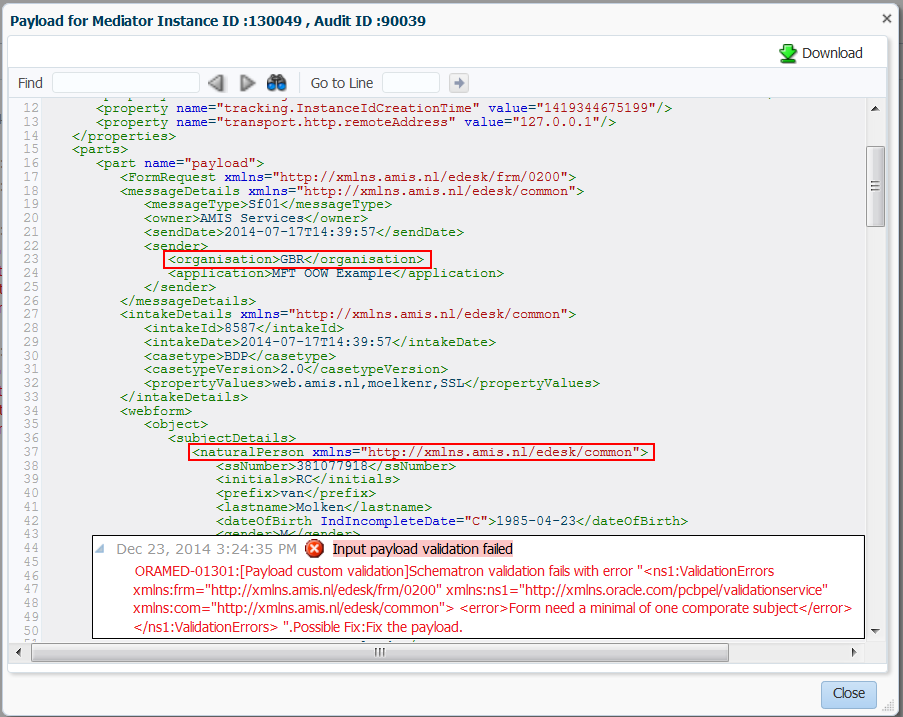
1. When editing the DVM you can change the values of the Mediator actions within a routing rule. For instance changing the schematron schema to use.  
     
   **GeneralValidation.sch** checks if the form has a minimum of one natural subject.  
   **CustomerValidation.sch** checks if the form has a minimum of one company subject.







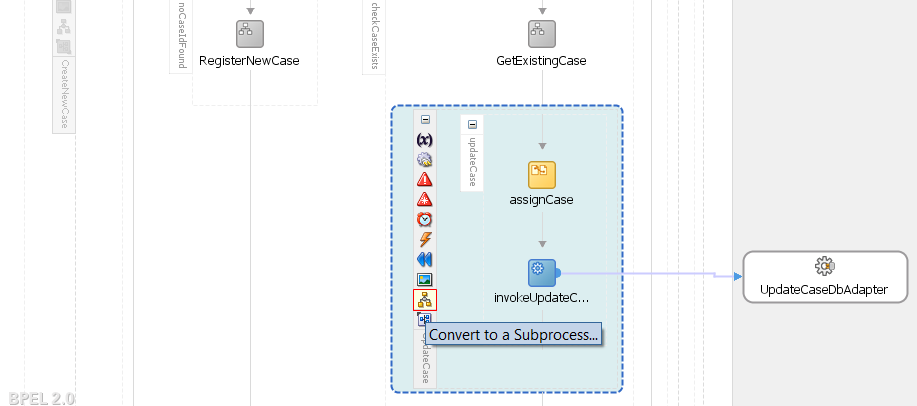
When a instance is initiated for sender AMIS the validation expects one natural subject because the GeneralValidation schema is used, but when the sender is GBR the validation uses a different schema which expects one company subject.



# 3. Re-use of BPEL code using Sub-processes

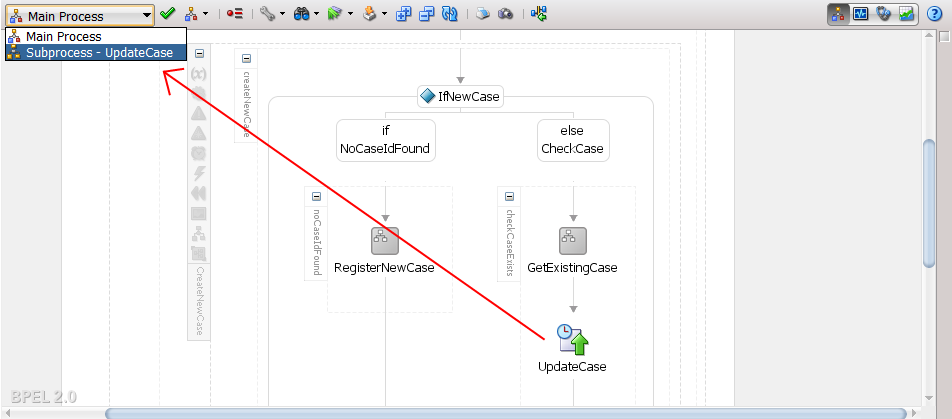
Re-use of code can be managed using BPEL Sub-processes, investigate this new feature by creating a inline sub-process from an existing scope in RegisterFormProcess (result is CaseServiceSubProcess).

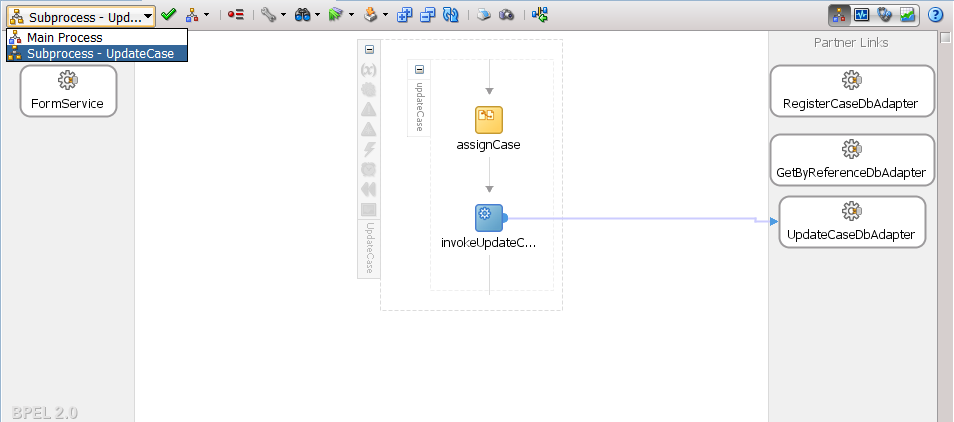
1. Open the RegisterFormProcess BPEL component and search for the UpdateCase scope under the CreateNewCase scope.
2. Right click on exiting scope, or click on the icon to create a (inline) sub-process.



|  |  |
| --- | --- |
|  | You can give it a name, choose to replace it with the call activity and some extra information as the label for the call activity, comment about the sub-process and an image to display for the call activity. |

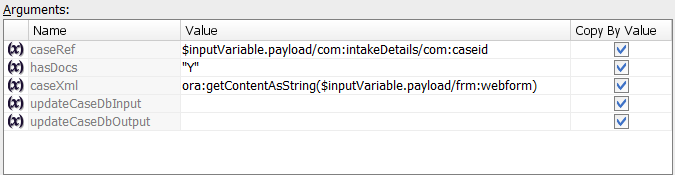
1. What happens with scope if it is converted to a sub-process?





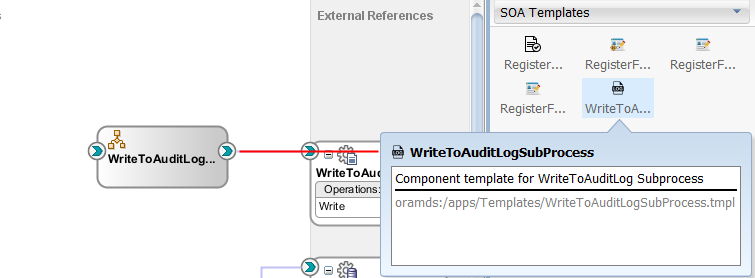
If you initialize variables with a default value, feature when creating variables, you should delete them so they can be overwritten using the call activity. You can re-initialize them without selecting a variable or delete the from tag from the source.

1. Assign values to the arguments of the UpdateCase call activity. To assign an expression first select the Copy By Value checkbox.

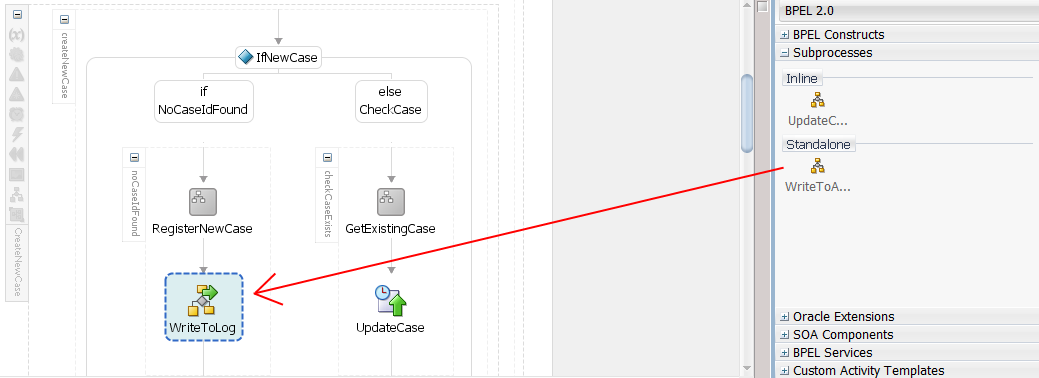


You can also create standalone subprocesses in the composite editor. Create a new Subporcess by dragging and dropping the Subprocess component onto the editor or using a template which includes the Subprocess.

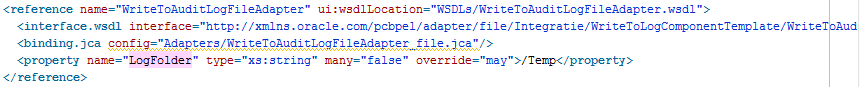
1. Prepare the composite by dragging and dropping the WriteToAuditLogSubprocess component template.



1. Open the RegisterFormProcess component and add a call activity to the standalone Subprocess. You can drag and drop the Subprocess from the Component palette which creates the call activity. Don’t forget to assign the arguments with relevant input.



1. The subprocess uses a File adapter with a logical folder, so don’t forget to set the LogFolder property in the composite.xml.



The call to the Subprocess is included in the audit logging.



# 4. New Editors in JDeveloper

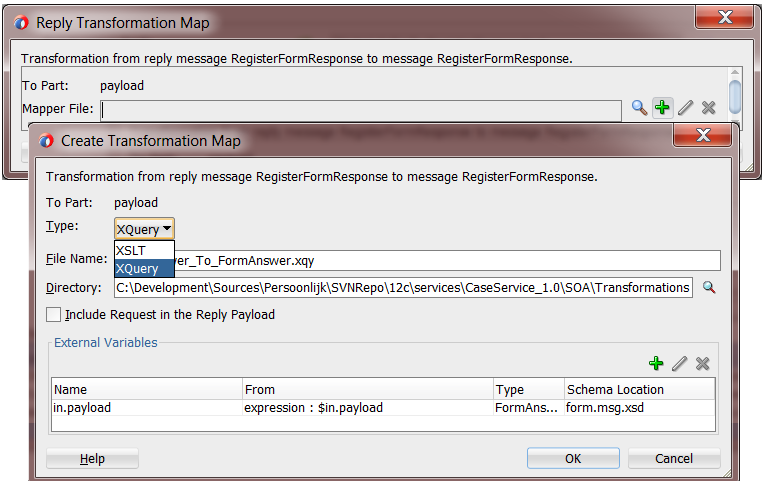
JDeveloper 12c has some revamped editors e.g. for XSLT, but also some new editors for XQuery and XSLT.

## 4.1. XQuery Editor

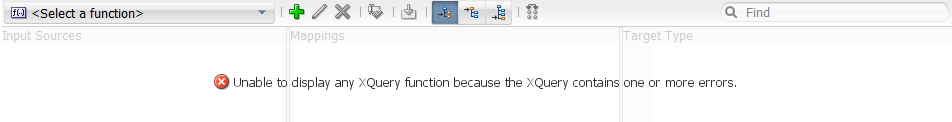
In JDeveloper 12c besides XSLT, XQuery can be used as a transformation language. In 12c there is a new editor for creating and editing XQueries. This editor is used in ServicesBus, BPEL and the Mediator.

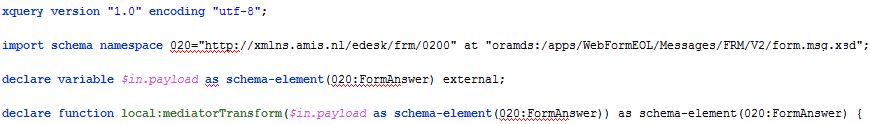
To investigate the basics of the editor open the mediator (CaseMediator) component of the SOA project (CaseService\_1.0).

1. Create a new XQuery transformation for the synchronous reply on the RegisterForm operations.

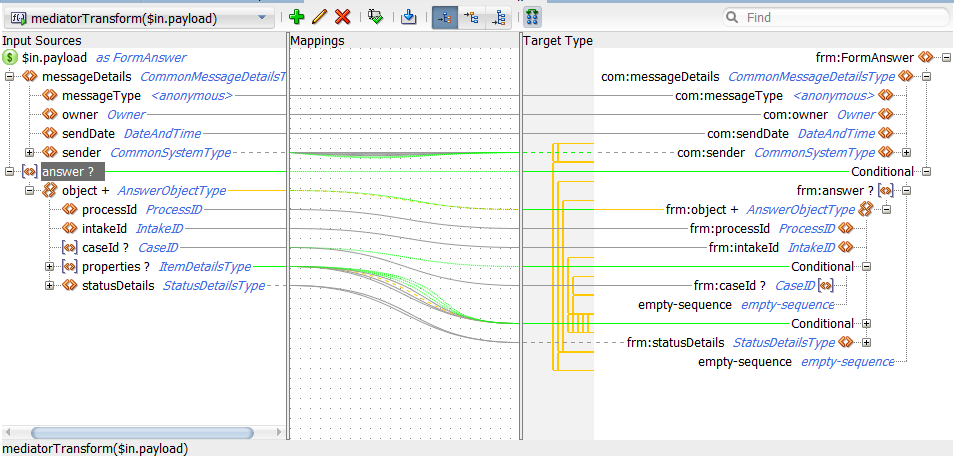


**Known bug**: When the XSD has a namespace where the last part are numbers. The prefix which the editor creates will also start with a number which result in an error, because it is not a valid XML name.





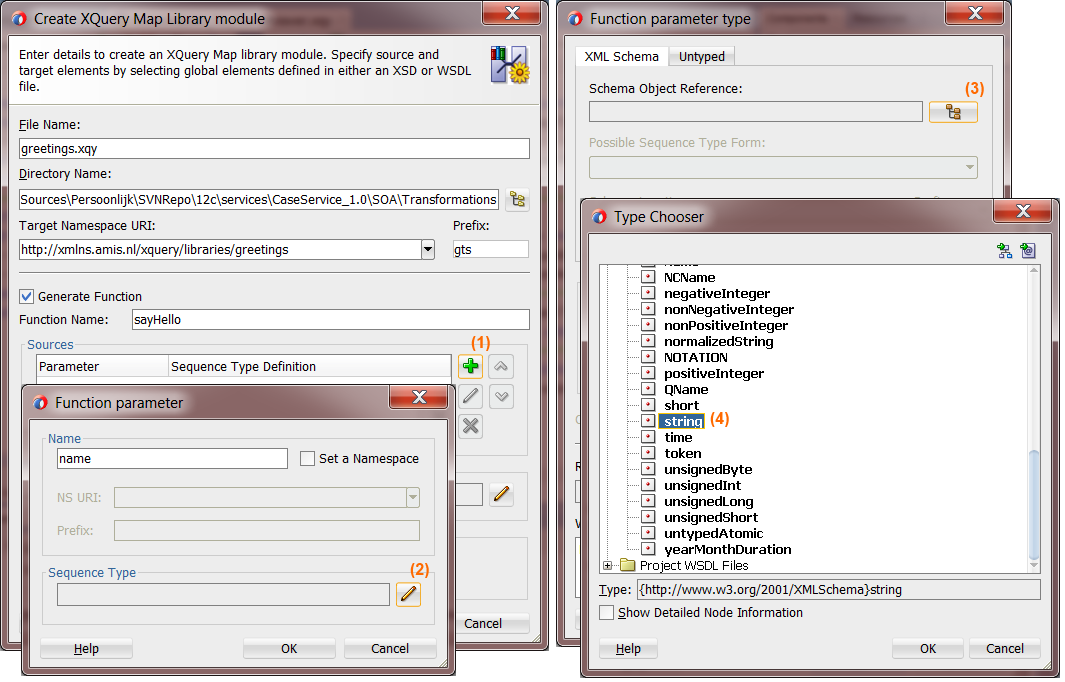
1. Go to the XQuery source and fix the error by renaming the prefix **020** to **fmr** without changing the namespace itself.
2. Go to the XQuery mapper and map the messageDetails and answer nodes exactly (source to target).

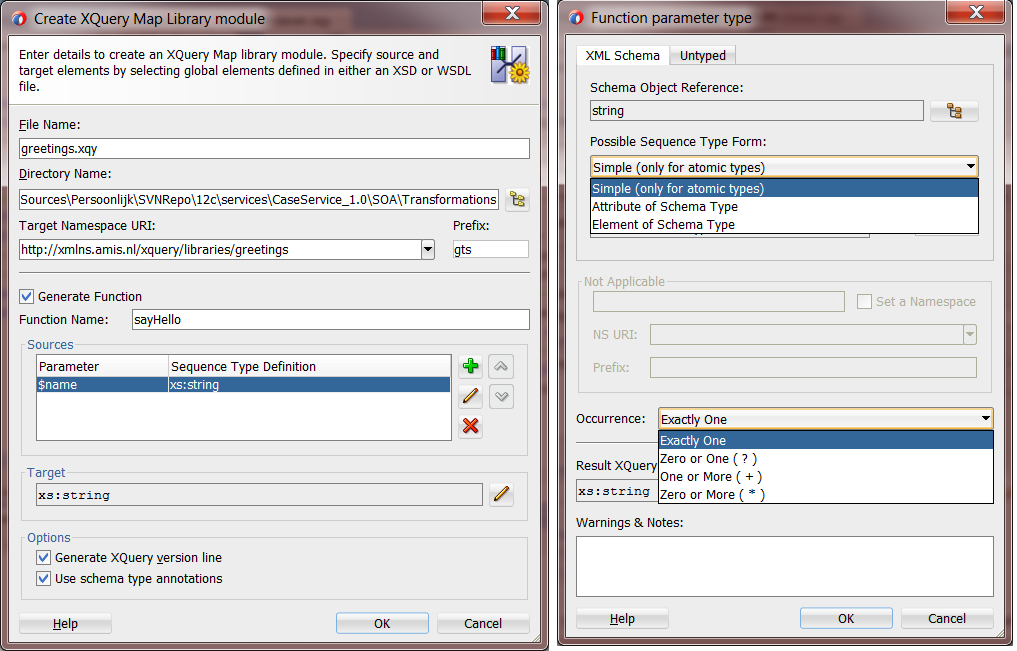


The black lines are used for required elements, green lines are used for (conditional) expressions and yellow brackets shows which elements are created within the expression.

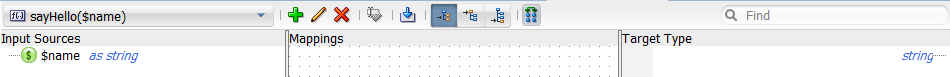
With the icons on top (from left to right) you can create new functions in the XQuery, rename, delete the current function, set the current functions as default and import XQuery libraries. The other icons are for the way the rules should be created (map, overwrite, append).

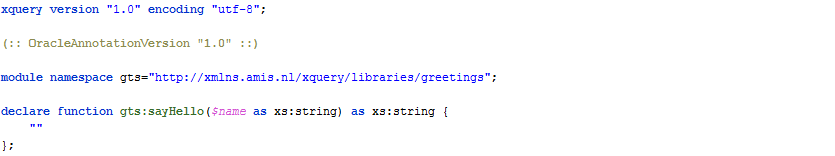
1. Before importing a XQuery library we are going to create one. Right-click on the project and select **new -> XQuery Library ver 1.0**. Call the library **greetings** and give it the namespace **http://xmlns.amis.nl/xquery/libraries/greetings** (prefix: **gts**). Create a function sayHello which receives a single string (param: **name**) and returns a single string.



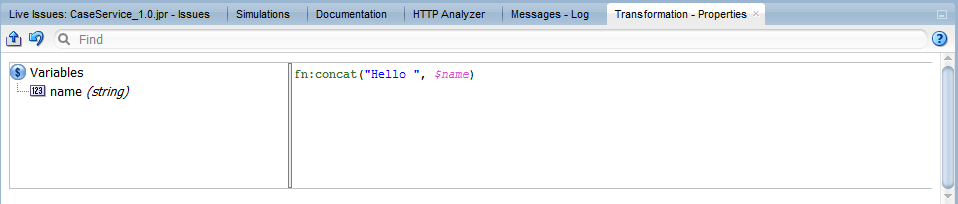


After clicking on the OK button it generates the XQuery for the library with the sayHello function.

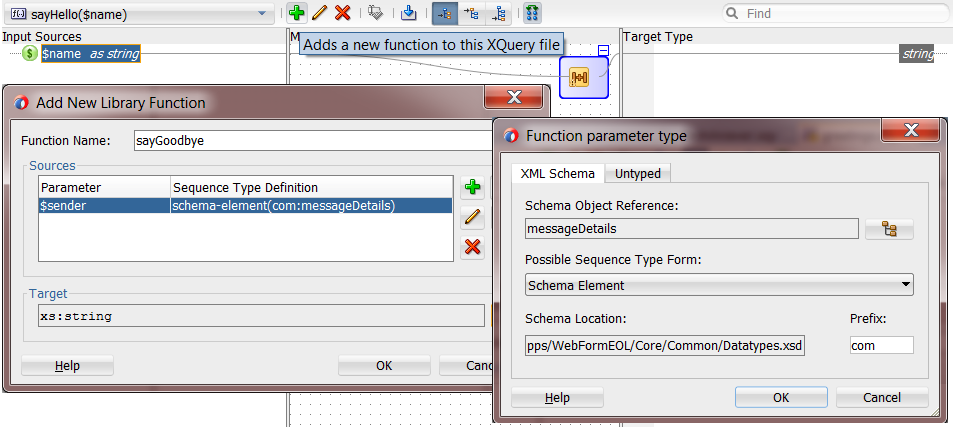




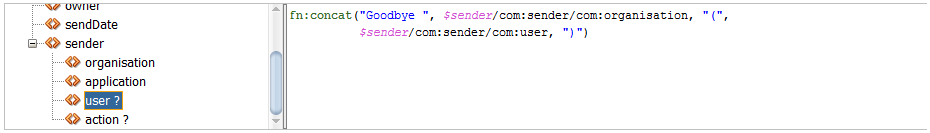
1. Add the concat function in the Mappings part and concatenate the string “Hello “ with the value of the input parameter.



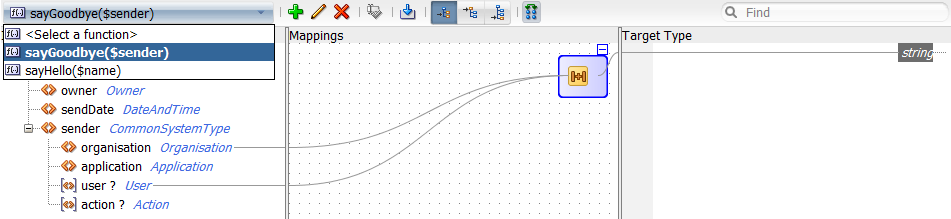
1. A library can contain multiple functions. Create a new function sayGoodbye with a single input based on the messageDetails element and a single string as output.



1. Use the concat function to concatenate the string “Goodbye “ with the senders organization and optional user to the target type.

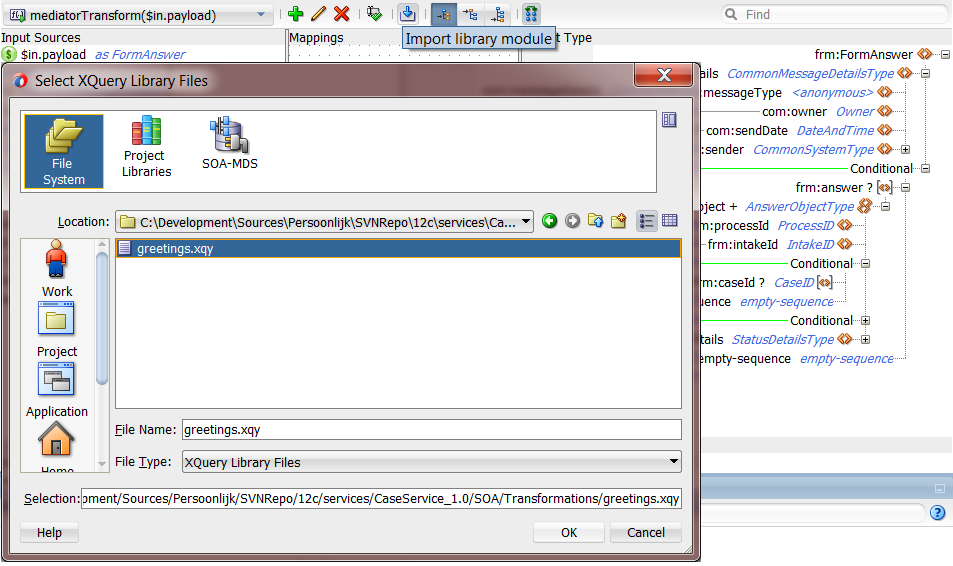


In the top left corner you can now switch between the existing functions of the library.



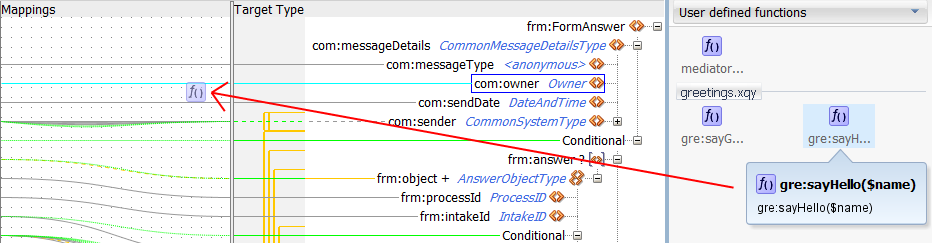
After creating the library it can be imported in other XQuery transformations.

1. Open the XQuery transformation created at the start and click on the import icon at the top of the editor. Select the XQuery library file **greetings.xqy** from the file system.

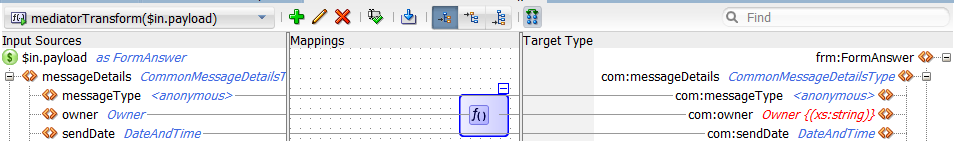


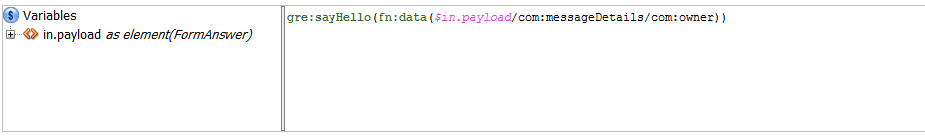
The functions of the library can be found in the Components palette under User Defined Functions.

1. Drag and drop the function sayHello onto the owner node of the Target Type. Preferable on the wire between the source and the target to automap the function.



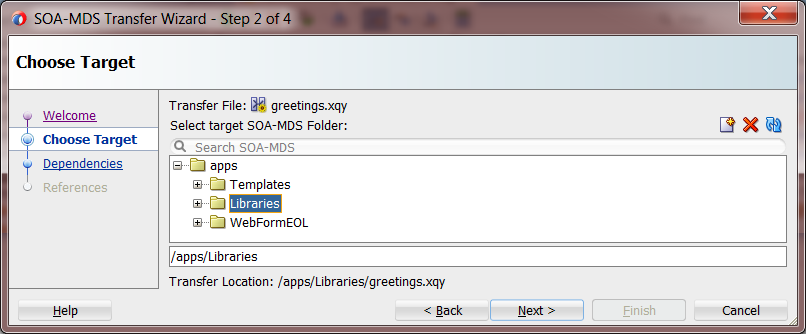
1. Select the function icon with the Mappings part of the editor to see the details in the Properties palette and/or to modify the expression.



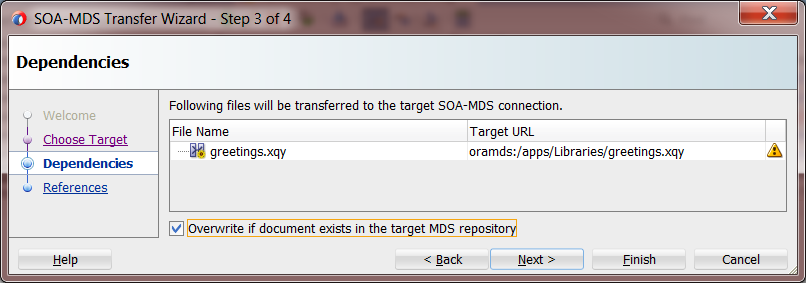


To re-use a library in multiple transformations the library can be placed in the MDS.

1. Right-click on the file, *Transformations/greetings.xqy*, in the project explorer and select to option to move it to the MDS.



Run throught the wizard and see that the reference to the file in the XQuery transformation is changed.



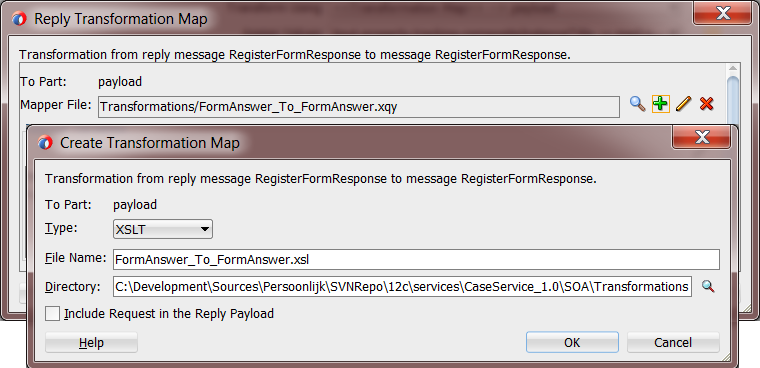


## 4.2. XSLT Editor

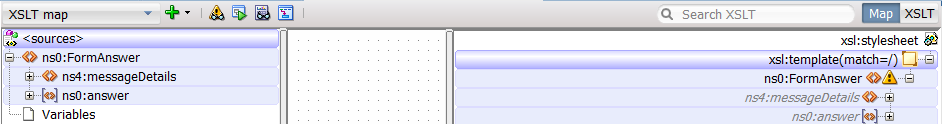
The XSLT editor is also revamped in JDeveloper 12c. It now supports multiple templates, import/include of other XSLT files and some nice features as the completion and execution view.

To investigate the basics of the editor open the mediator (CaseMediator) component of the SOA project.

1. Create a new XSLT transformation for the synchronous reply on the RegisterForm operations. Replace the existing transformation with a new one.

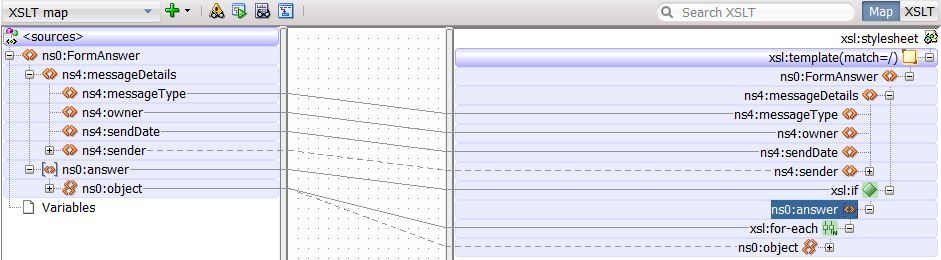


When the editor is opened the differences are immediately visible. At the top of the editor there are a few small icons with some big features.



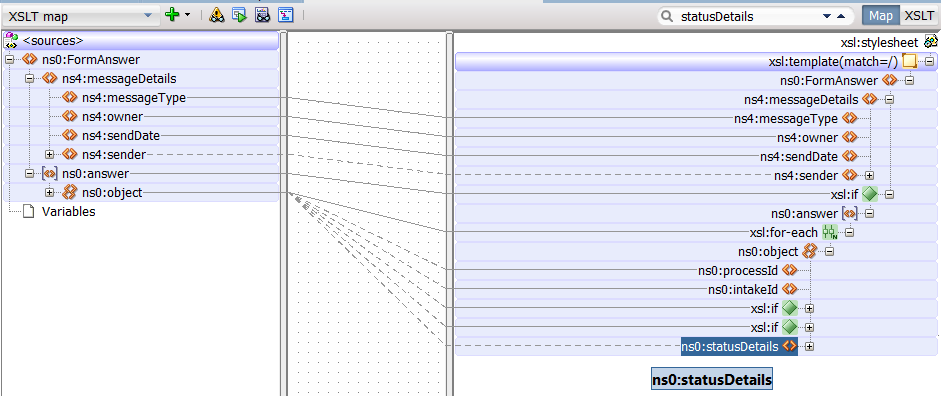
From right to left. The editors has to views to use the editor in. The Map view, which is default, is used for simple mappings without the use of multiple templates or import/include of other XSLT files. In the Map view you can create/use global variables (string values) and parameters (based on XSD schema). In the XSLT view you can create (named) templates, apply them to any node. Once you create multiple templates you can’t use the Map view anymore.

1. Go to the XSLT mapper and map the messageDetails and answer nodes exactly (source to target).



Left of the View modes there is a search box. With this box a search can be done to find a specific node in the created XSLT.

1. Search for the statusDetails node. When one or more nodes correspond to the search value (complete or partial) the tree is expanded and first node found is highlighted.

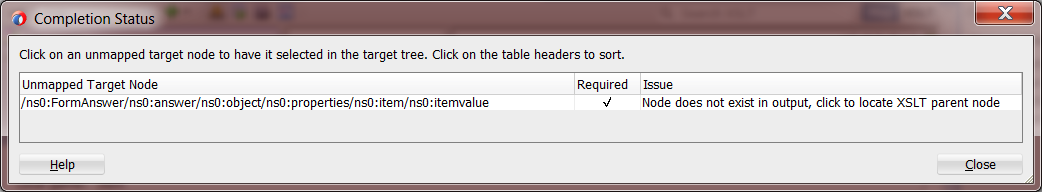


The first icon on the right is a new feature in 12c and opens the **Execution View** window. This window shows the way the XSLT is constructed. This view is only useful if the XSLT contains multiple templates, because than it shows in which order they are executed.

The two icons to the left of that are two functions that were already present in 11g; generate report and test XSLT.

The second icon on the left shows the **Completion Status View**. This is also new in 12c and shows which required elements are not (yet) present in the XSLT but should be.

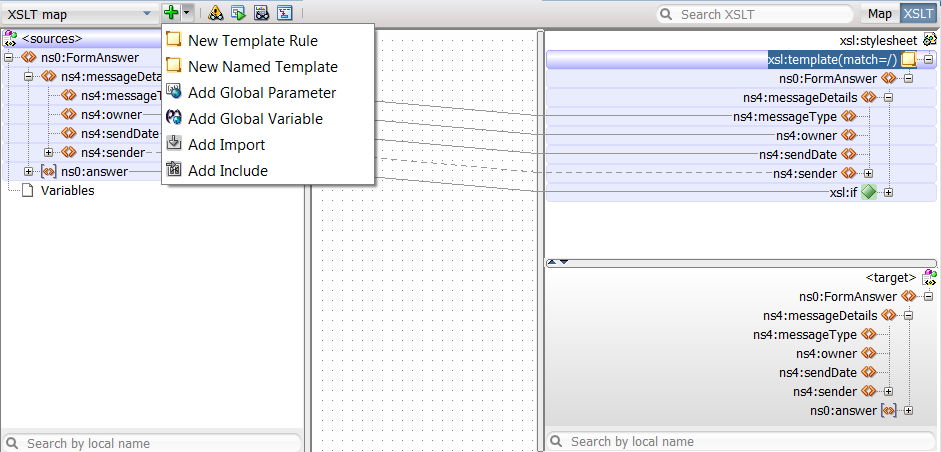
1. Open the Completion Status window and check it for unmapped nodes. When you double-click on a row it will jump to the node.



The element is mapped, but the editor gives the error because it had also has a mapping to a xsi:nil attribute.

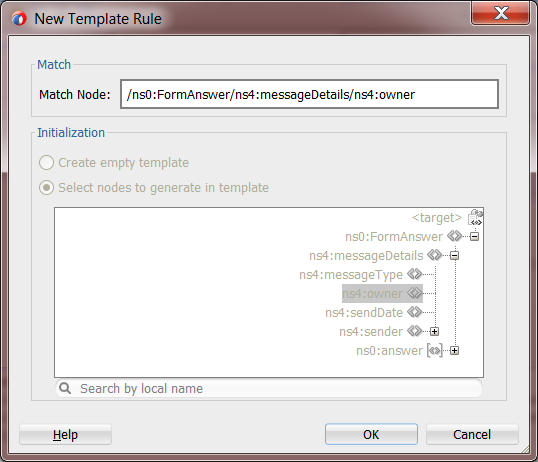
1. Fix the mapping so that it doesn’t have the mapping / expression on the **nil** attribute.

Switch to the XSLT view. In this view you can do much more than in the Map view. For instance creating multiple templates. With the first icon on top of the editor templates can be created and other XSLT files can be imported.



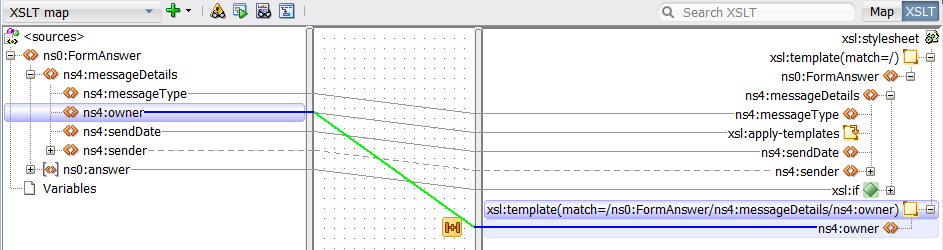
Templates can also be created directly on nodes in the XSLT stylesheet.

1. Right-click on the messageDetails/owner node and select the option “Create in Template”. Select the values in the “New Template Rule” window.

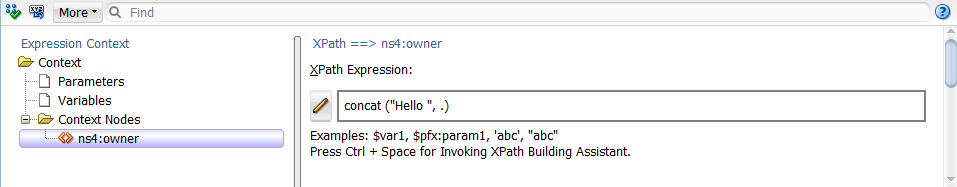


Use the concat function to concatenate the string “Hello “ with the value of the owner node (source).

1. Drag and drop the function preferable on the wire between the owner node of the source and the one from the template.



1. Select the function and edit the values of the function so that the value of the owner is concatenated with the string “Hello “.



## 4.3. Fault Policy Editor

Check for a detailed example the blog post on the AMIS technology blog: <http://technology.amis.nl/2014/07/08/soa-suite-12c-new-visual-editor-creating-fault-policies/>