Oracle® Retail Service Backbone

Developers Guide

14.0

E49441-01

December 2013

ORACLE®

Oracle® Retail Service Backbone Developers Guide, 14.0

E49441-01

Primary Author: Sanal Parameswaran

Contributing Author: Gloreen Soans

reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

This documentation is in preproduction status and is intended for demonstration and preliminary use only. It may not be specific to the hardware on which you are using the software. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to this documentation and will not be responsible for any loss, costs, or damages incurred due to the use of this documentation.

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your beta trial agreement only. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Software License and Service Agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

Contents

Send Us Your Comments

Oracle® Retail Service Backbone Developers Guide, 14.0.

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the Online Documentation available on the Oracle Technology Network Web site. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at http://www.oracle.com.

Preface

This Developers Guide describes the integration and flow requirements of the Retail Service Backbone Product.

Audience

This guide is for:

- Developers
- Integrators and implementers

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see the following documents in the Oracle Retail Service Backbone Release 14.0 documentation set:

- *Oracle Retail Service Backbone Integration Console Guide*
- *Oracle Retail Service Backbone Security Guide*
- *Oracle Retail Service Backbone Implementation Guide*
- *Oracle Retail Service Backbone Release Notes*

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to re-create
- Exact error message received
- Screen shots of each step you take

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 14.0) or a later patch release (for example, 14.0.1). If you are installing the base release and additional patch releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch releases can contain critical information related to the base release, as well as information about code changes since the base release.

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times not be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

This process will prevent delays in making critical corrections available to customers. For the customer, it means that before you begin installation, you must verify that you have the most recent version of the Oracle Retail documentation set. Oracle Retail documentation is available on the Oracle Technology Network at the following URL:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

An updated version of the applicable Oracle Retail document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-02 is an updated version of a document with part number E123456-01.

If a more recent version of a document is available, that version supersedes all previous versions.

Oracle Retail Documentation on the Oracle Technology Network

Documentation is packaged with each Oracle Retail product release. Oracle Retail product documentation is also available on the following Web site:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

(Data Model documents are not available through Oracle Technology Network. These documents are packaged with released code, or you can obtain them through My Oracle Support.)

Documentation should be available on this Web site within a month after a product release.

Conventions

The following text conventions are used in this document:

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

1

Getting Started with the RSB Developer Guide

This chapter provides an overview of types and styles of integration addressed by RSB and details of how to use this guide.

Introduction

RSB (Retail Service Backbone) is a web service based integration pattern implementation for Oracle Retail. RSB enables loose coupling between Oracle Retail and external applications and applications within Oracle Retail Suite. RSB is built on the top of Oracle Service Bus (OSB).

- RSB provides automated OSB configurations for web service deployment and security configurations
- RSB packages all of the Oracle Retail web services
- RSB provides tooling for the full life cycle management of OSB hosted Web Services (Development, Compilation, Deployment and Upgrades) and automatically adds instrumentation for runtime operations monitoring (using Retail Integration Console application)

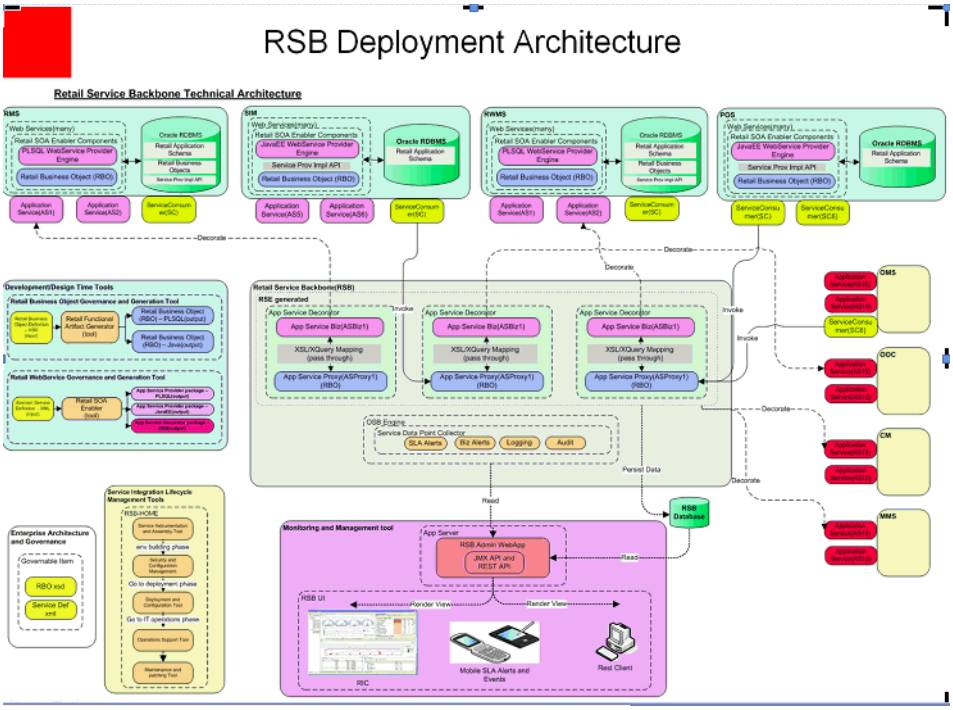
Developers often need to integrate third-party applications to Oracle Retail applications through RSB. This guide is intended to provide guidance on how to integrate third-party applications to RSB. This guide also provides insight to configure some of the RSB features to adapt to user requirements programmatically.

Types of Integrations Addressed by RSB

Oracle Retail uses three types of integration patterns:

- Request Reply
- Fire and Forget
- Bulk Data

Technical Architecture



2

Building RSB Integration Flows

This chapter introduces RSB integration flows and describes how to setup development and test environments.

Development Tools

The underlying infrastructure for RSB is built using OSB (Oracle Service Bus). Any RSB programming activity invariably involves OSB programming. The tools provided by OSB are the same tools used for RSB programming.

The primary recommended development tool for RSB programming is eclipse OEPE (Oracle Enterprise Pack for Eclipse). This is a preconfigured version of Eclipse with OEPE plugins.

OSB Console vs Eclipse OEPE

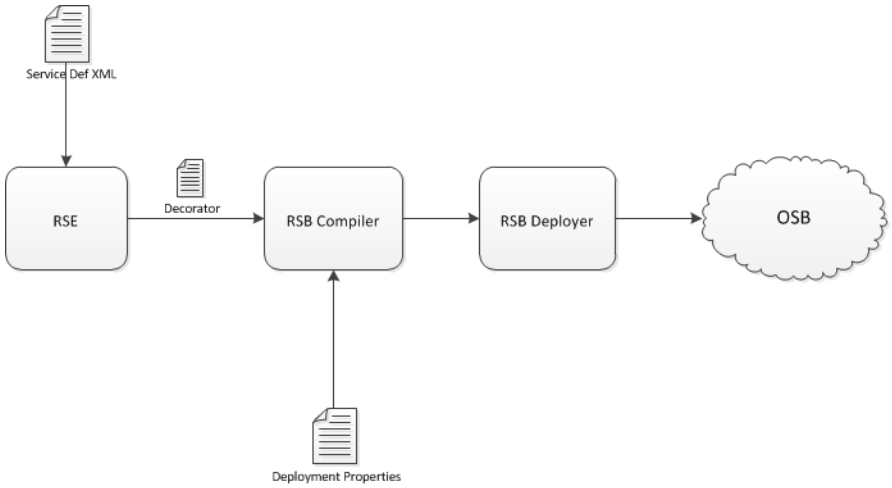
There are two ways to make programmatic changes to web services hosted in OSB server: OSB console and Eclipse OEPE. However, OSB console is an operational tool and is not recommended as a programming tool, even though OSB console provides the feature to make programmatic changes. The recommended approach is to use Eclipse OEPE for any programming changes to OSB/RSB components and use OSB console for operational changes to the OSB components. Also, it is important to note that when OSB projects/jars are re-deployed in OSB server, any operational changes made to the earlier version deployed in the server will be lost and these operational settings will need to be configured again in the new deployed projects.

Installing OEPE

Download and install OEPE from Oracle (http://www.oracle.com/technetwork/developer-tools/eclipse/downloads/index.html). After OEPE is installed, the pre-configured eclipse will be available in the oepe_<version> directory (for example, Oracle/Middleware/oepe_11.1.1.8.0). Use this eclipse for your OSB/RSB programming tasks.

OEPE does not need to be installed in the same machine where RSB builder tool is located. OEPE can be installed in a development environment, and the RSB jars can be copied to that machine. After making changes to the jars, they can be copied back to RSB builder and then deployed. The complete development lifecycle is explained later in this chapter.

Introduction to RSB Decorator jar Files



RSB provides Decorator PAKs for Oracle Retail applications. There is one PAK for each Oracle Retail application. Each PAK contains a set of jar files which are OSB deployable jars and are also known as decorators in the RSB context. Decorators are generated using Retail SOA Enabler (RSE) tool. The RSE tool uses the service definition XML file as input for generating the decorators. RSE generates one decorator for each service defined in the service definition XML. The decorator jar contains OSB artifacts related to that service. Each decorator jar contains a proxy service and a business service which are related to the service for which the decorator jar is generated. For more information about RSB builder tool and how it is used to compile and deploy the decorators, see the *Oracle Retail RSB Implementation Guide*.

The list of all application service decorator PAKs in this release is provided in Appendix A.

Note: For more information, see *Oracle Retail SOA Enabler (RSE) Guide*.

Introduction to RSB Service Integration Flow jar Files

RSB Functional Integration Flows are OSB integration services that are not decorators. Decorators have one proxy service and one business service related to the application service but service integration flows are not tied to a specific service. The purpose of service integration flows is to provide capabilities that range across multiple application services.

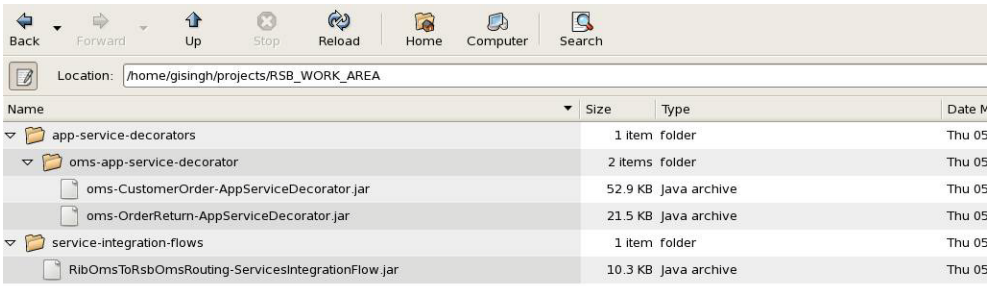
`RsbServiceIntegrationFlowPak14.0.0ForRibOmsToRsbOmsRouting_eng_ga.zip` is the only PAK available for this release. This PAK contains a proxy service which routes the data coming from RIB-OMS to various RSB decorator services.

How to Setup RSB Workbench

RSB workbench is a development area for integration developers who want to modify the existing RSB decorator projects for various purposes such as adding new functionality or integration with third-party applications.

Workbench area should be in the same machine where OEPE has been installed. In this document we will refer to that location as RSB_WORK_AREA.

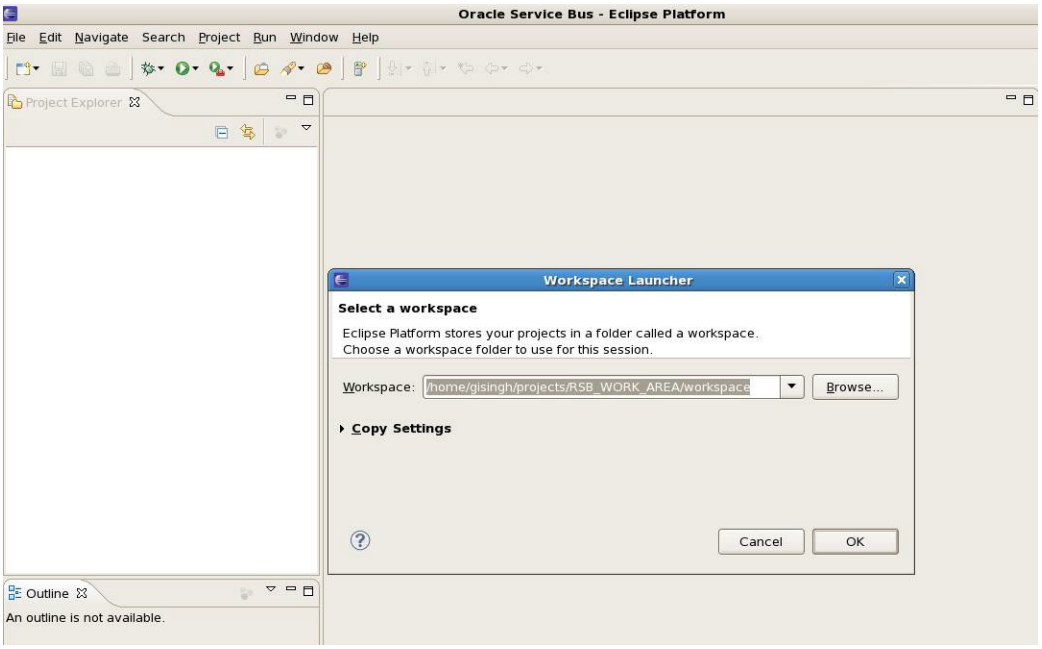
RSB has two types of OSB projects which can be customized: app service decorators and service integration flows. Therefore, it is recommended to have a directory for each type of project in RSB_WORK_AREA. The following is a screenshot of the recommended directory structure:



As shown above, the app-service-decorators folder has an application specific folder which contains the decorator jars for that application. Service integration flows are not application specific, therefore those types of jars can be directly within that folder.

OEPE Workspace

When an OSB jar is imported in OEPE, it extracts the jar inside the OEPE workspace. The extracted folder will have all the OSB project related files that were packaged in the jar. You can create the workspace at any location in the machine. For this document purpose, we will create a workspace folder inside RSB_WORK_AREA. We will refer to that location as OSB_WORK_SPACE in this document. Following is a screenshot of the workspace inside OEPE.



To summarize, RSB_WORK_AREA is the location where jars are copied to and from the RSB builder location and OSB_WORK_AREA is the location where jars are imported as OSB projects and are worked upon.

Development Lifecycle

When working on modifying RSB decorator jars or service integration flow jars, it should follow a certain lifecycle. This lifecycle should work in conjunction with RSB lifecycle. For details about all the lifecycle phases, see Oracle Retail RSB documentation.

For development lifecycle, the decorator jars or service integration flow jars must be copied from rsb-home/service-assembly-home folder.

Following are the steps that should be followed in the order mentioned below:

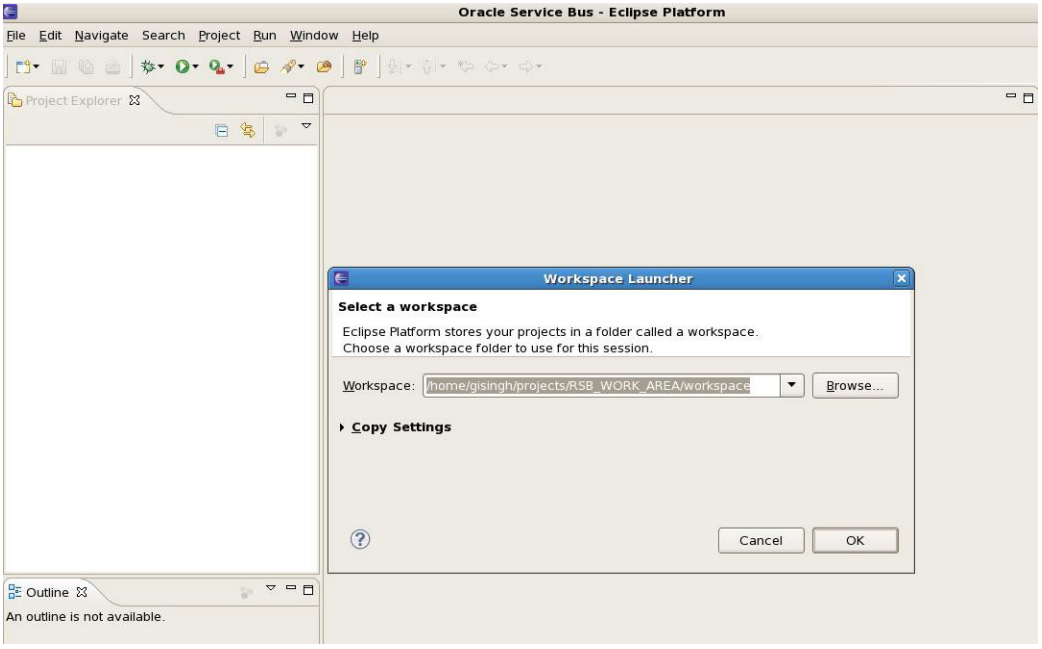
1. Copy the decorator jar or service integration flow jar that you want to modify from rsb-home/service-assembly-home to an appropriate location in RSB_WORK_AREA. The folder structure for RSB_WORK_AREA has been shown above in the screenshot.
2. Import the jar into OEPE where the OEPE workspace is OSB_WORK_SPACE. The steps for importing jar have been shown below.
3. OEPE will extract the jar and create an OSB project. The extracted jar will be saved in OSB_WORK_SPACE. The steps in the next section will show a screenshot of how the extracted jar looks like.
4. Make changes to the OSB project as needed.
5. Export the updated project as a jar to RSB_WORK_AREA. The name and location of the jar should be same as the jar that was imported.
6. Copy the updated jar to rsb-home/service-assembly-home at the same location from where it was copied.
7. Follow the RSB compilation and deployment process to deploy the modified jar in server.

Note: After copying modified jars into rsb-home/service-assembly-home, do not run the download-home/bin/check-version-and-unpack.sh script because that will overwrite the jars with the original jars from the PAKs.

How to Import RSB Decorator jar into OEPE

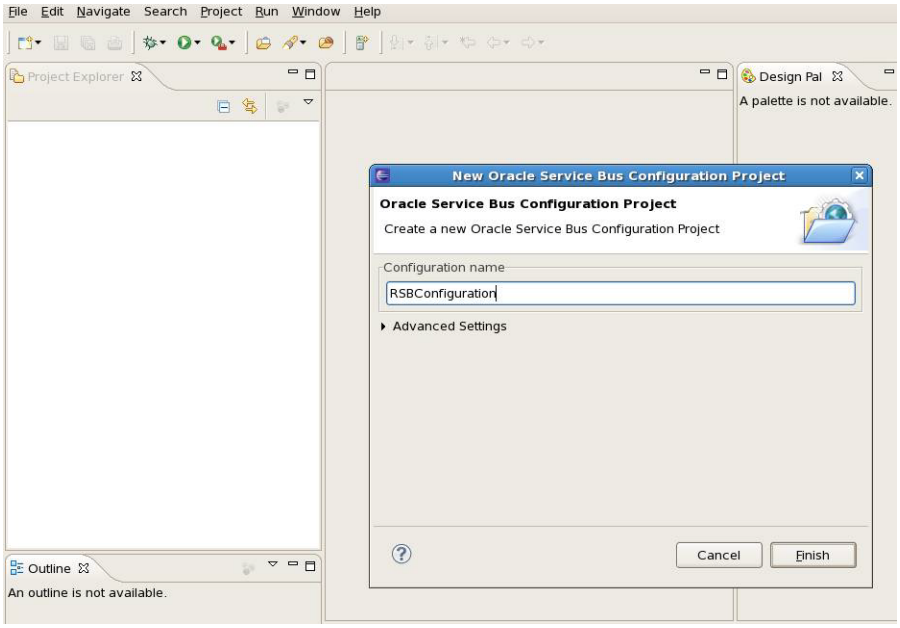
This section provides a step-by-step guide for how to import RSB decorator jar into OEPE workspace.

1. If you already have OSB_WORK_SPACE open as eclipse workspace, then you can ignore this step. Otherwise, you need to switch to the right workspace. To do so, select **File > Switch Workspace > Other**. Enter the path to the workspace as in the following screen.



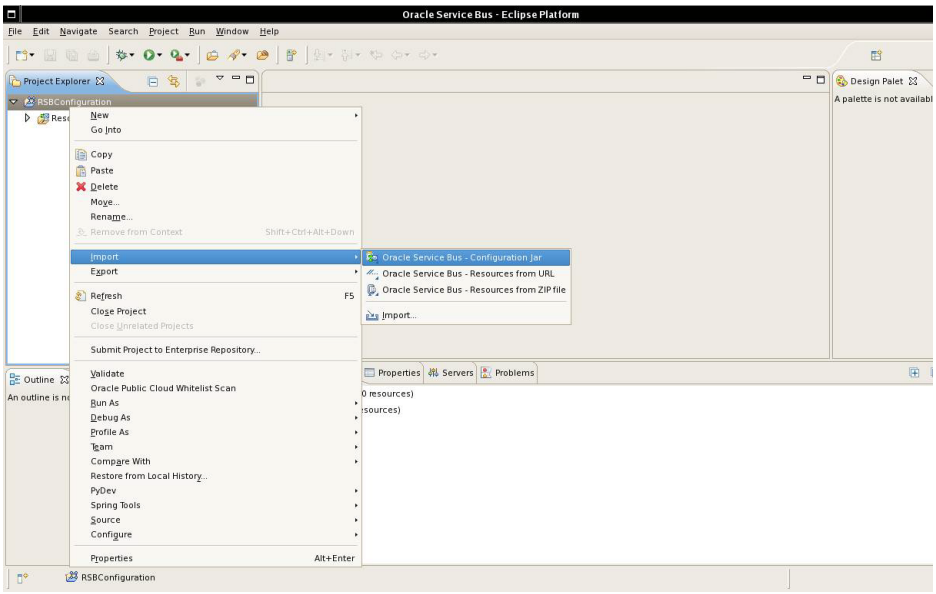
2. In Eclipse, to switch to Oracle Service Bus perspective, click **Window > Open Perspective > Oracle Service Bus**.

3. Now we need to create OSB configuration project. Select **File > New > Oracle Service Bus Configuration Project**. Enter the project name as RSBConfiguration.

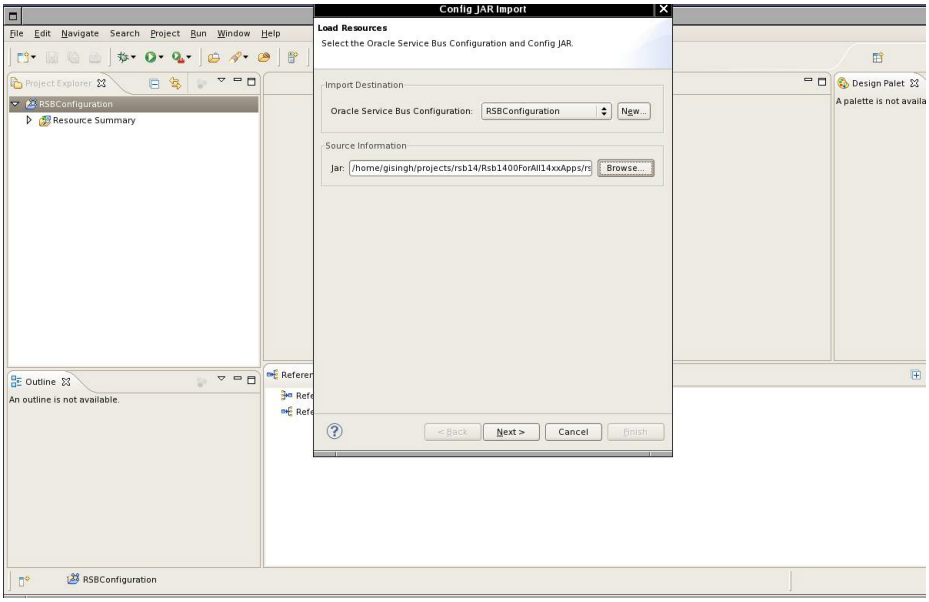


Click **Finish**. This creates an OSB Configuration Project, in which you can import RSB decorator or service integration flow jar files.

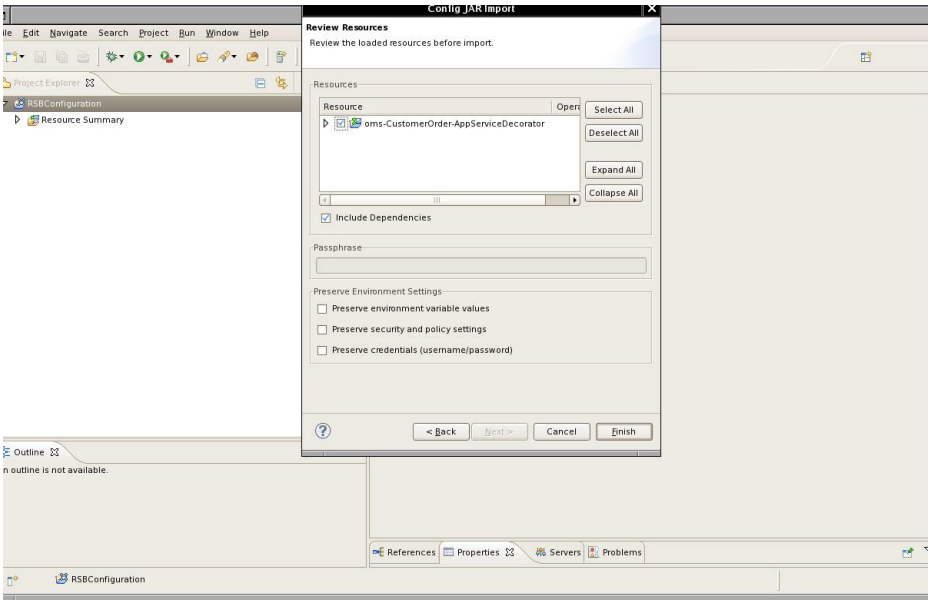
4. Now to import a decorator jar, right-click RSBConfiguration and select **Import > Oracle Service Bus - Configuration Jar** as shown below.



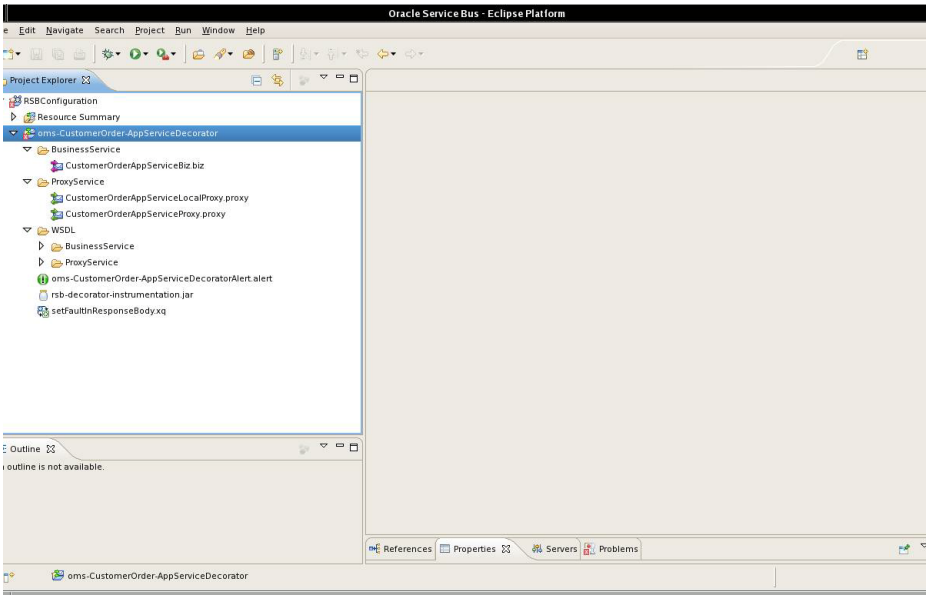
5. Specify the path to the jar file that you want to import.



6. Click **Next**.



7. Click **Finish**.



Now the decorator jar will be imported into workspace, and the jar will be extracted inside the workspace. It will also show the project name in the Project Explorer window of OEPE.

With this the workbench area is setup for the OSB project development. Any changes you make here will be saved in the workspace. After you finish making changes, you can export the project to a jar file.

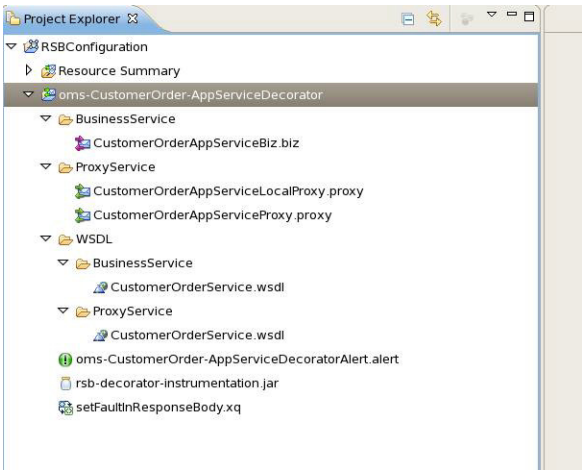
Note: While making changes to OSB projects, it may be cumbersome to copy the jar to rsb-home/service-assembly-home every-time and go through compilation and deployment phases to test the changes. Instead, you can test the OSB project by deploying the jar in a development OSB server environment and test that all the changes are working fine. Once the changes are working as desired, the jar file can be copied to rsb-home/service-assembly-home for final deployment.

Components of RSB Decorator Project

After importing a decorator jar into workspace, the directory structure looks like following:

```
<appName>-<ServiceName>-AppServiceDecorator
    BusinessService
    <ServiceName>AppServiceBiz.biz
    ProxyService
    <ServiceName>AppServiceLocalProxy.proxy
    <ServiceName>AppServiceProxy.proxy
    WSDL
    BusinessService
    <ServiceName>Service.wsdl
    ProxyService
    <ServiceName>Service.wsdl
    <appName>-<ServiceName>-AppServiceDecoratorAlert.alert
    Rsb-decorator-instrumentation.jar
    setFaultInResponseBody.xq
```

An example screenshot when oms-CustomerOrder-AppServiceDecorator.jar is imported is shown below.



Business Services

RSB Decorator projects include one business service by default. This business service is based on the WSDL which is available in WSDL BusinessService folder. By default, the WSDLs of Proxy and Business Services are similar. When customizing a decorator to work with an external service, the WSDL of that service should be copied in this folder. Following is the naming convention for business service file: <ServiceName>AppServiceBiz

For example, in oms-CustomerOrder-AppServiceDecorator project, the name of business service will be CustomerOrderAppServiceBiz

Local and Remote Proxy Services

RSB Decorator projects include two proxy services by default. They have naming convention as follows:

- <ServiceName>AppServiceLocalProxy
- <ServiceName>AppServiceProxy

<ServiceName>AppServiceProxy: This proxy service is based on HTTP transport protocol. Clients invoking the service from remote JVM should invoke this proxy service. This service takes request from the web service client and routes it to <ServiceName>AppServiceLocalProxy service. This proxy service does not have any business logic, its only purpose is to allow invocation from remote clients and it is recommended to be kept that way.

<ServiceName>AppServiceLocalProxy: This proxy service is based on local transport. All the message processing takes place in this proxy service.

Why two proxy services in decorator jar? Every decorator project has two proxy services packaged inside it. The reason for that is to provide the flexibility to call the service either from remote or from local JVM. It also provides the flexibility to configure the security as needed. When a proxy service needs to invoke another proxy service, it can directly invoke the local proxy service which will save the overhead of processing security headers of that message.

WSDL files

Every decorator project has two WSDL files packaged in it.

Proxy Service WSDL: This WSDL is available under **WSDL>ProxyService folder**. The proxy services packaged in a decorator jar are based on this WSDL. This WSDL should never be modified as consumers invoke the decorator services based on this WSDL, any change to this WSDL will break the service contract.

Business Service WSDL: This WSDL is available under **WSDL>BusinessService** folder.

Alert Destination

Every decorator jar has an alert destination packaged in it. The filename of the destination follows the format:

`<appName>-<ServiceName>-AppServiceDecoratorAlert.alert`
This is the default alert destination which logs the alert as well as sends the alert to default reporting JMS provider. Any pipeline or SLA alerts configured in decorator will be sent to this destination.

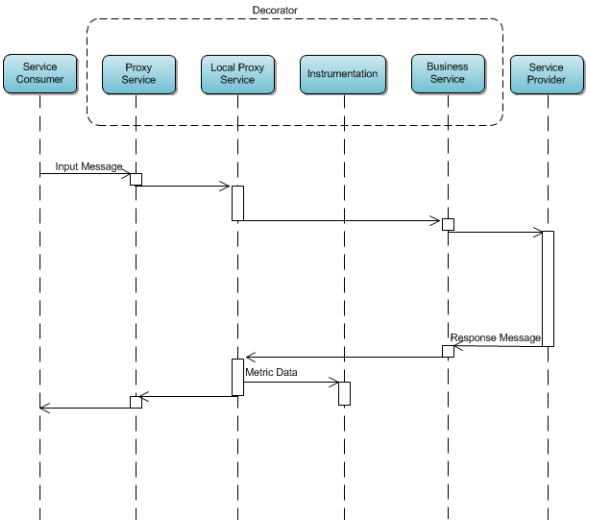
Instrumentation Jar File

A java archive file named `rsb-decorator-instrumentation.jar` is packaged in the decorator project. This jar contains java classes which contain the code for instrumentation purposes.

Fault XQuery File

There is an xquery file named `setFaultInResponseBody.xq` packaged in every decorator project. This xquery contains the code to build appropriate SOAP fault before returning it to the client.

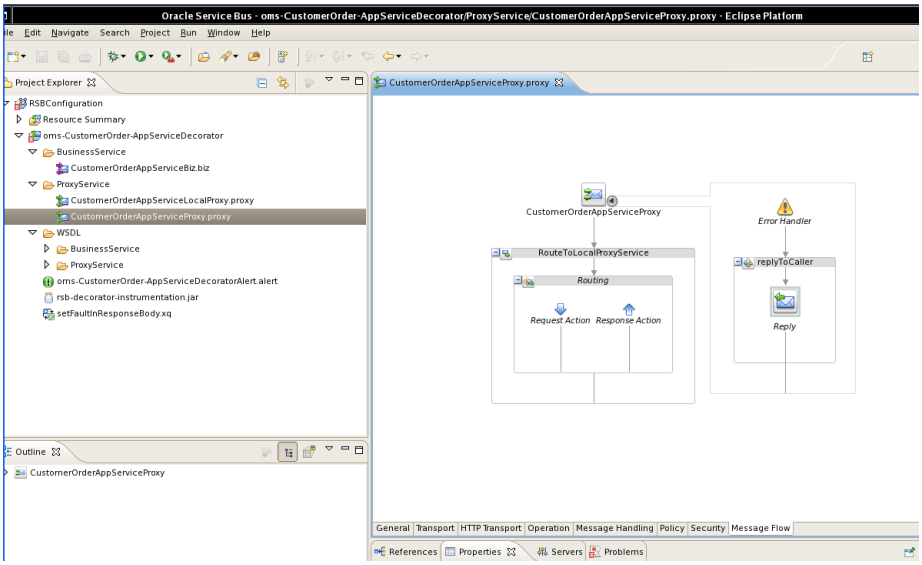
RSB Decorator Message Flow



Proxy Service client > Remote Proxy Service > Local Proxy Service > Business Service > Edge-app Application Service.

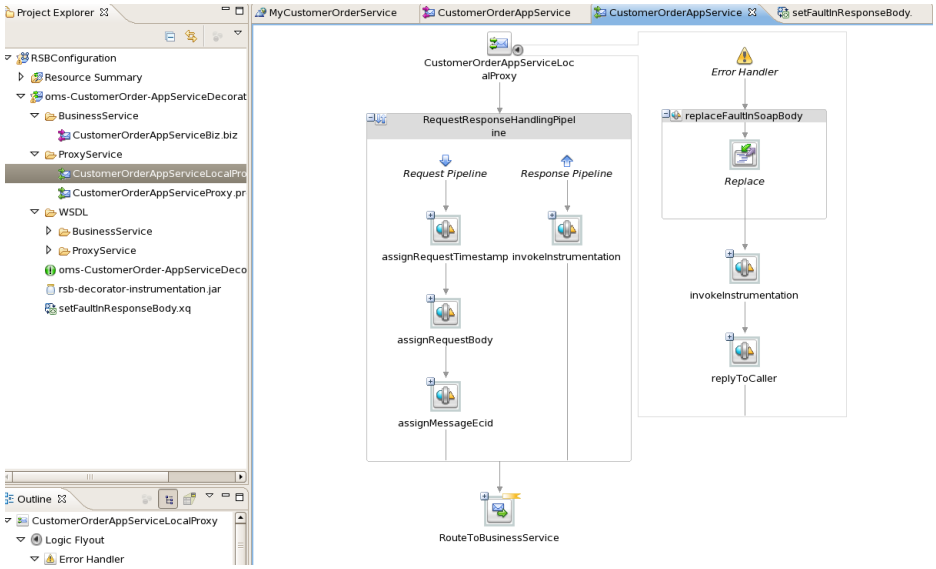
Message Flow in Remote Proxy Service

The following is a screenshot of remote proxy service of a decorator jar:



Message flow in Local Proxy Service

The following screenshot shows the message flow in a local proxy service of a decorator jar:

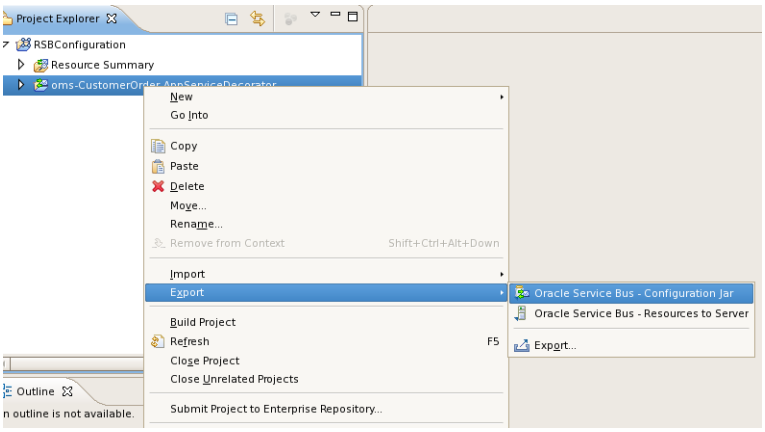


How to Export RSB Decorator Project

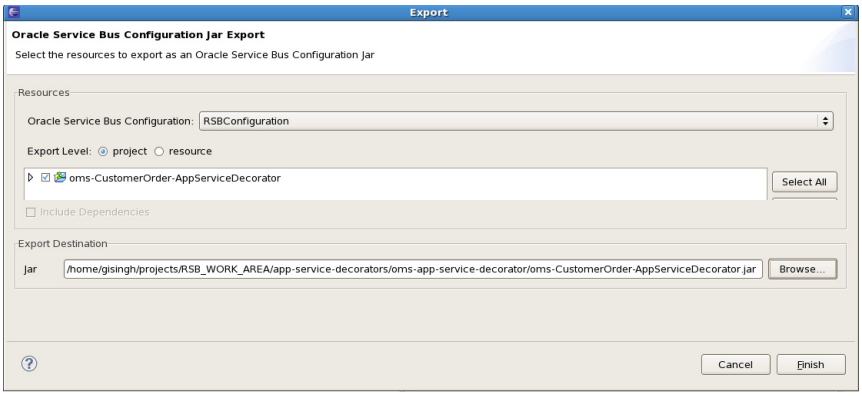
Once you have completed changes in a decorator project, you can export it back to the jar and deploy and test.

To export the decorator project to the jar do the following:

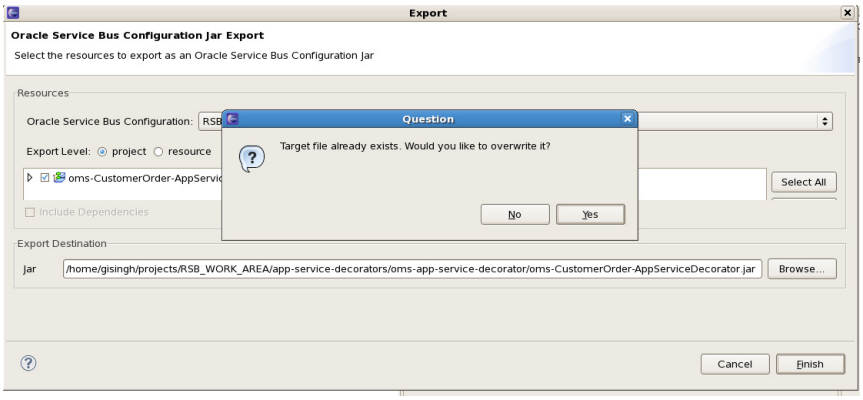
1. Right-click the project you want to export, and select **Export > Oracle Service Bus - Configuration Jar**.



2. Select the jar file name, this should be the same jar that was imported originally into OEPE.



3. Click **Finish**.



4. Click **Yes**. It will update the jar file with latest changes. The export process is complete.

3

Integration with Third-Party Application Services

Oracle Retail application landscape of the customer has a variety of applications servicing different business functions. There is a legitimate need to integrate these applications. Customers may have one or more of these applications from vendors other than Oracle. Both Oracle and non-Oracle applications should be able to integrate as long as the interface requirements are met.

In this document we are describing the process and instructions to integrate a third-party application to an Oracle Retail application using RSB. We will be providing instructions and example to show how to integrate using RSB. For this purpose, we are assuming Order Management System (OMS) as the third-party application. OMS is only a representational application that implements retail order management functionality. Actual applications that will be used instead can be any third-party applications like Yantra, and so on.

Any third-party applications that can consume or provide SOAP based web-services can be integrated with Oracle Retail application through RSB. While there can be complex integration scenarios, this document describes only those where the services can be integrated by adapting the interfaces of the consumer and provider. This adaptation is done by modifying message in OSB layer.

Types of Customization

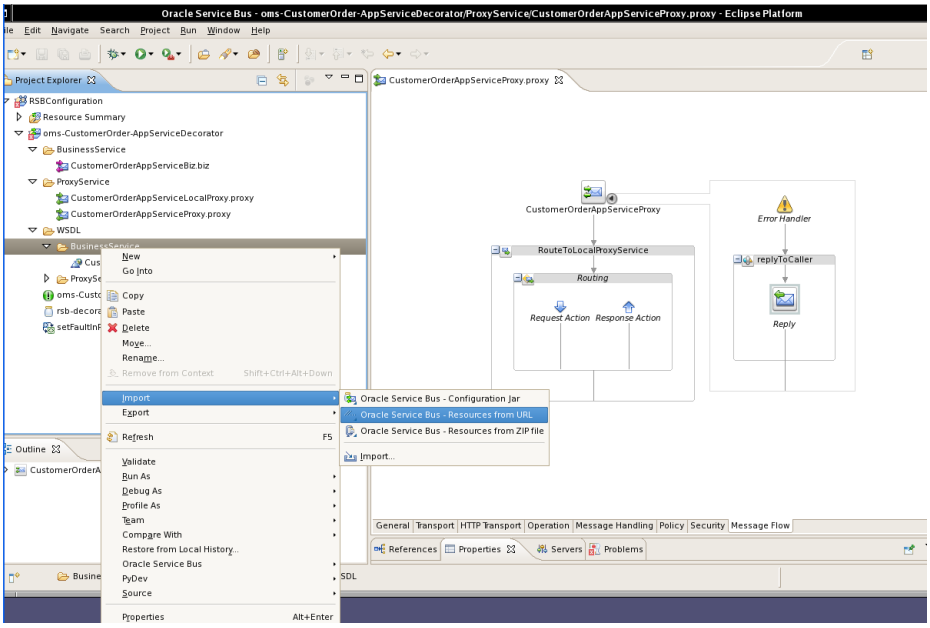
The web services you want to integrate to RSB is likely to be different from what the corresponding provided decorators expect. These differences can be broadly classified into two categories:

- Payload is different
- Service and operation names are different

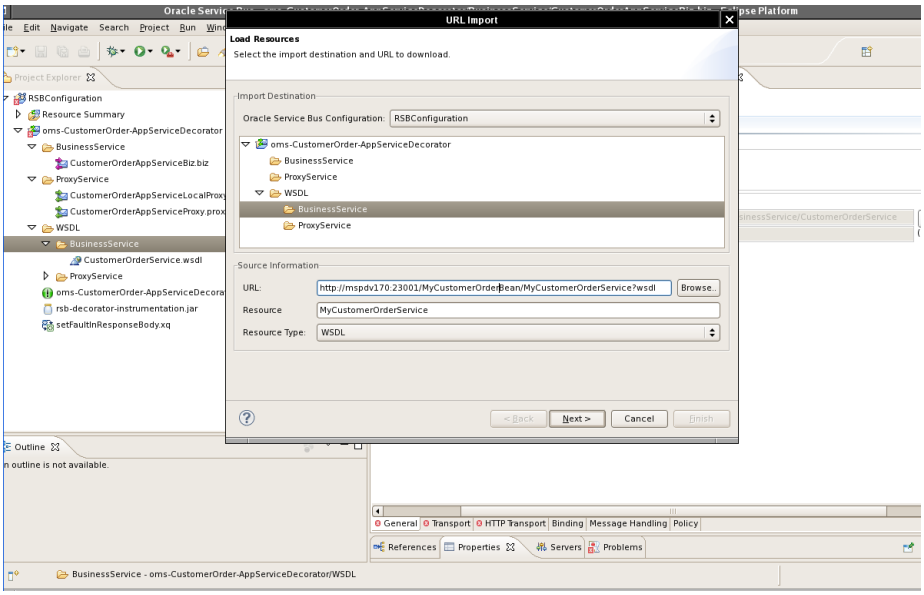
How to Import WSDL into RSB Decorator Project

In order to integrate with third-party application services, first step is to import the WSDL file of third-party application service into the decorator project. Following are the steps:

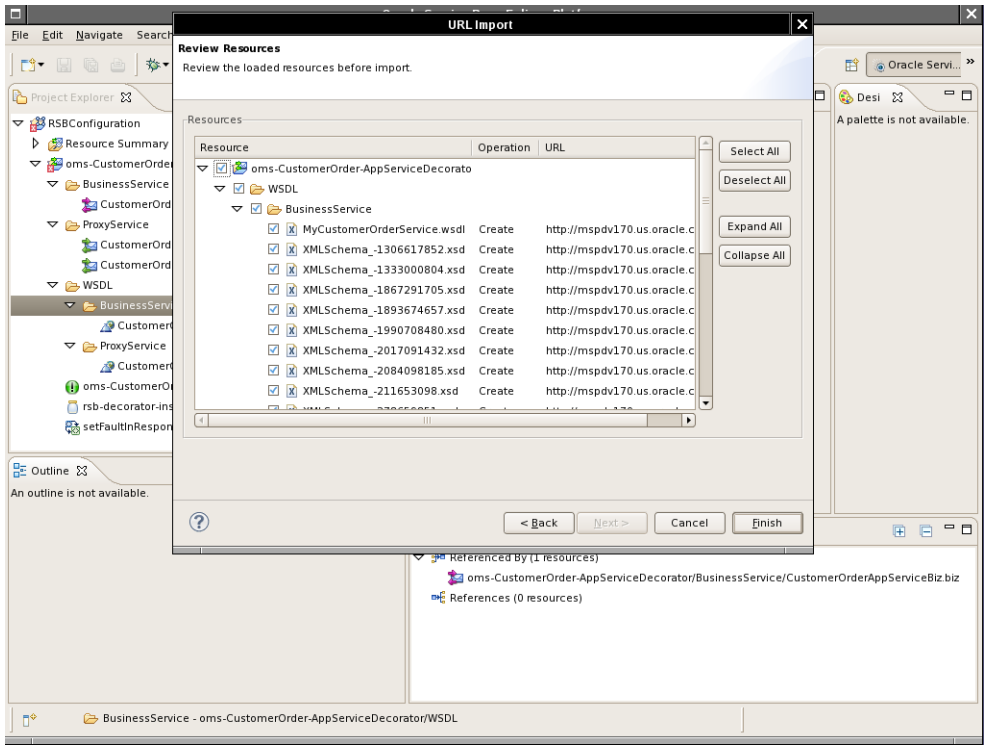
1. Right-click **WSDL >BusinessService** and select **Import >Oracle Service Bus - Resources** from the URL.



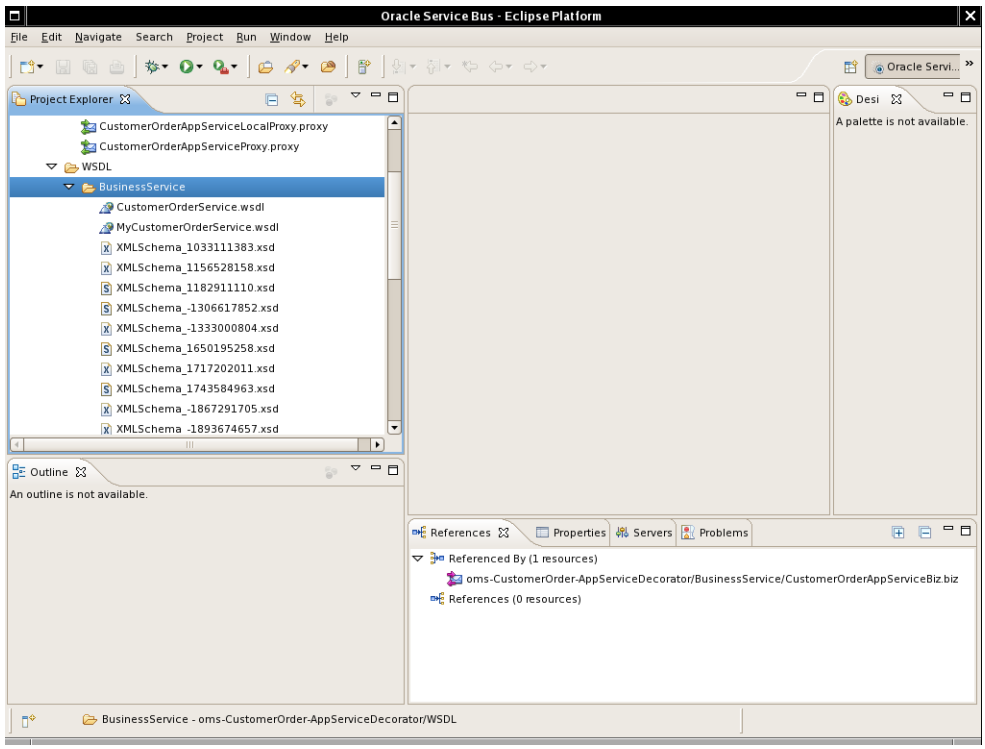
2. Select Resource Type as **WSDL**. Also provide the URL of the WSDL. Alternatively, if you have the WSDL downloaded to the local machine, you can browse to that location and select the WSDL file. Here open the WSDL and verify the WSDL can be successfully accessed.



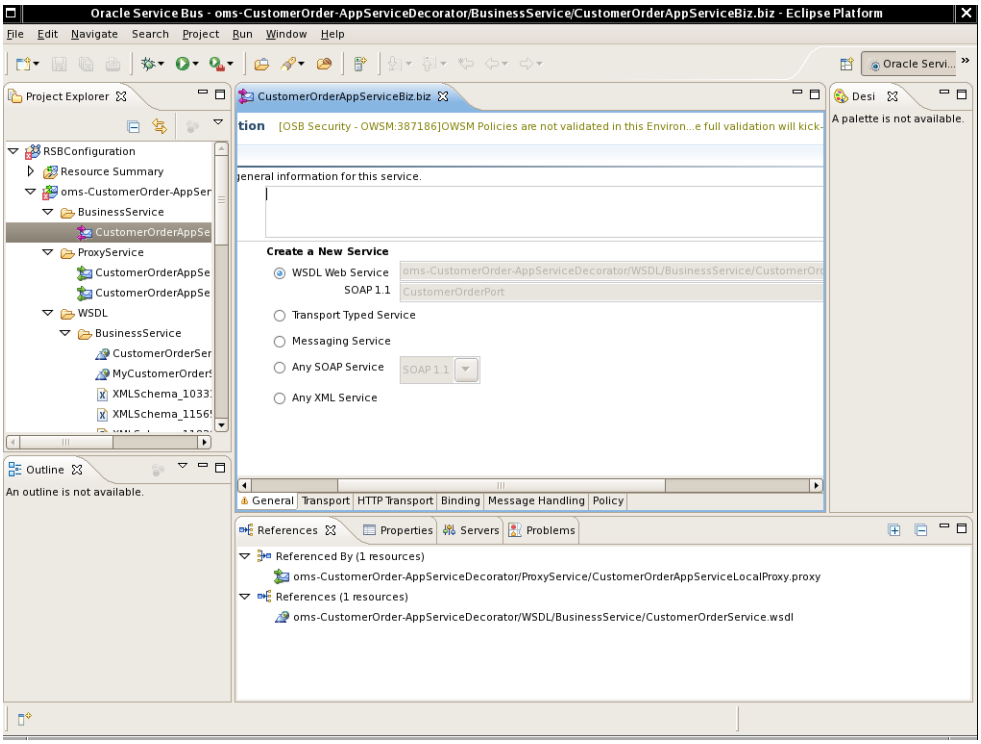
3. Click **Next** button. It will show the WSDL and all related XSD files:



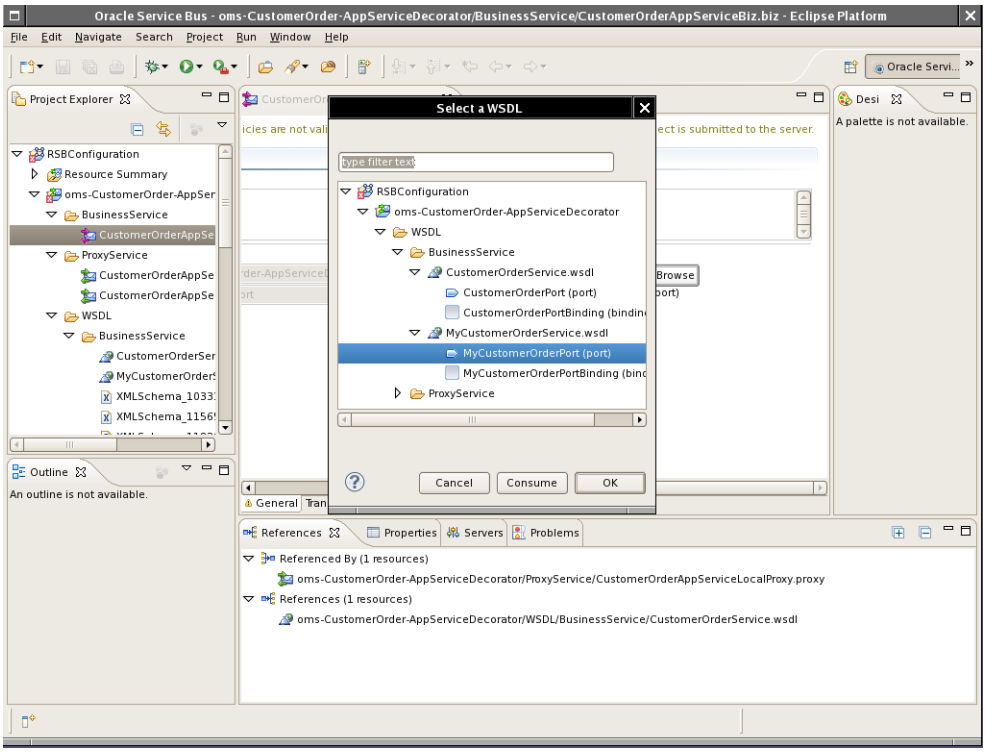
4. Select the WSDL file and click **Finish**. As you can see, the new WSDL file is added to the project:



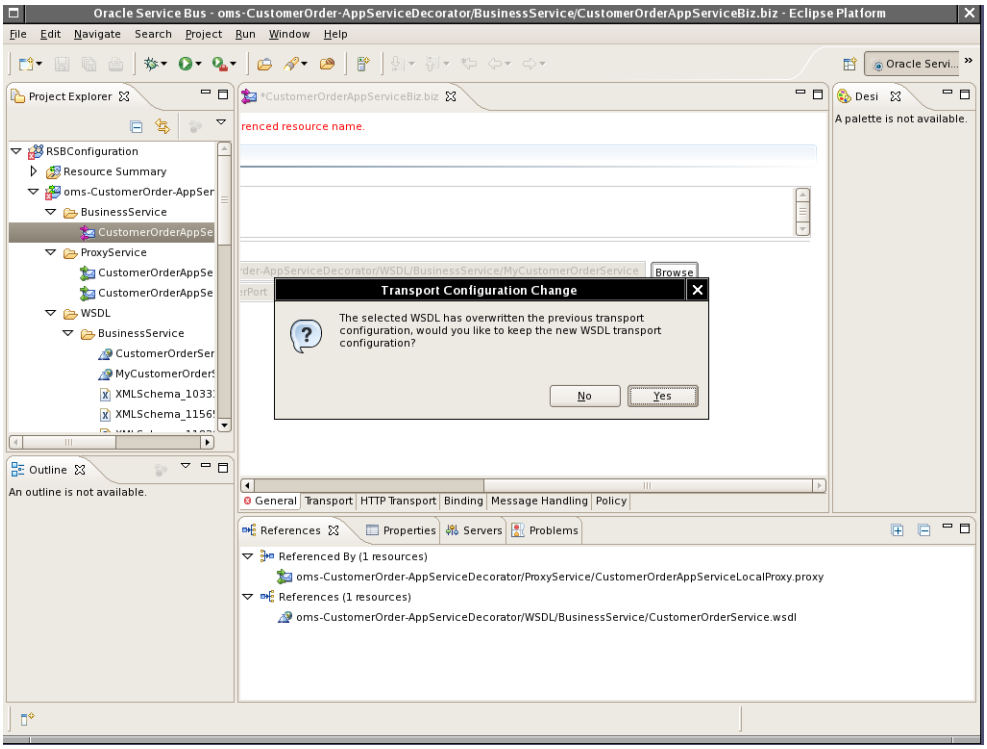
5. The next step is to modify the business service to use new WSDL. In order to do so, open the business service file and go to General tab:



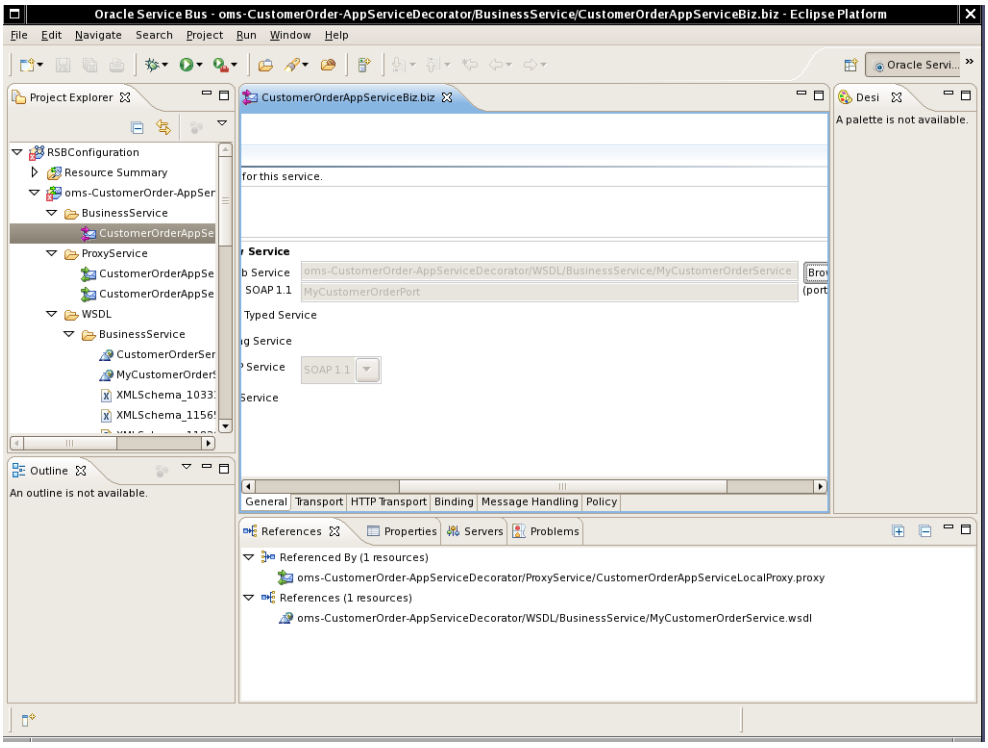
6. Click **Browse** for WSDL Web Service field.



7. Select the new WSDL (for example, CustomerOrderPort(port)) and click **OK**.



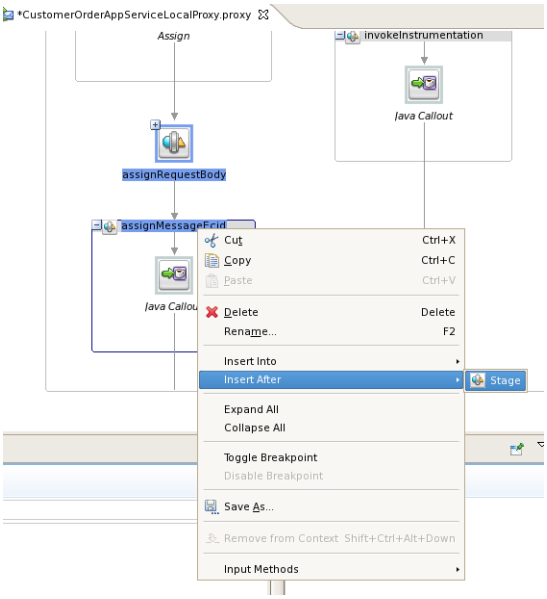
8. Click **Yes**. And the new business service should be displayed.



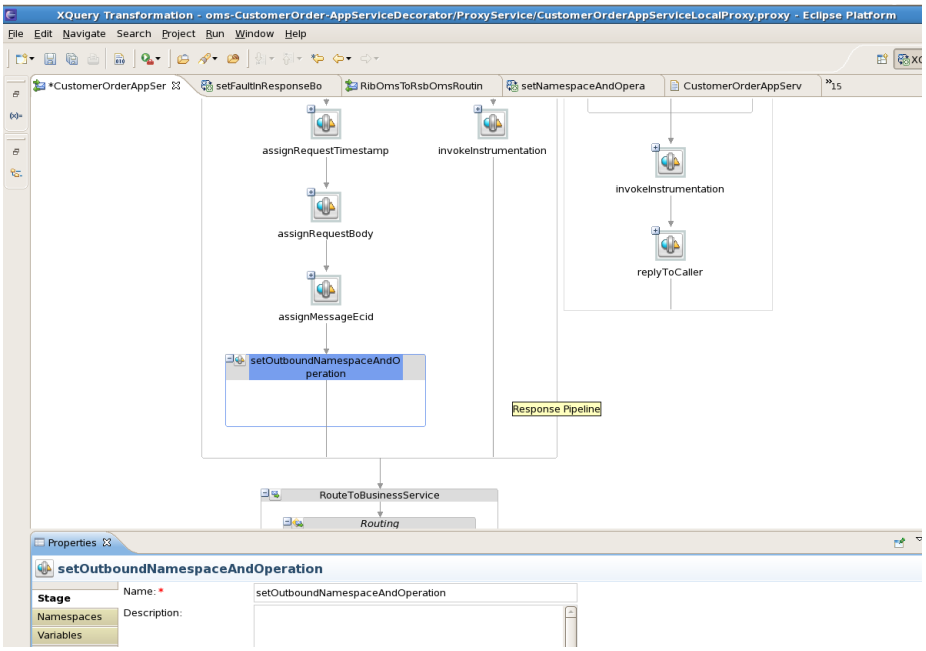
How to Map Namespaces and Operation Names

When the business service is changed to use a new WSDL, then the SOAP request of proxy service will not work as-is with the new business service because the business service WSDL may have different namespaces and names for operations and services. So now the proxy service message flow will need to be modified to transform the incoming message to the expected format of business service. In order to do these transformations XQuery files can be used. A sample for making these changes in proxy service message flow is shown below.

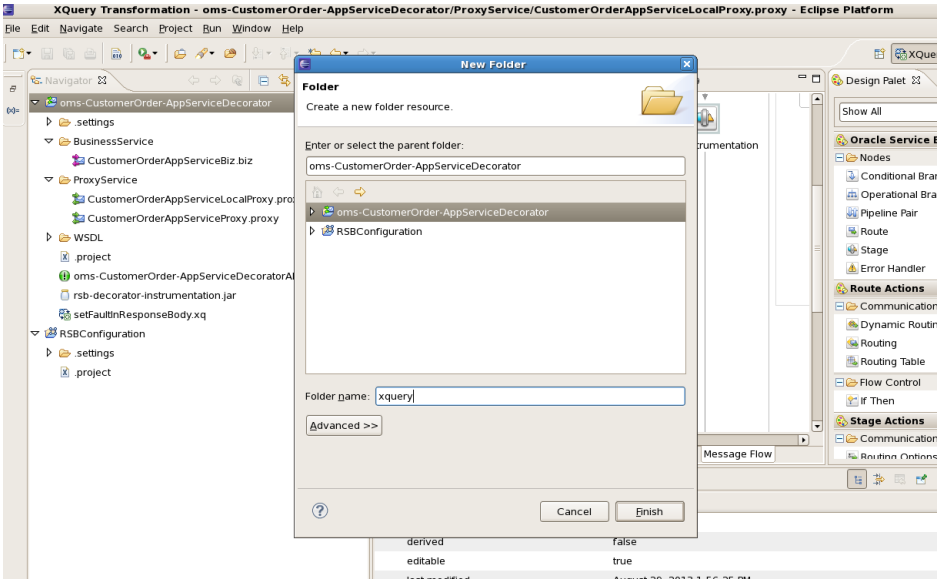
1. The first step is to add a new stage in request pipeline of local proxy service message flow. To add a new stage, right-click the assignMessageEcid stage, and select **Insert After > Stage**.



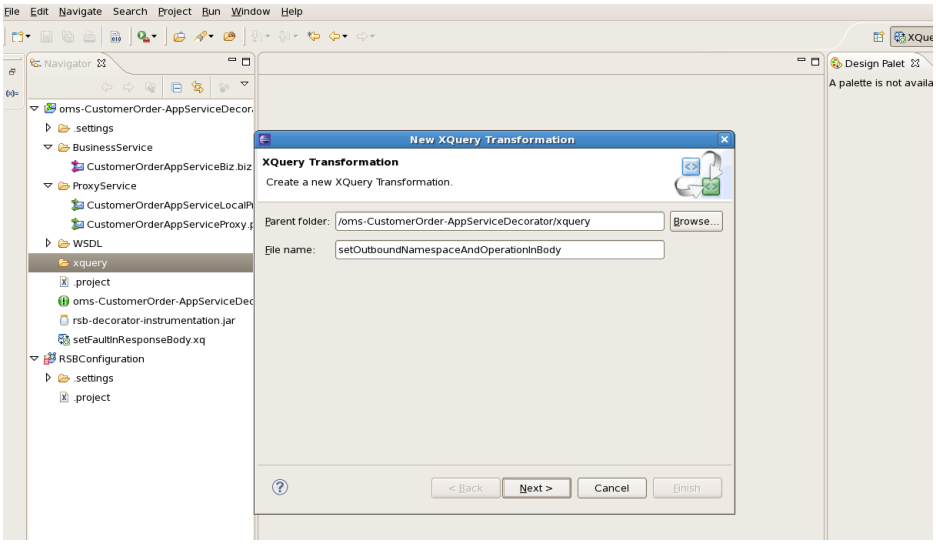
2. Enter the name of stage as setOutboundNamespaceAndOperation.



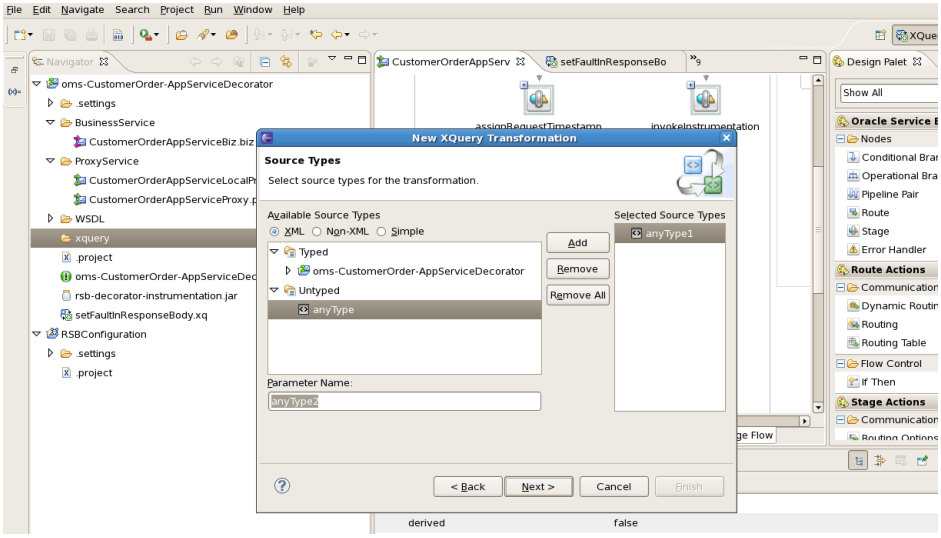
3. To create a new xquery file, you can create a folder xquery where all the xquery files will be saved. To create the folder, right-click the project name and select **New>Folder**. For Enter or select the parent folder, verify the AppServiceDecorator is selected For e.g. oms-CustomerOrder-AppServiceDecorator. Enter **xquery** as the folder name.



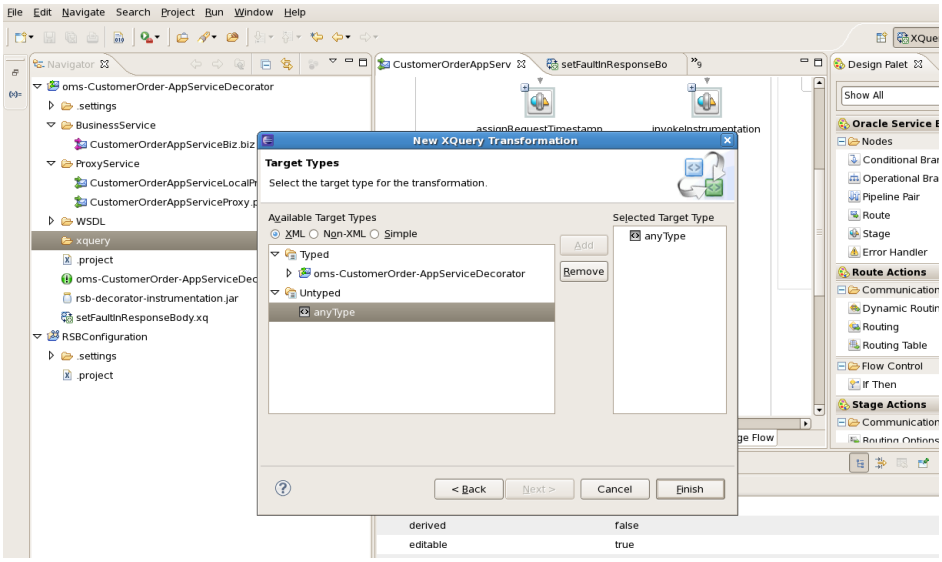
4. Now right-click the xquery folder and select **New>XQuery Transformation**. Enter the name setOutboundNamespaceAndOperationInBody.



5. Click **Next**. Now we need to select Source type for transformation. Since this is not based on any xsd, select **anyType**, and then click **Add**. Verify anyType is displayed under the Selected Source Types.

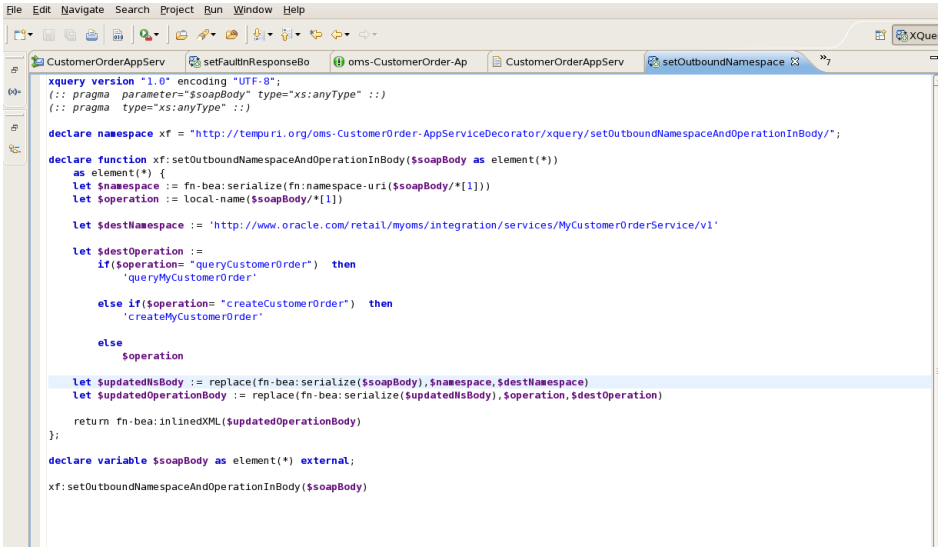


6. Click **Next** and select **anyType** for target Types as well. Click **Add**, and verify anyType is listed under the Selected Target Type.



7. Click **Finish**. In the source view of the file, enter the code as shown in the screenshot below. In this code, the variable $namespace contains the namespace of the incoming request xml and $operation contains the operation name in the incoming request. Further, we check for each incoming operation name and assign the corresponding outbound operation name in $destOperation variable.

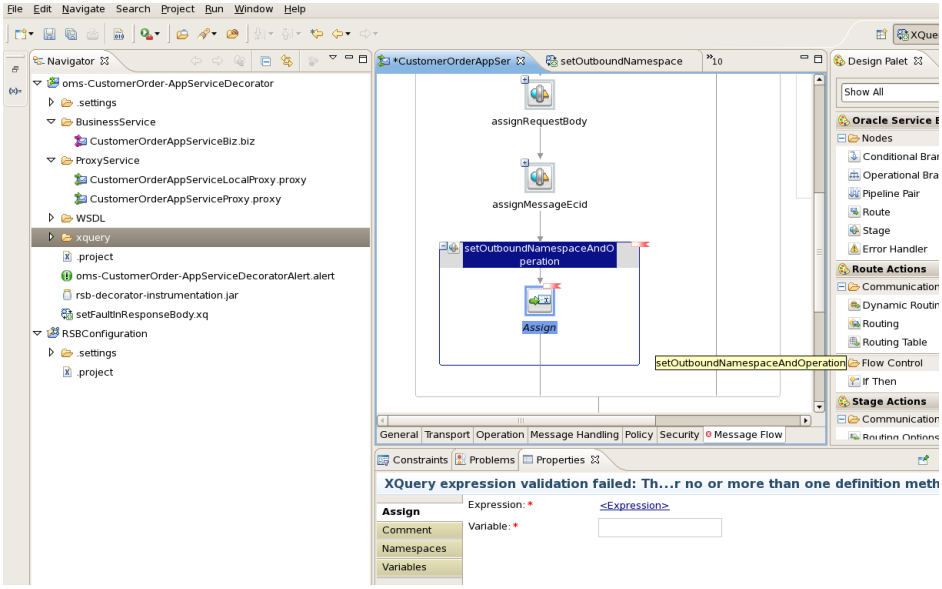
   For example, when the incoming operation name is queryCustomerOrder, the outbound operation name needs to be queryMyCustomerOrder. The namespace is at service level, so we find the service namespace from the new business service WSDL and assign it to $destNamespace variable. The sample xquery shown in the screenshot is listed in Appendix A. You can copy the code and make changes appropriate to your requirements.



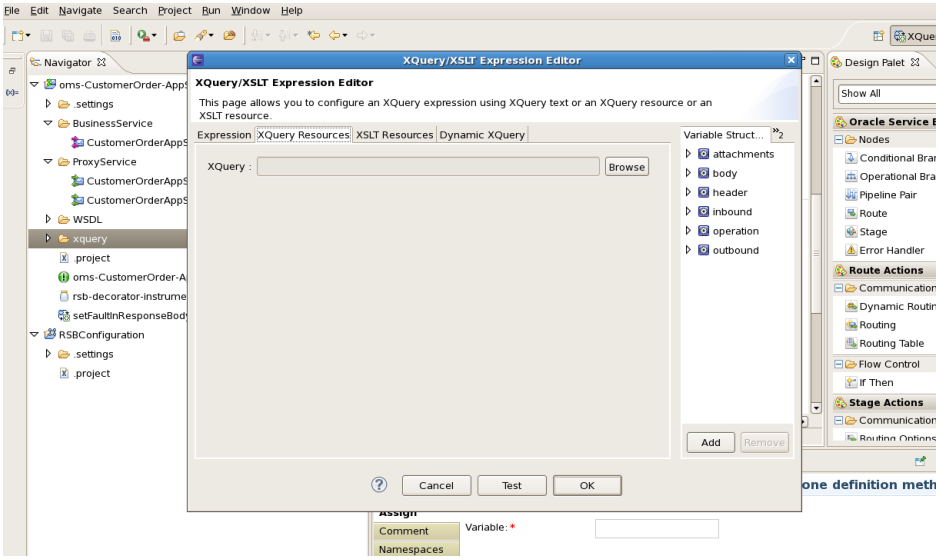
8. Return to the stage in message flow and add an Assign action. Steps to add Assign action are:

Right click the stage, that is, **setOutboundNamespaceAndOperation**.

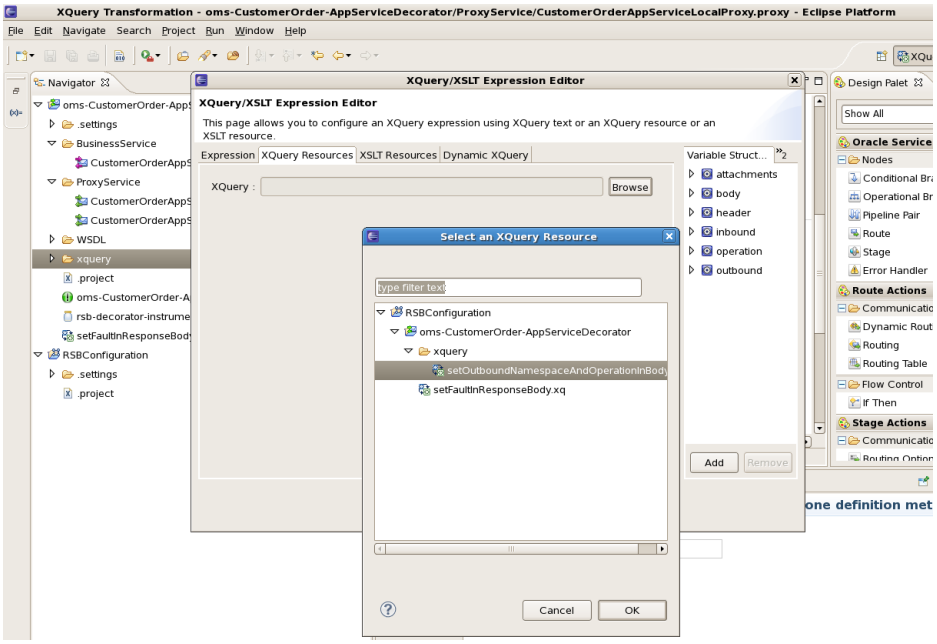
Select **Insert Into > Message Processing > Assign**.



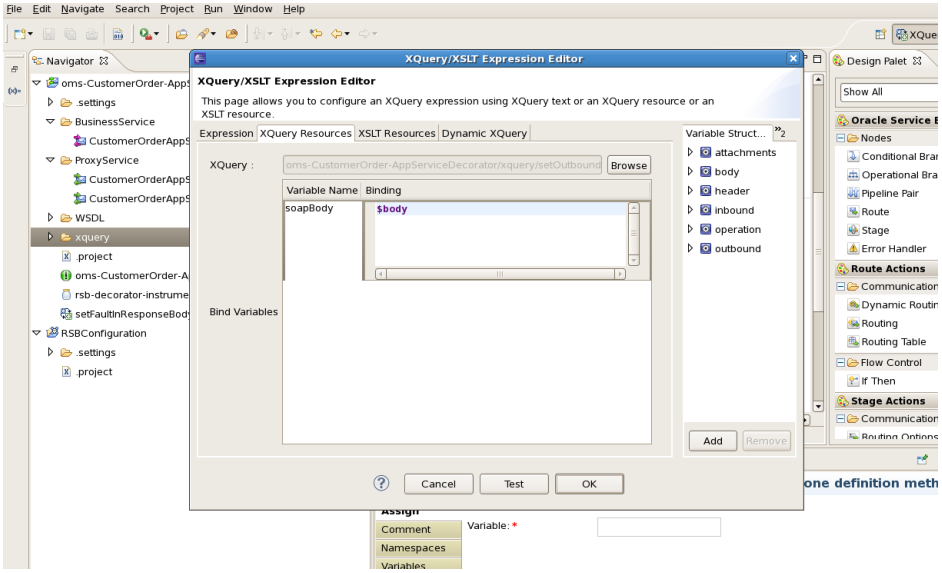
9. In the Expression field of Assign action, click the **<Expression>** link and go to XQuery Resources tab:



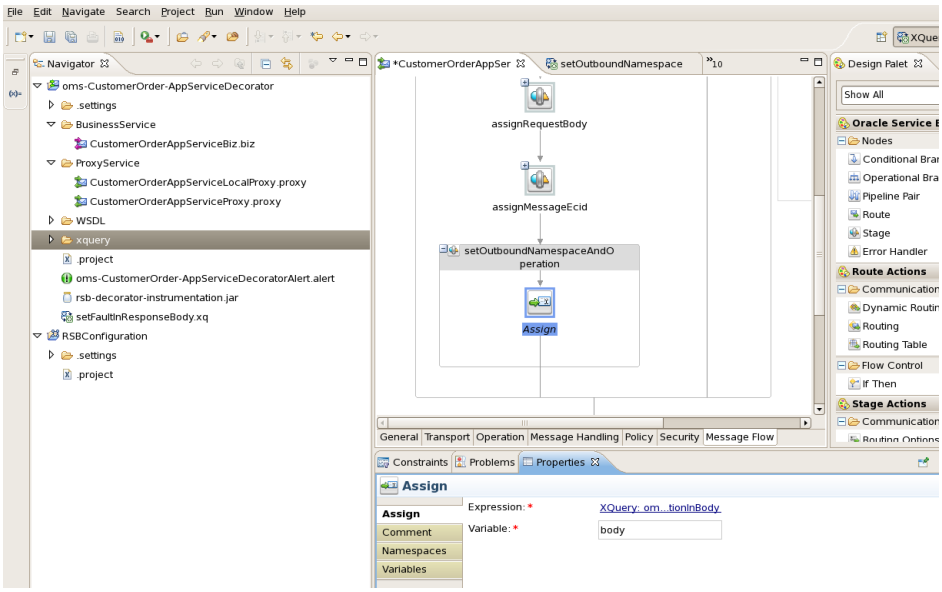
10. Click **Browse** and select the new xquery file.



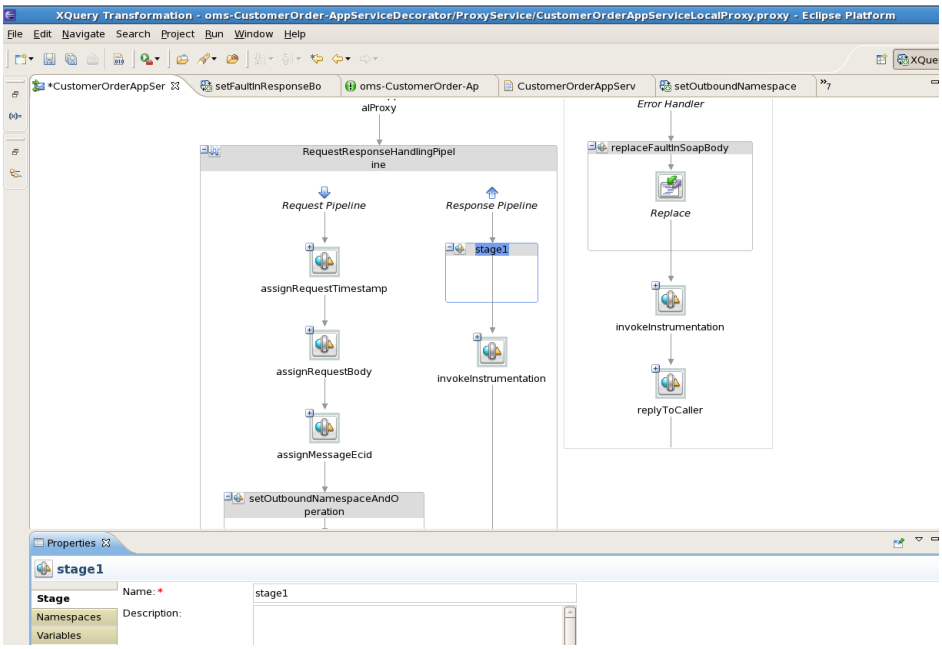
11. Click **OK** and in the **Binding** field, enter the value as **$body**.



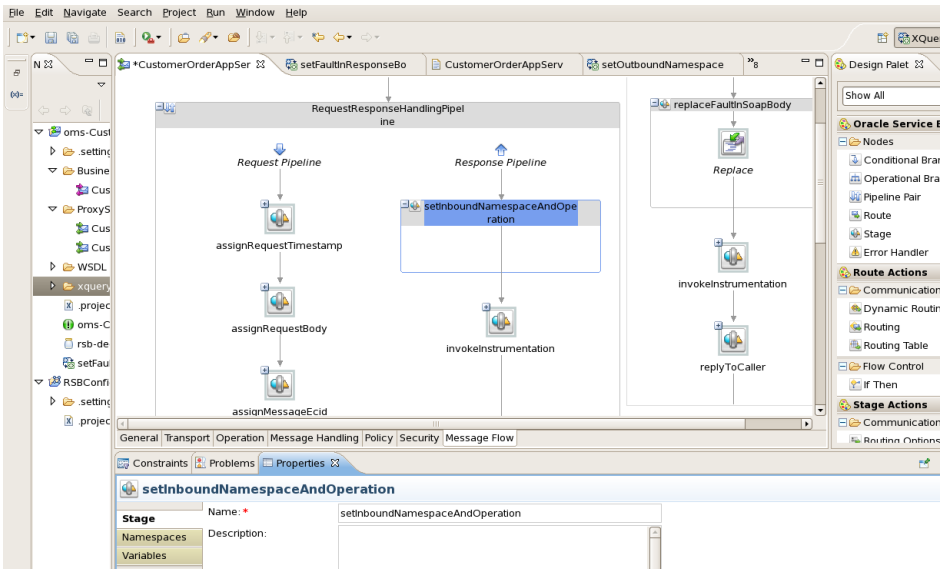
12. Click **OK**. In the Variable field of Assign action, enter the value as **body**.



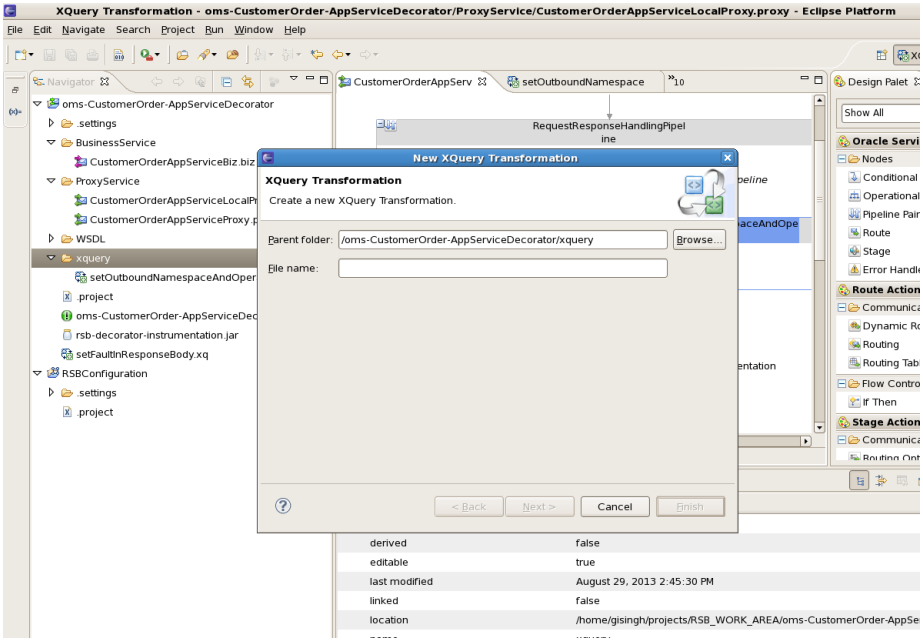
13. Perform similar transformation of namespaces and operation names in the response pipeline but in the reverse order. This is because the response returned from business service must be converted to the response message format which conforms to the proxy service WSDL. To do this, add a new stage in response pipeline.



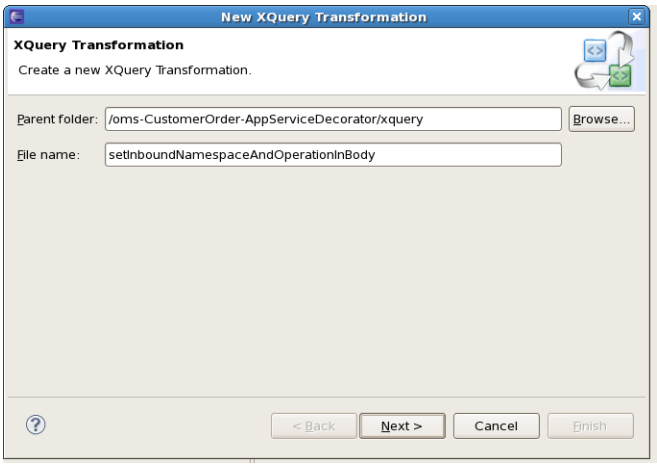
14. Enter the stage name as setInboundNamespaceAndOperation



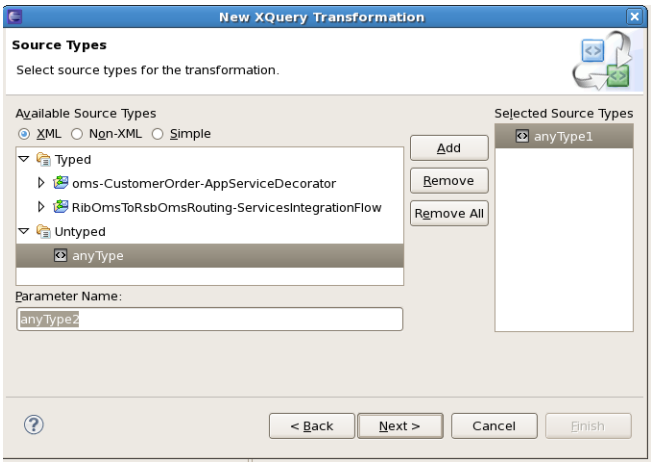
15. Now we need to create an xquery file to do the mapping. Right click the xquery folder and select **New > XQuery Transformation**.



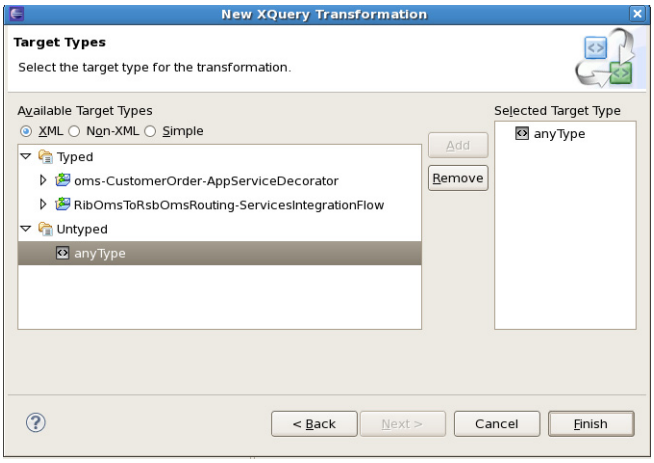
16. Enter file name as setInboundNamespaceAndOperationInBody.



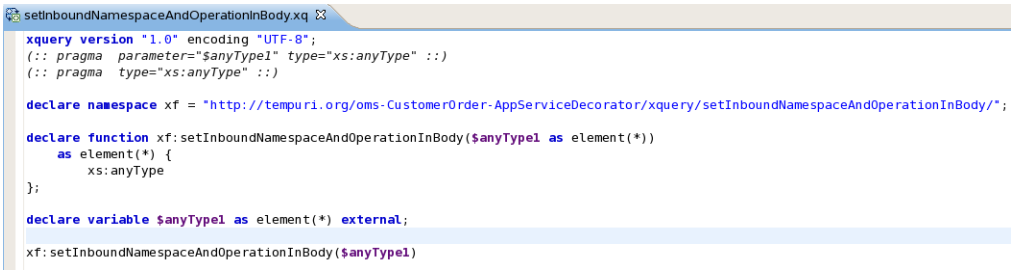
17. Click Next. Select anyType for Source Types field:



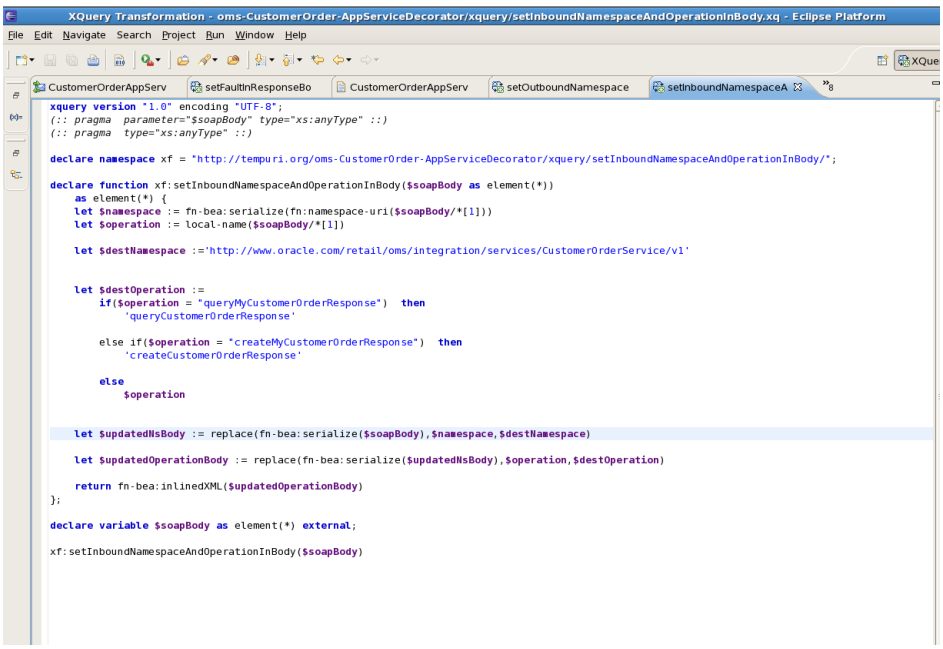
18. Click Next and in Target Types select anyType



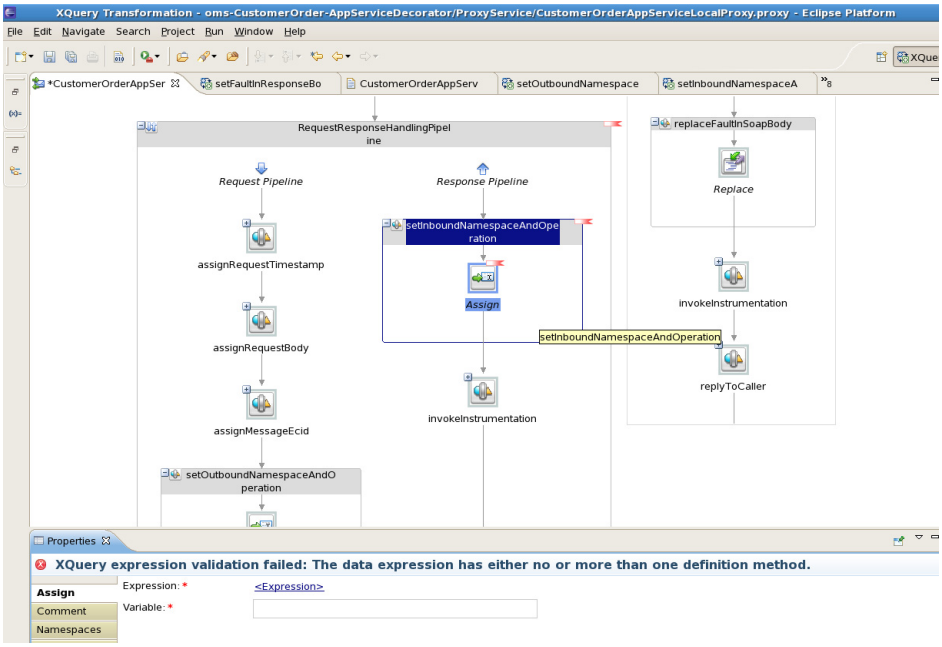
19. Click Finish. Goto the Source tab of xquery file:



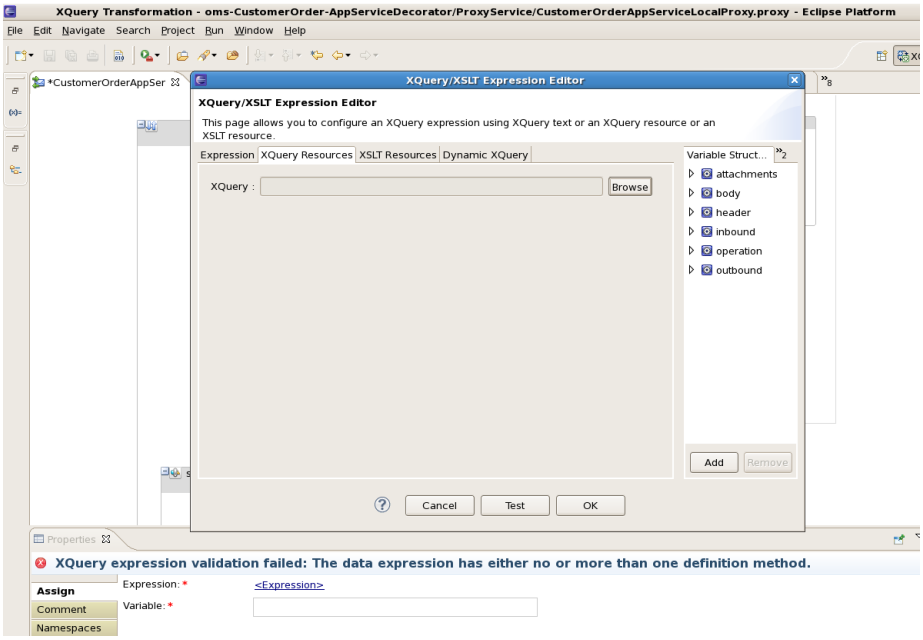
20. Enter the code as shown below. In this code, the variable $namespace contains the namespace of the response xml and $operation contains the operation name in the response. Further, we check for each operation name and assign the corresponding proxy service operation name in $destOperation variable. For example, when response operation name is queryMyCustomerOrderResponse then the new operation name needs to be queryCustomerOrderResponse. The namespace is at service level, so we find the service namespace from the proxy service WSDL and assign it to $destNamespace variable. The sample xquery shown in the screenshot is listed in Appendix A. You can copy the code and make changes appropriate to your needs.



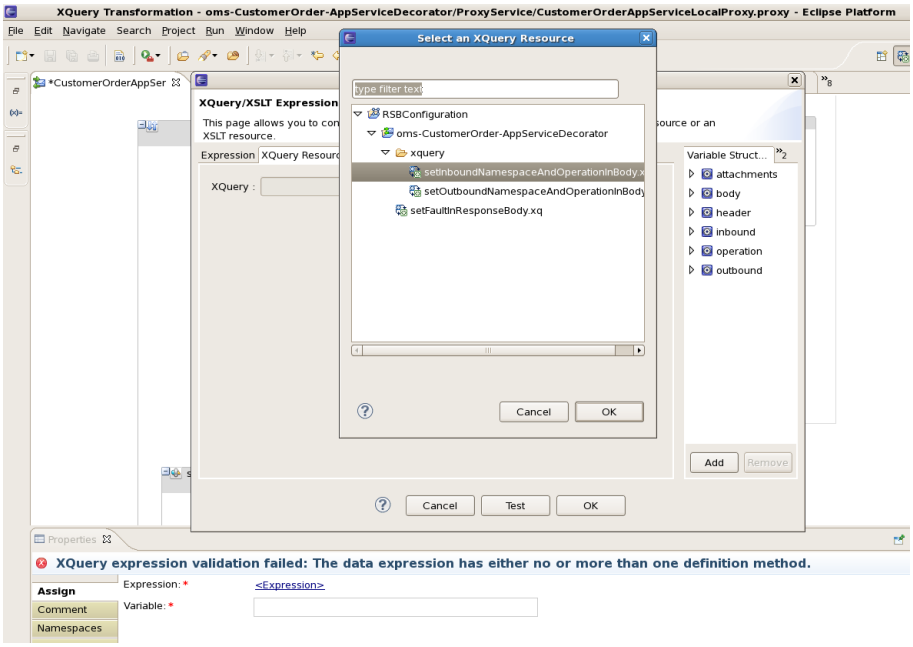
21. In the message flow, in the setInboundNamespaceAndOperation stage add an Assign action by right-clicking on setInboundNamespaceAndOperation stage and select **Insert Into > Message Processing > Assign**.



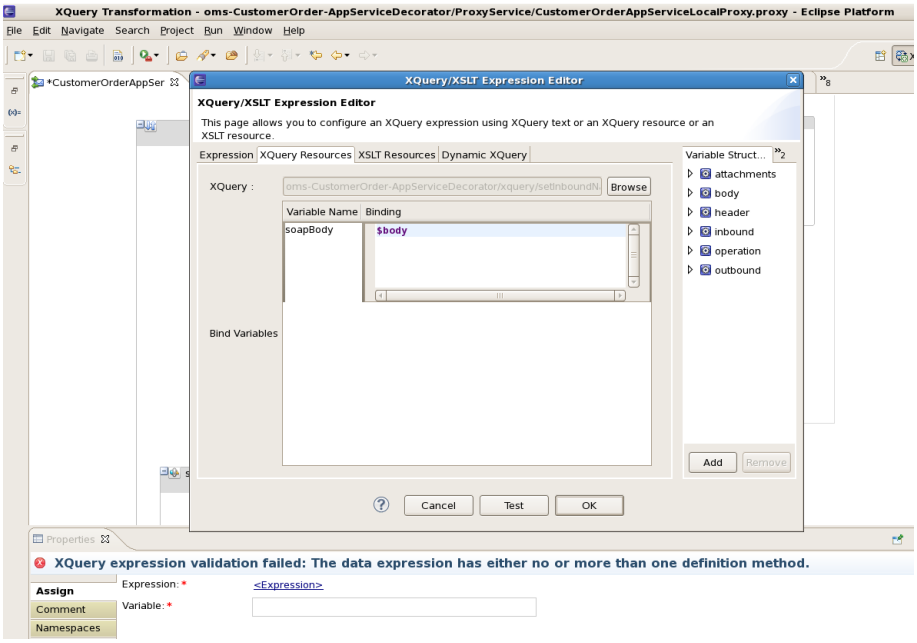
22. In the Properties window of Assign action, click the **Expression** link to open the Expression dialog box and switch to XQuery Resources tab.



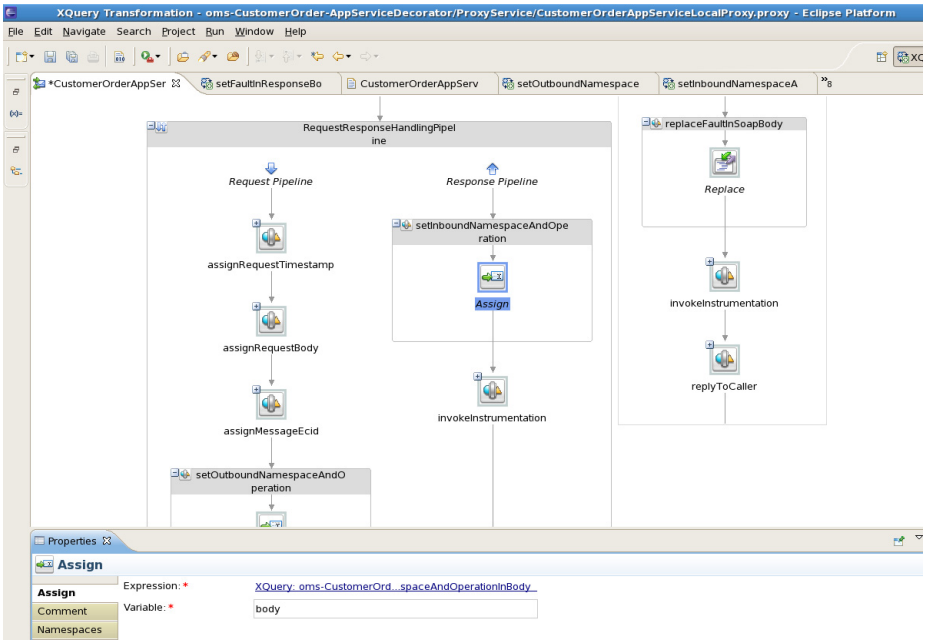
23. Click **Browse** and select the xquery file.



24. Click **OK**. In the Binding column, enter the value **$body**.



25. Click **OK** and in the Variable field, type **body**.

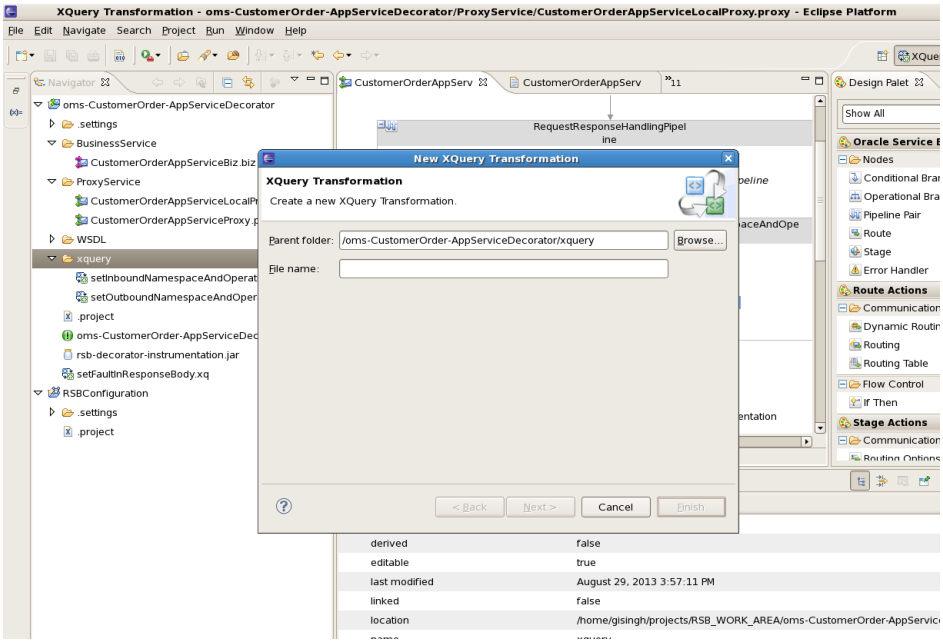


The above completes the steps for namespace and operation mapping.

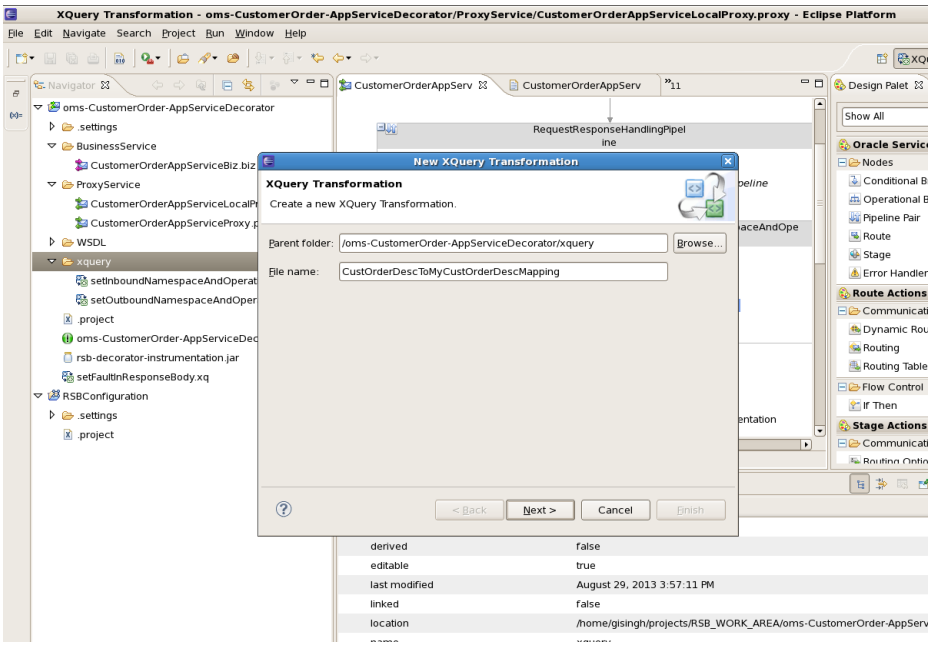
How to do Payload Transformation

The proxy service request message payload types may be different from the payload types that are required by the new business service WSDL. Therefore we need to transform the incoming request payload to the format expected by the business service. For payload transformation, follow the steps as shown below:

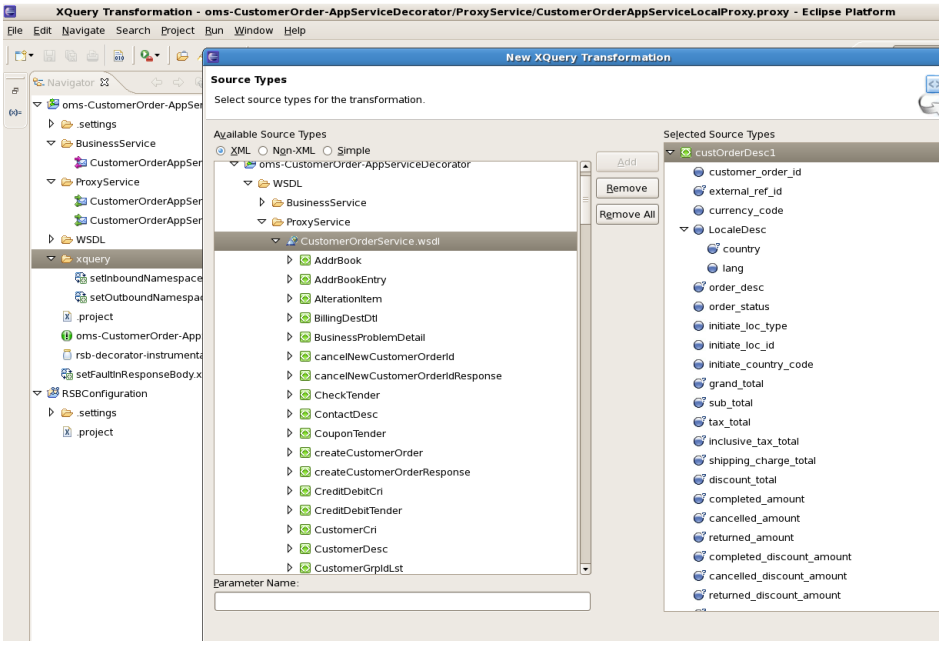
1. In our example, the proxy service payload is of type CustOrderDesc and the business service payload is of type MyCustOrderDesc. So first we need to create xquery files which transform the payload from CustOrderDesc to MyCustOrderDesc type. Right-click the xquery folder and select **New > XQuery Transformation**.



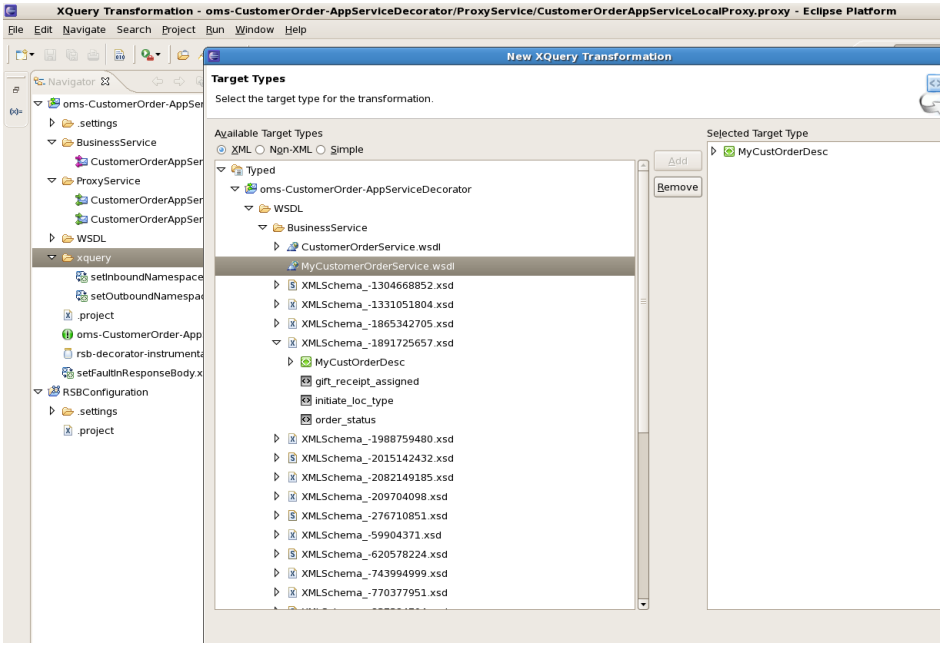
2. Enter file name as **CustOrderDescToMyCustOrderDescMapping**.



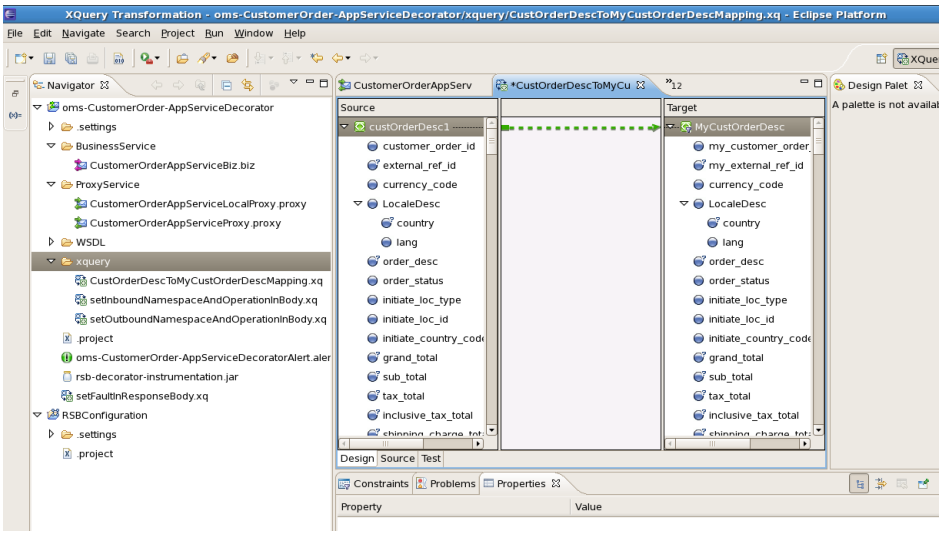
3. Click **Next**. Select CustOrderDesc element from the proxy service WSDL which is CustomerOrderService.wsdl and click **Add**. Verify that it gets added in the Selected Source Types window.



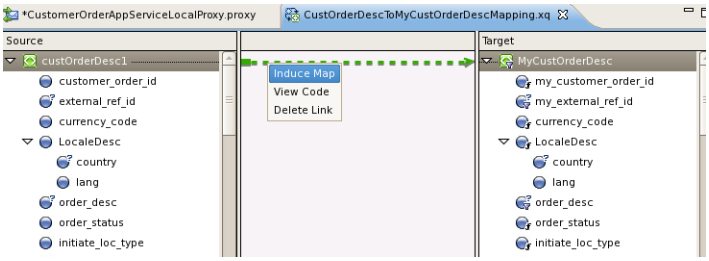
4. Click **Next**. For Target Types, select MyCustOrderDesc from the new business service WSDL which is MyCustomerOrderService.wsdl. Click **Add**. Verify that it gets added in the selected Target Type window.



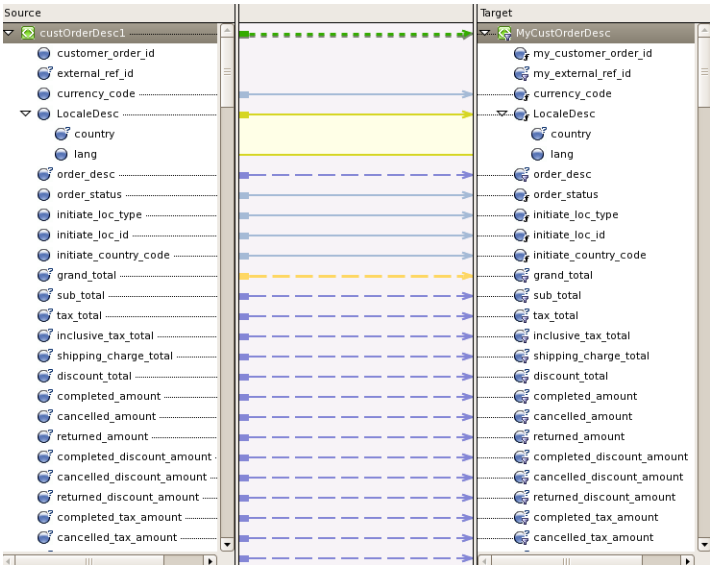
5. Click **Finish** and go to the Design tab of xquery file. In the Design view of the xquery file, you can drag and map the elements from source to target type. So drag from CustOrderDesc and connect to MyCustOrderDesc:



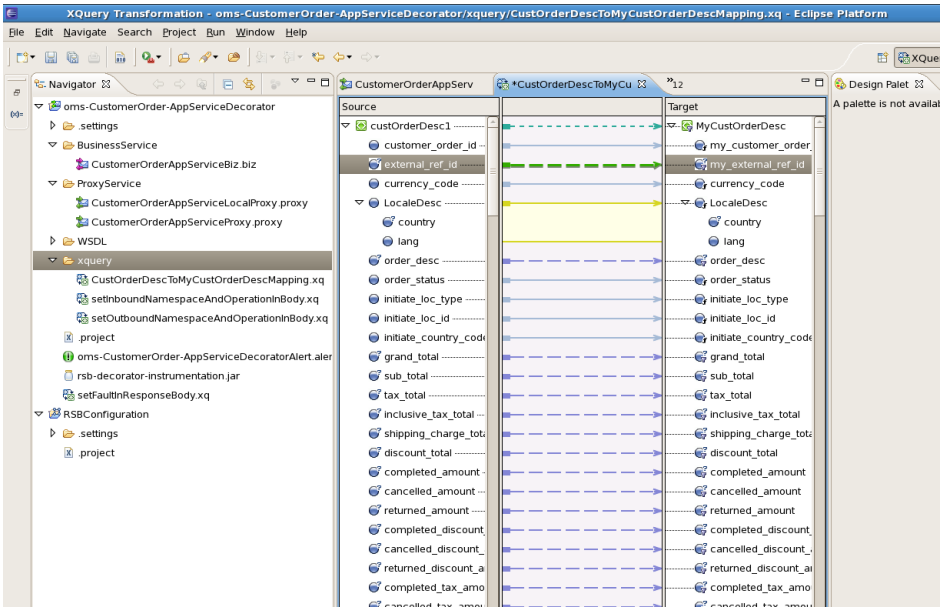
6. Right-click the link between the source and target types and select Induce Map. It will auto map all the fields which can be mapped automatically.



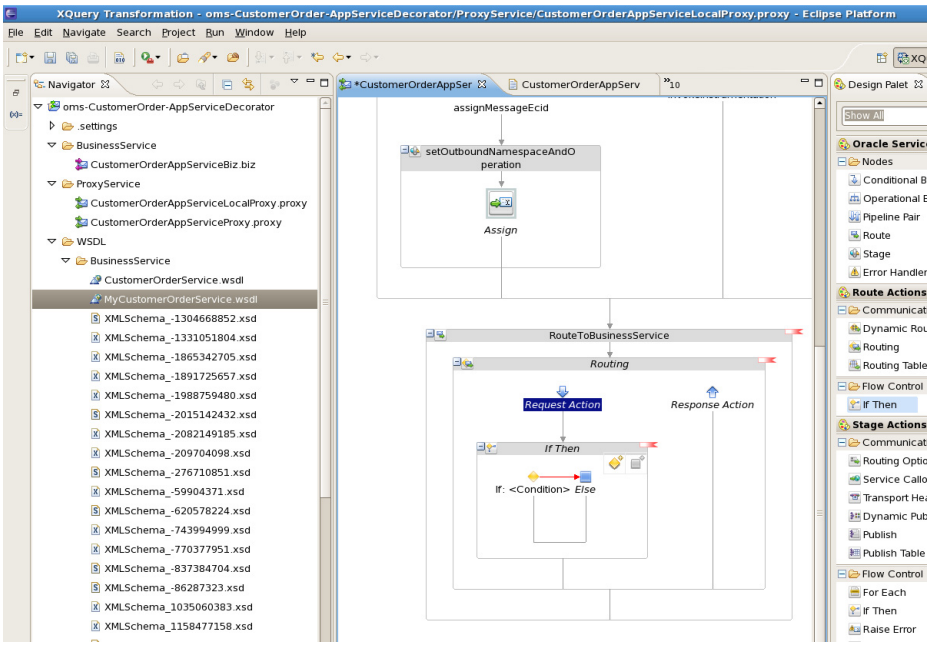
7. After mapping it looks like below:



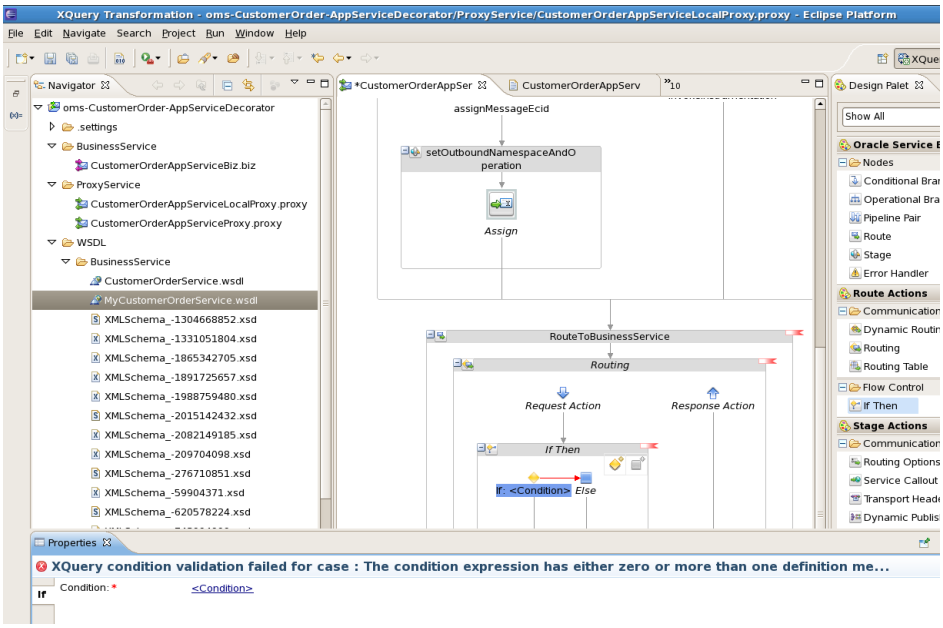
8. For the fields which are not auto-mapped, we need to map them manually. Drag and connect those fields one by one. You may have to write xquery functions for complex mapping. This document will not go into the details of mapping using XQuery. Please refer to XQuery documentation for more details. Following is the screenshot after mapping fields:



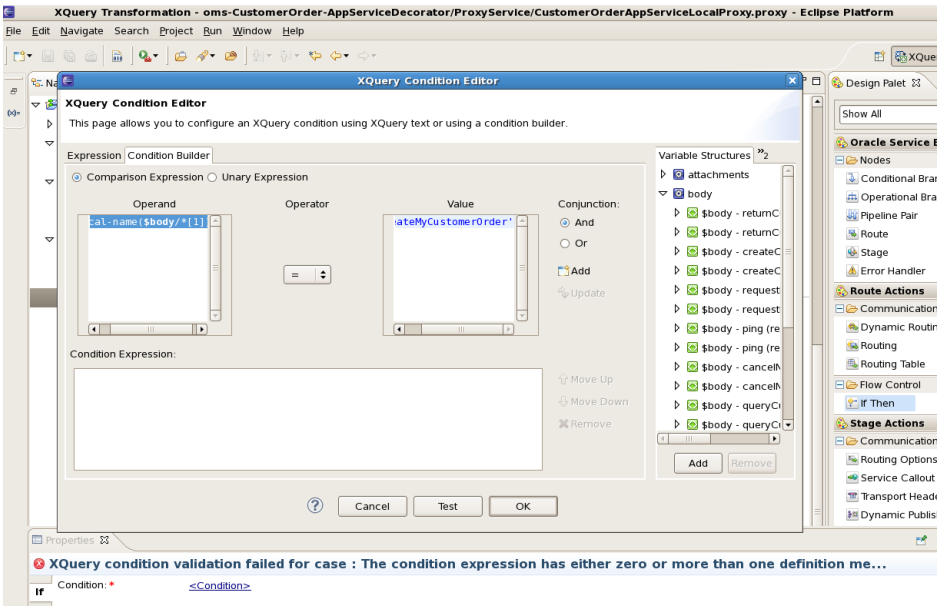
9. Once the mapping in xquery file is complete, go to the Routing node and in the Request pipeline, right-click Request Action and select **Insert Into > Flow Control > If Then to add an If Then flow**:



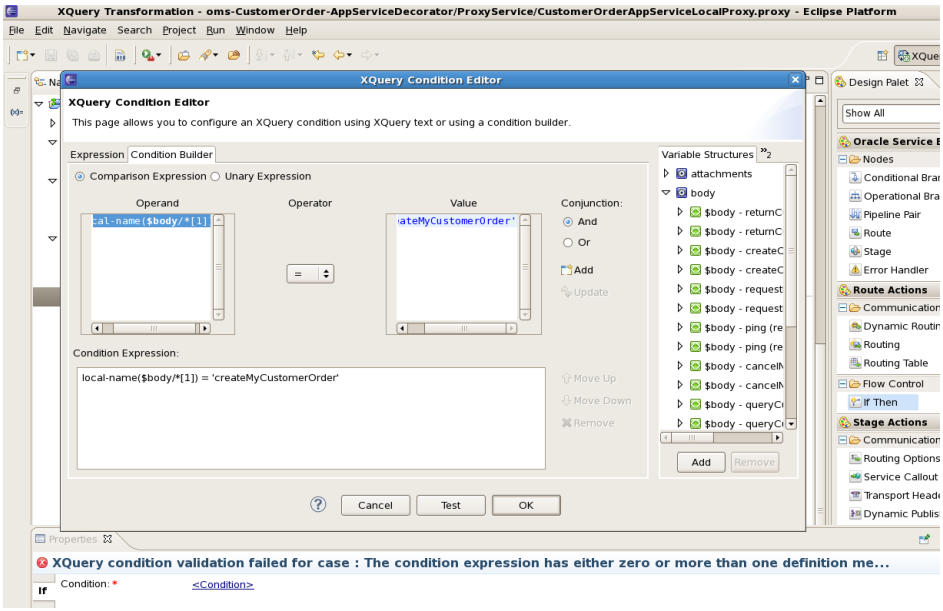
10. Go to the Properties window of first If Condition.



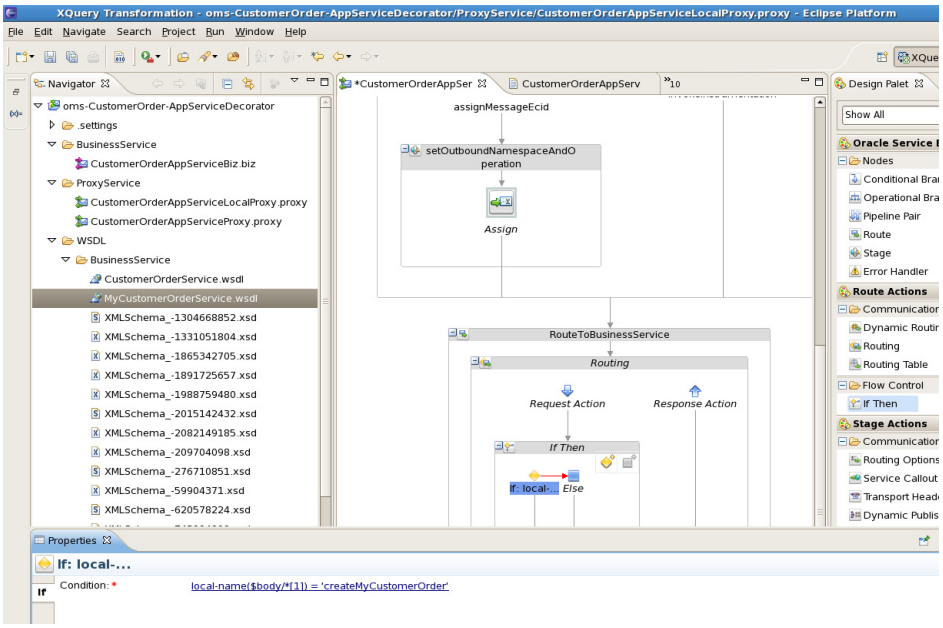
11. Click the **<Condition>** link and go to **Condition Builder** tab. Here we need to build the condition for payload mapping. We will check for operation name to build the condition. Enter local-name($body/*[1]) in the Operand field, which gives the operation name in request xml. Select **=** in the Operator field. In the Value field, we need to enter operation name which we want to look for, enter 'createMyCustomerOrder'.



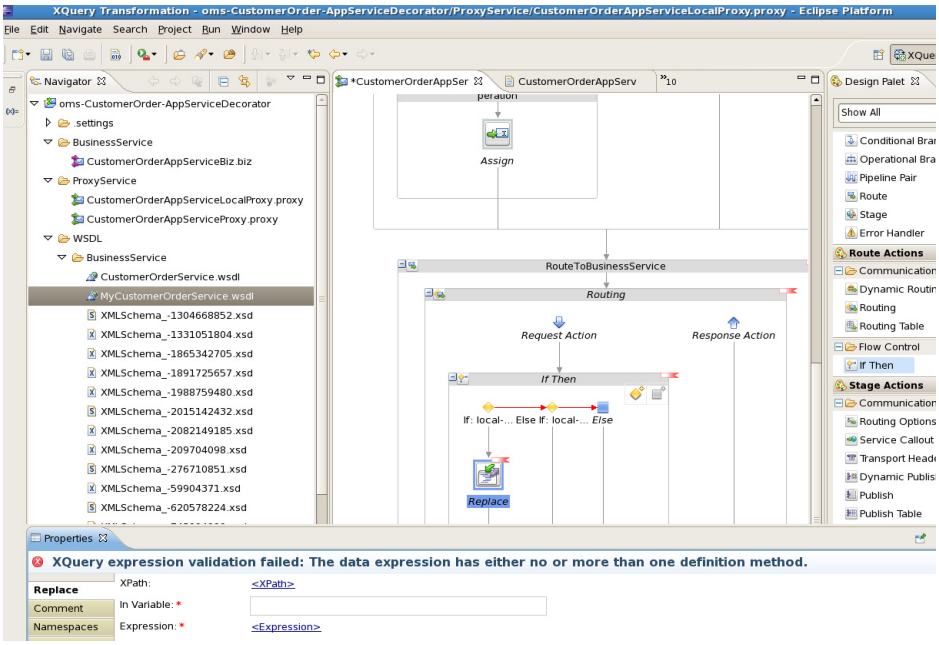
12. Click **OK** to add condition to Condition Expression.



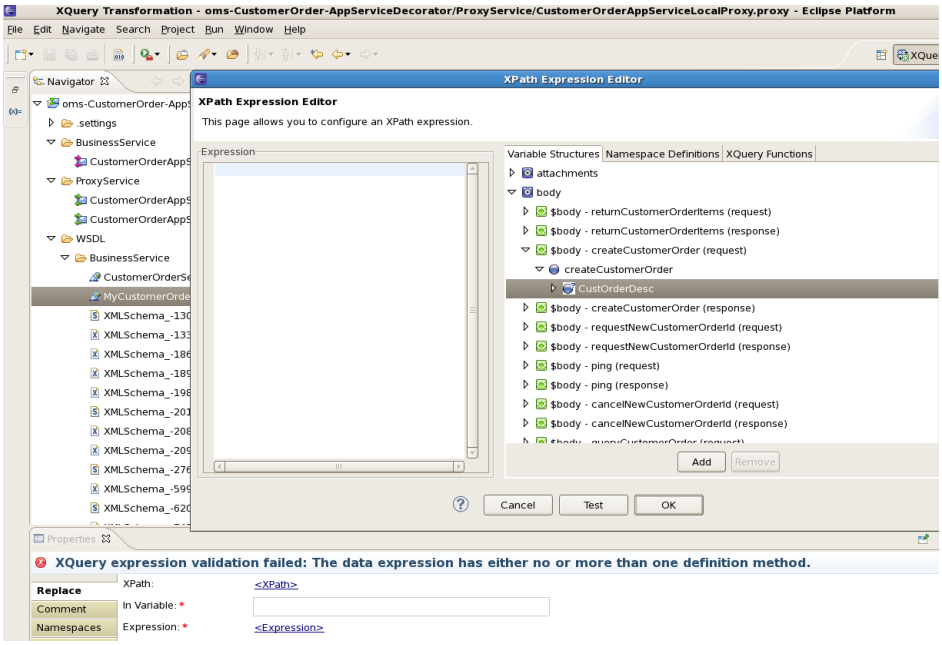
13. Click **OK**.



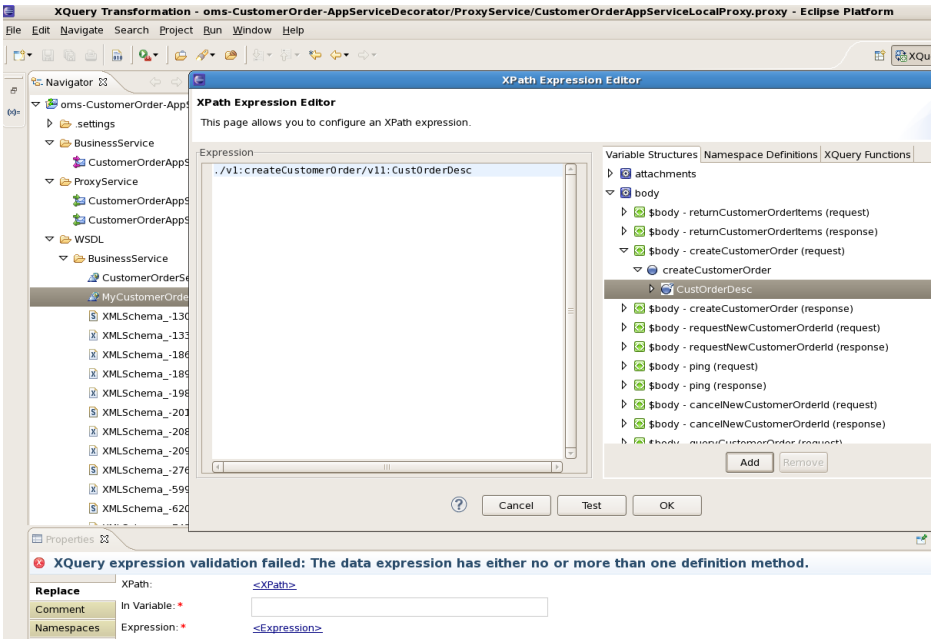
14. Add a Replace action for the first condition



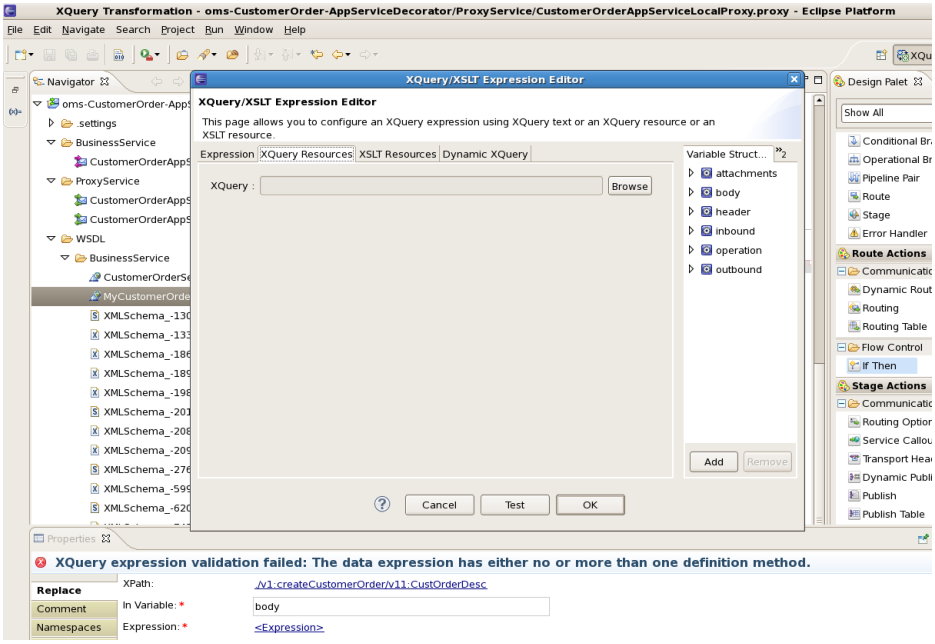
15. Click the **<XPath>** link of Replace action. Here we need to provide xpath of the variable which we need to transform.



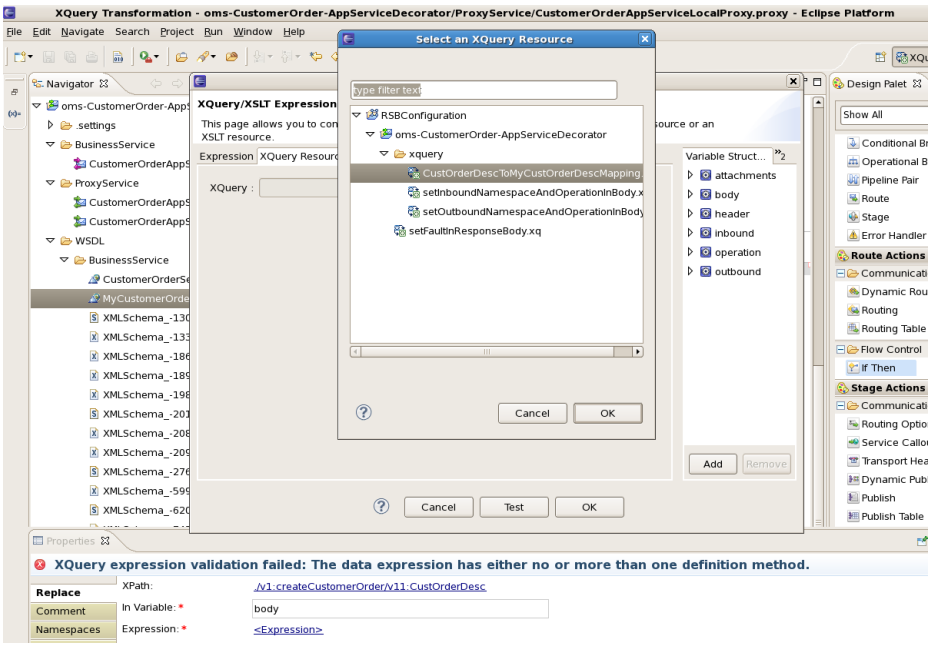
16. Enter the xpath expression.



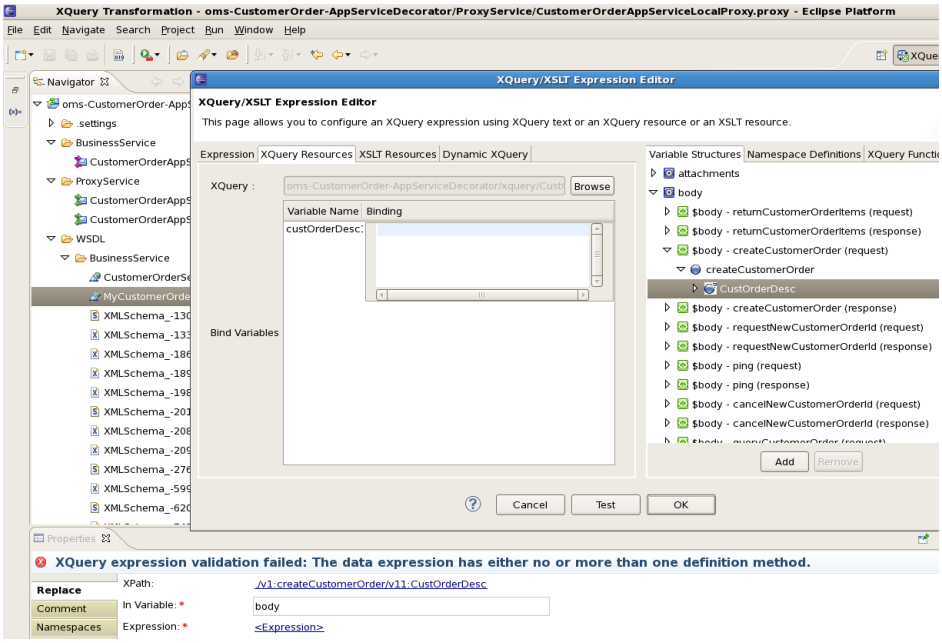
17. Click **OK** to return to the main window. Then click the **<Expression>** link. Here we need to provide the xquery which will return the transformed payload.



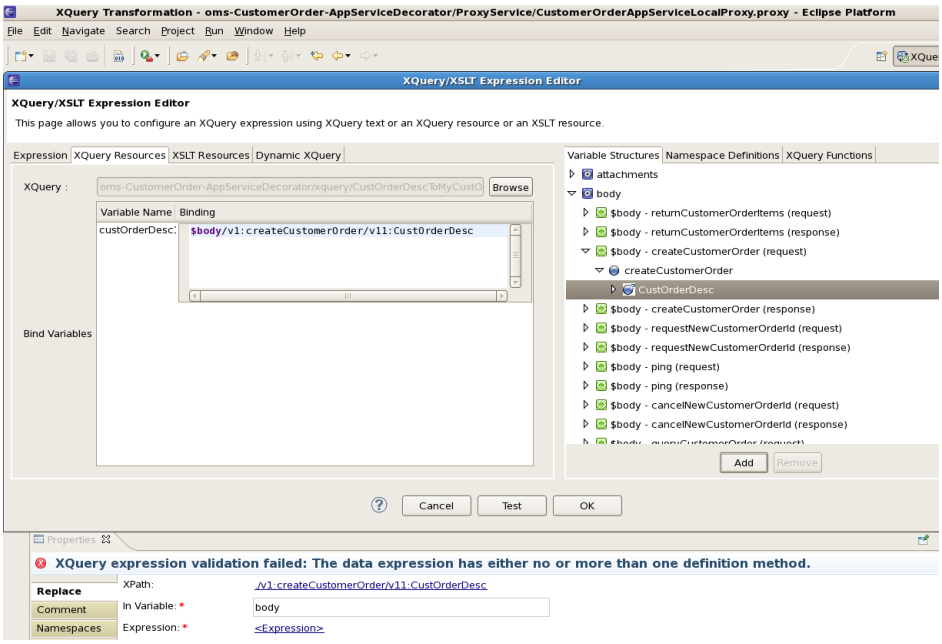
18. Click **Browse** and select the xquery file.



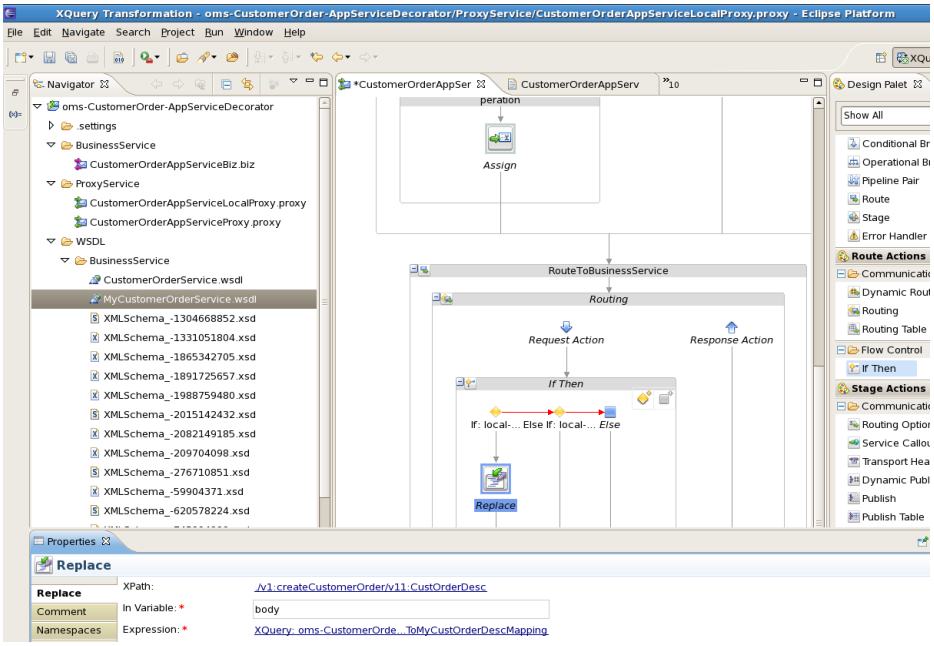
19. Click **OK**.



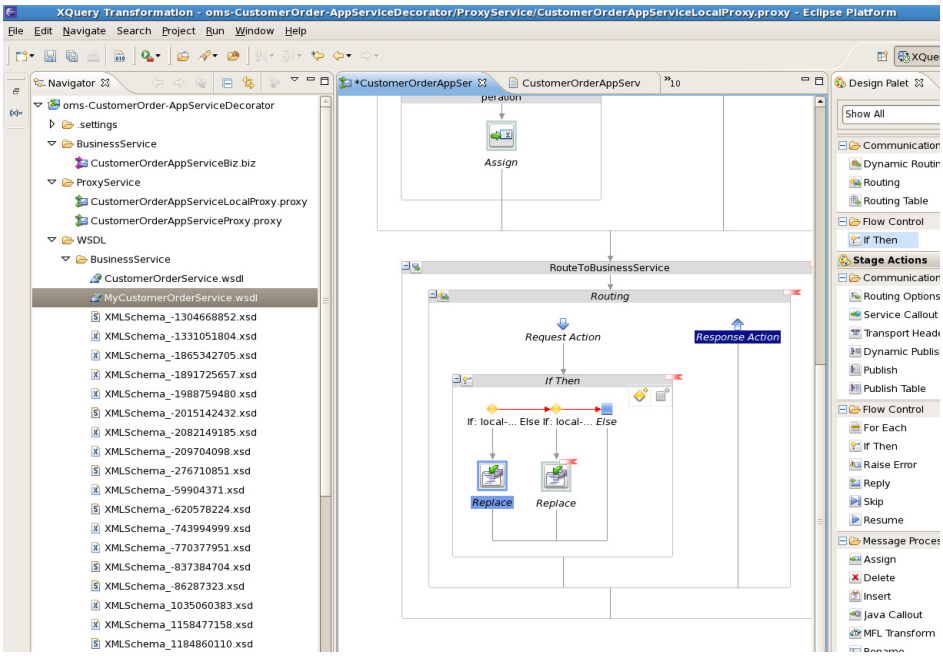
20. In the Binding field, we need to provide path of the input payload which needs to be transformed. Select **CustOrderDesc** for this example.



21. Click OK

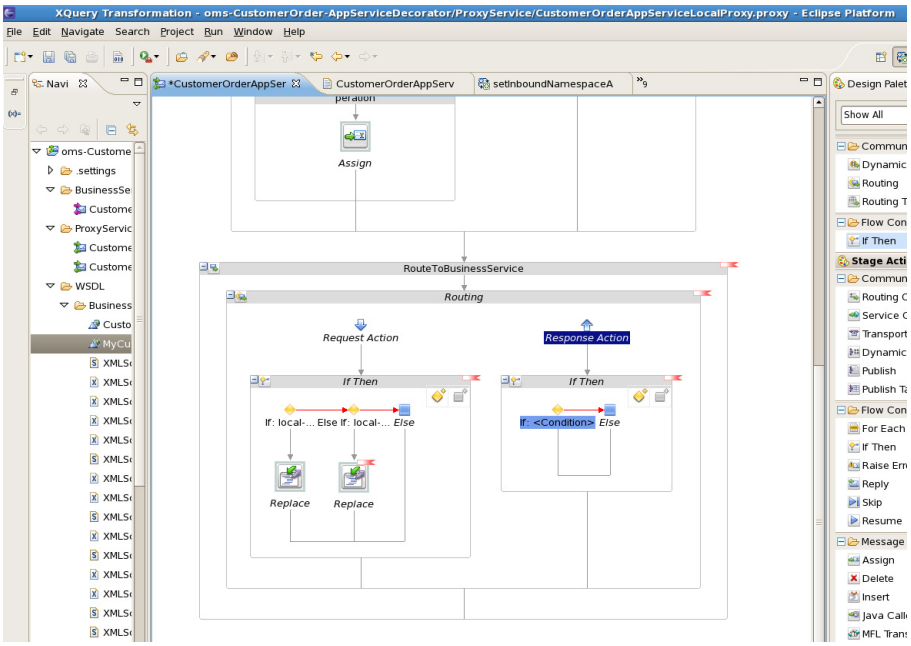


22. More Replace actions can be added in other if conditions to transform payload for each operation type. You will need to write xquery files for each input payload to output payload transformation.

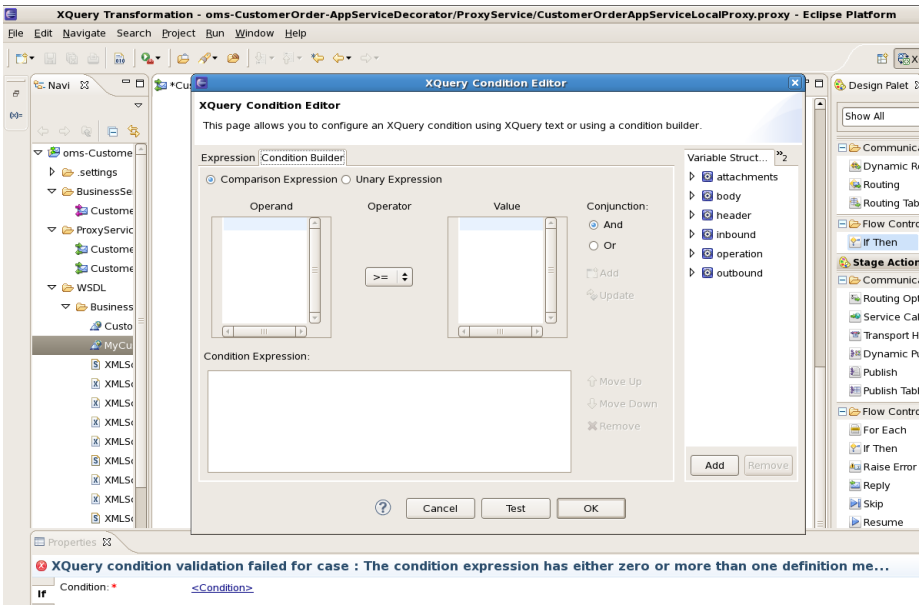


23. We need to do similar transformation in reverse order in Response pipeline as well which will transform MyCustOrderDesc to CustOrderDesc. So first we need to write an xquery file which transforms MyCustOrderDesc to CustOrderDesc. The steps for creating xquery file will be exactly similar to the steps for creating CustOrderDescToMyCustOrderDescMapping file. The only difference will be that the source and target types will be reversed in this case. Now the source type will be MyCustOrderDesc and target type will be CustOrderDesc. Save the new file as MyCustOrderDescToCustOrderDescMapping.

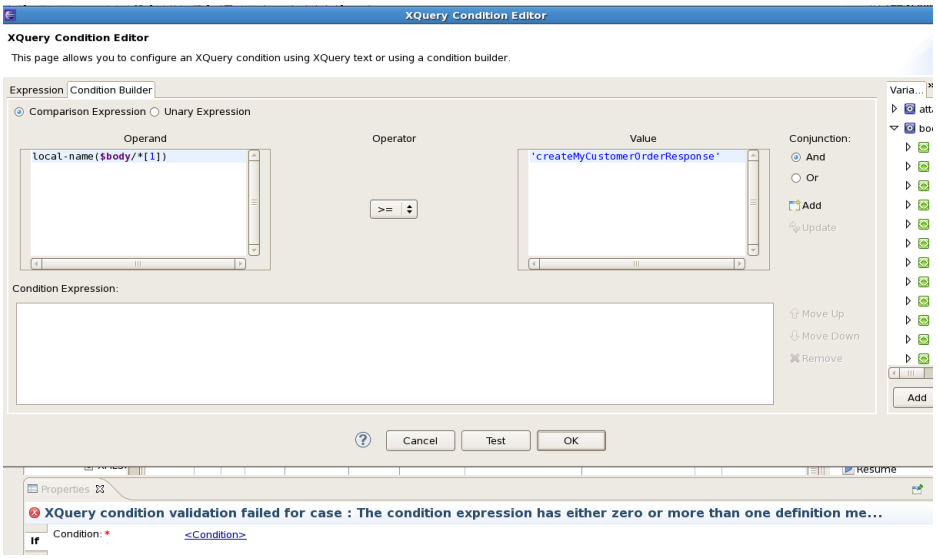
    After creating the xquery file, add an If Then flow in Response Action as shown below:



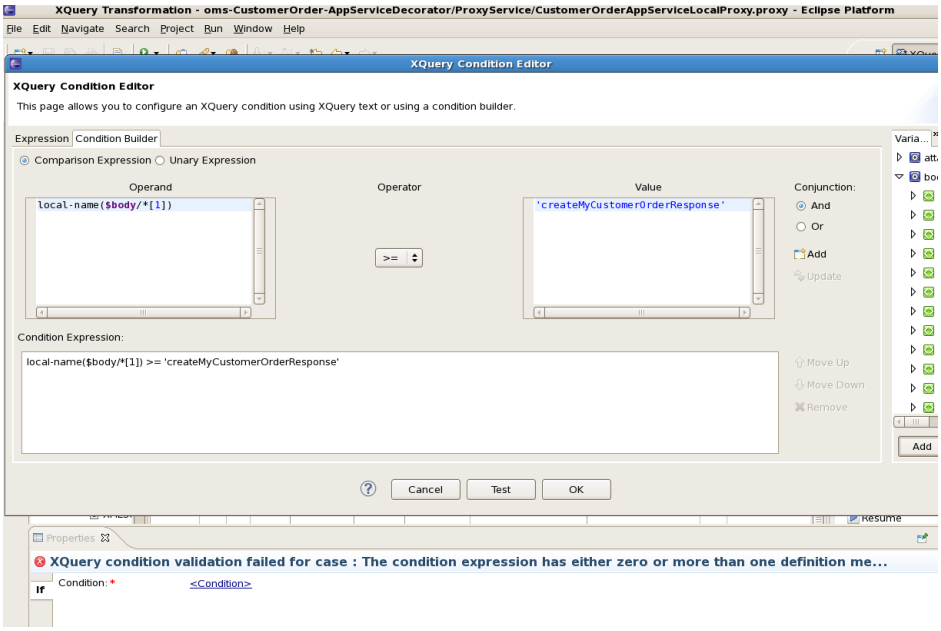
24. Click **<Condition>** link for first If condition and go to the **Condition Builder** tab.



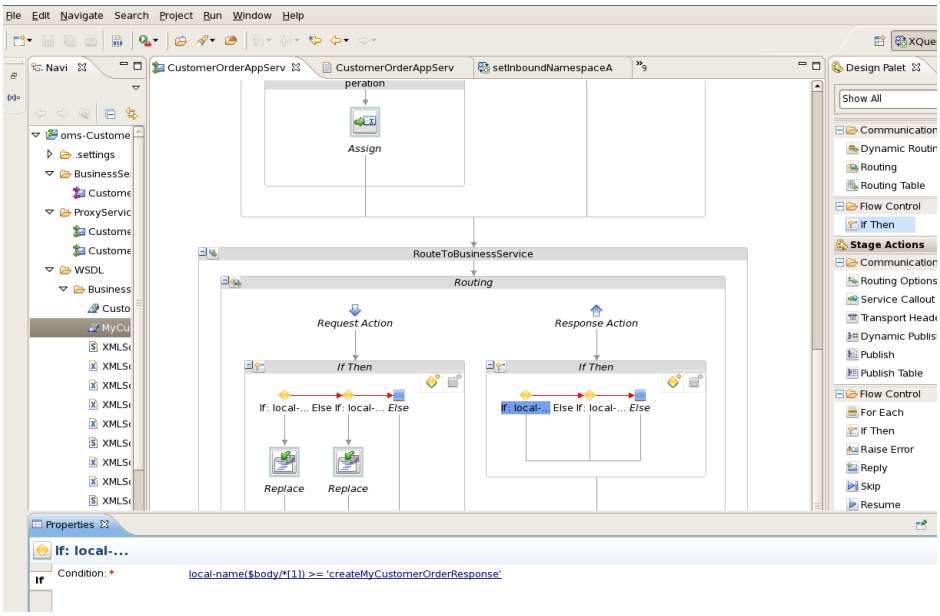
25. Enter the values in Operand, Operator and Value fields as show below:



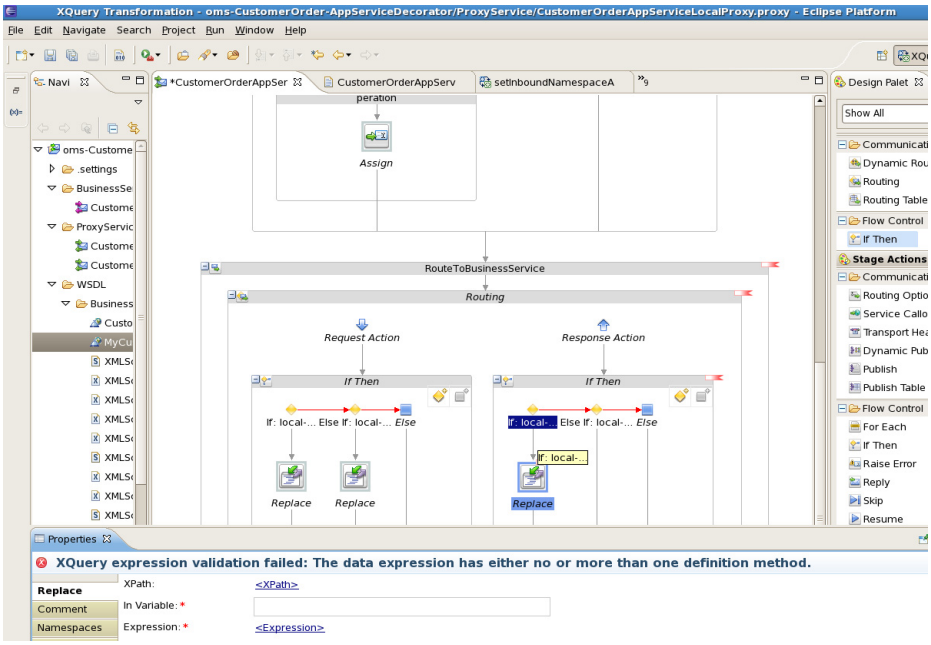
26. Click **OK** to add condition to Condition Expression window.



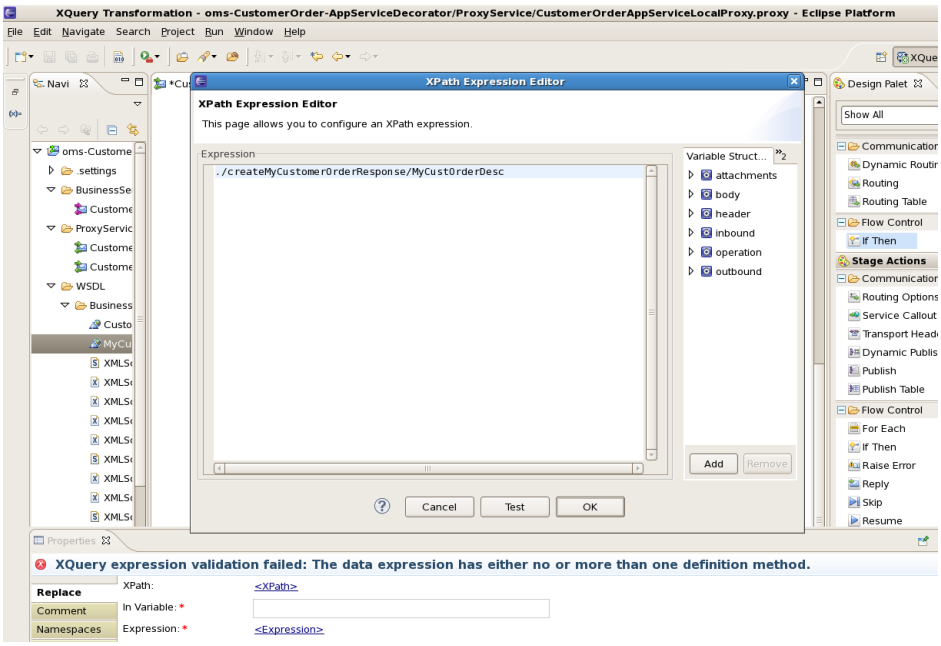
27. Click **OK** to go back to properties window.



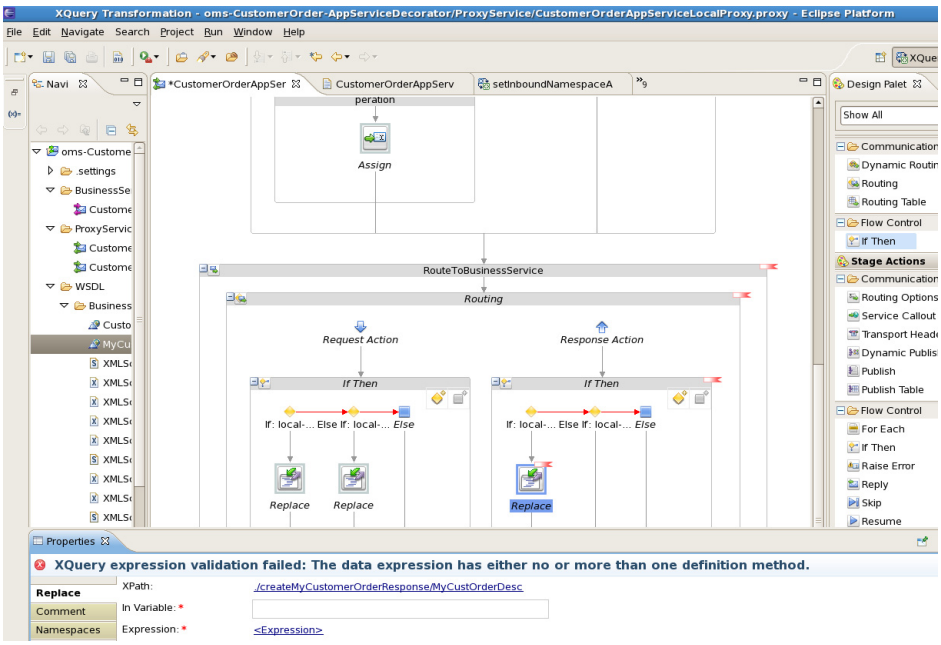
28. Add a Replace action in first if condition flow:



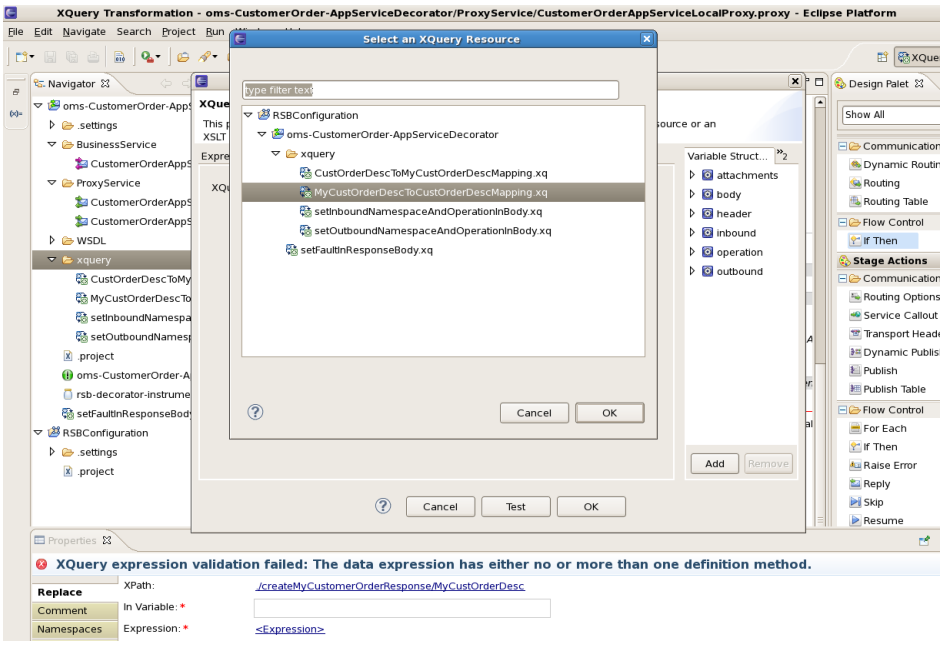
29. Click the **<xpath>** link. Here we need to enter the xpath of the element which needs to be transformed. Enter as shown below:



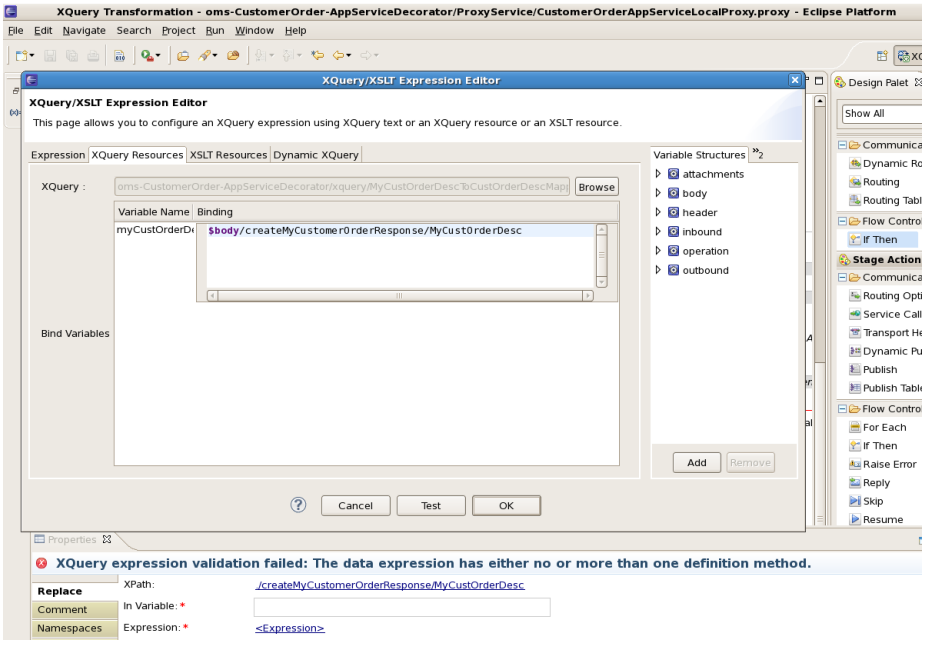
30. Click **OK**.



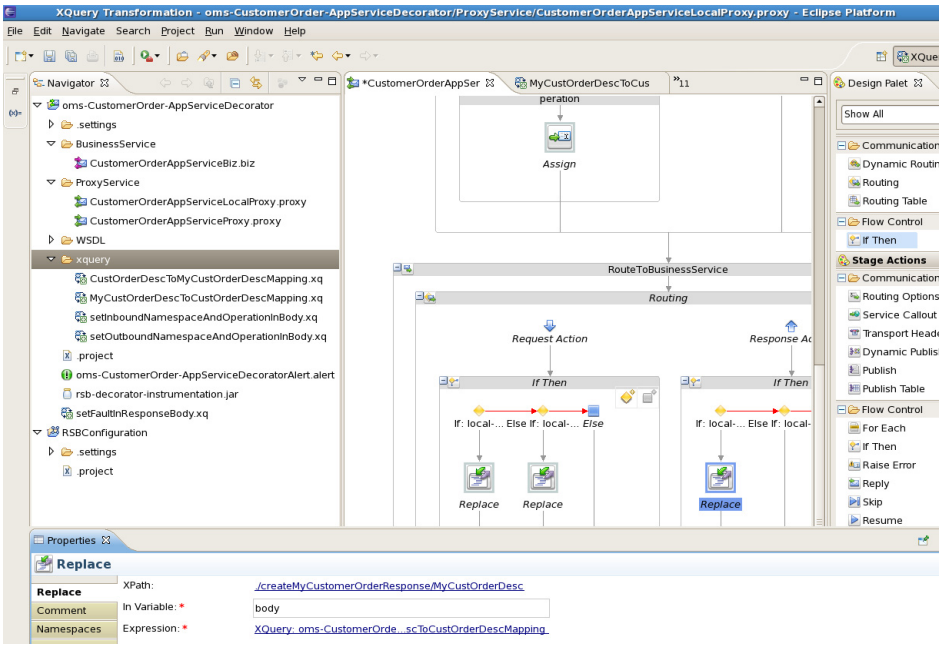
31. Click the **<Expression>** and go to XQuery Explorer window to select the xquery file. Select **MyCustOrderDescToCustOrderDescMapping** xquery file.



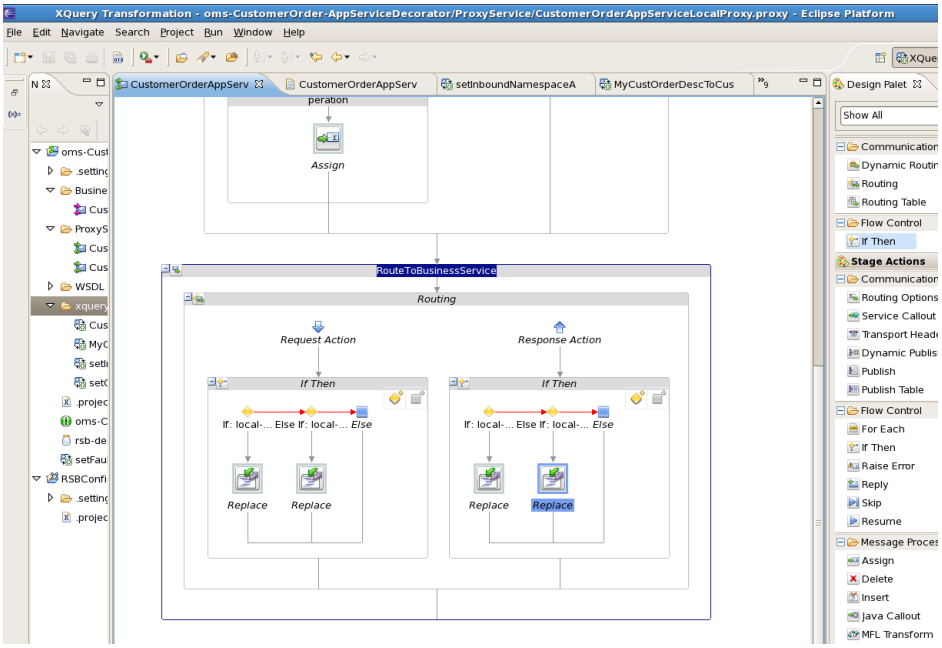
32. Click **OK**. In the binding field enter path to MyCustOrderDesc as that is the source payload for transformation:



33. Click **OK**. In Variable field enter body.



34. Add more replace actions for other if conditions in the if then flow:



This completes the steps for payload transformation in a message flow.

4

Introduction to Alerts

Alerts are generated by OSB monitoring framework and they help to diagnose problems when they occur. Oracle Service Bus provides two types of alerts:

- SLA Alerts
- Pipeline Alerts

Pipeline/Business Alerts

Pipeline alerts can be raised in the message flow of the proxy service. You can use the alerts in a message flow for:

- Detecting business errors in a message flow.
- Indicating business occurrences.

SLA Alerts

SLA alerts are raised in Oracle Service Bus to indicate potential violation of the Service Level Agreements (SLAs). You can use SLA alerts for:

- Monitoring and generating email notification of WS-Security errors
- Monitoring the number of messages passing through a particular pipeline
- Detecting the violation of service level agreements with third-party products
- Detecting a non-responsive endpoint

Consider the following use case to verify the service level agreements:

Assume that a particular proxy service is generating SLA alerts due to slow response time. To investigate this problem, you must log in to the Oracle Service Bus Administration Console and review the detailed statistics for the proxy service. At this level, you can identify that a third-party web service invocation stage in the pipeline is taking a lot of time and is the actual bottleneck.

You can use these alerts as the basis for negotiating SLAs. After successfully negotiating SLAs with the third-party web service provider, you should configure alert metrics to track the web service provider's compliance with the new agreement terms.

There are different ways to add SLA alerts and Business alerts in RSB decorator projects. SLA alerts can be added from OSB console after the decorator has been deployed in a OSB server. Pipeline alerts can be added from either OEPE or from OSB console. It is recommended to add pipeline alerts in the OEPE and then export the decorator jar.

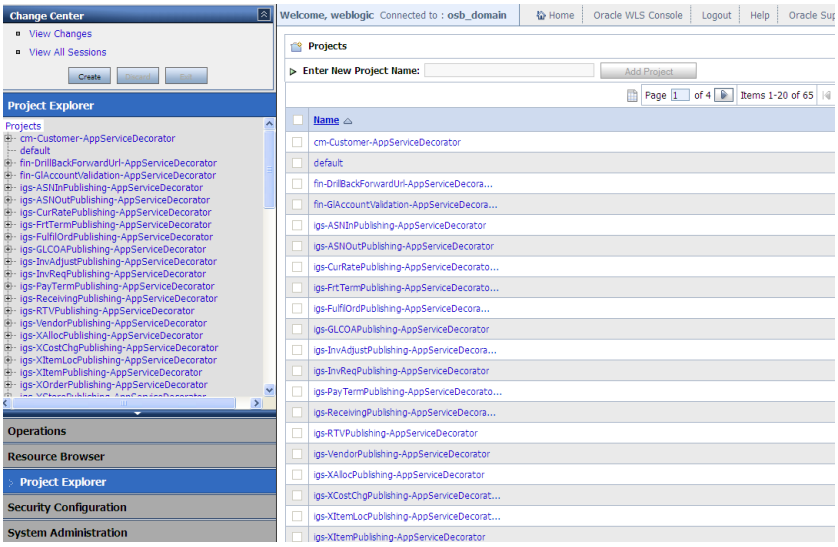
Default Alerts in RSB Decorator Projects

RSB decorator jars have a default SLA alert configured for each proxy and business service. The default alert rule name is ErrorCountRule. This alert is configured to generate an SLA alert whenever an error condition occurs in the message flow. This is just a sample SLA alert. It is recommended to delete this rule and create a new rule for the actual SLA criteria for that environment.

How to add new SLA alert

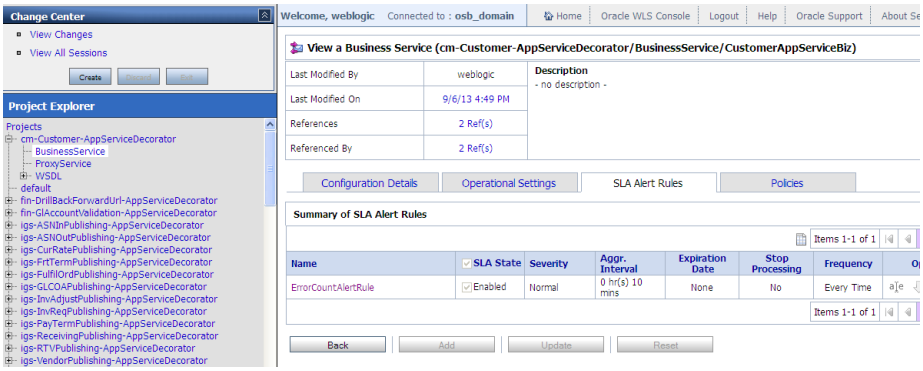
SLA alerts are operational settings and they can be added or modified only from OSB console. Follow the steps to delete the default alert rule and add a new rule:

1. Log in to OSB console and access the Projects page.

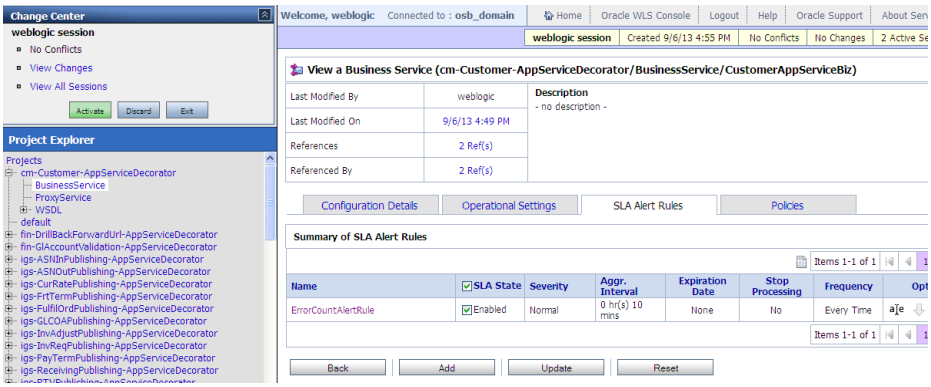


2. Click the project for which you want to modify SLA alert.

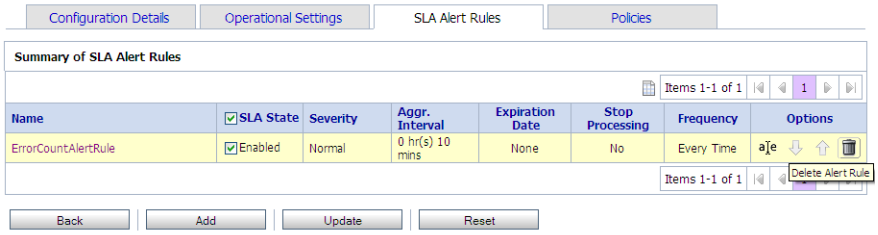
   The steps to add or modify an SLA alert are same for both business service and proxy service. In this example, we will show steps for a business service. Browse to the business service and go to SLA Alert Rules tab of that service.



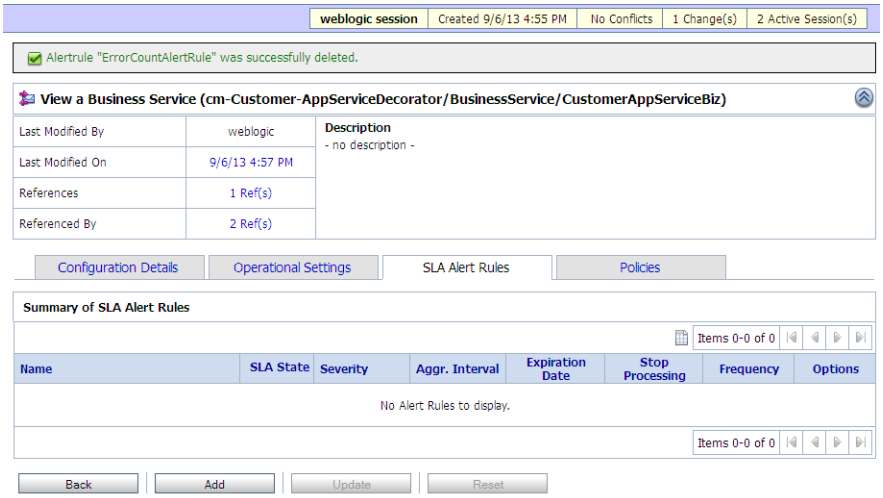
3. Click **Create** in Change Center to create a new session.



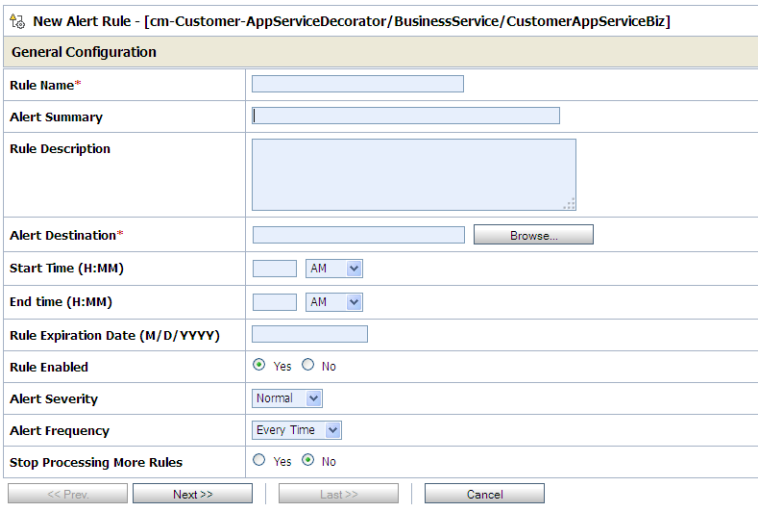
4. Click **Delete** in the alert rule row.



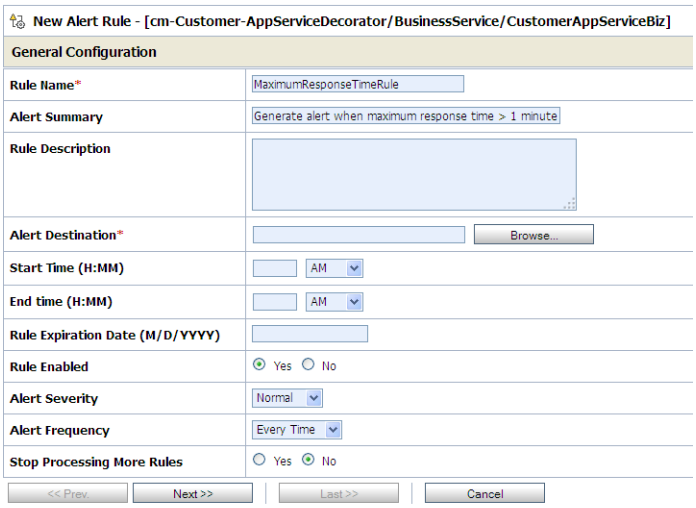
5. The rule is deleted.



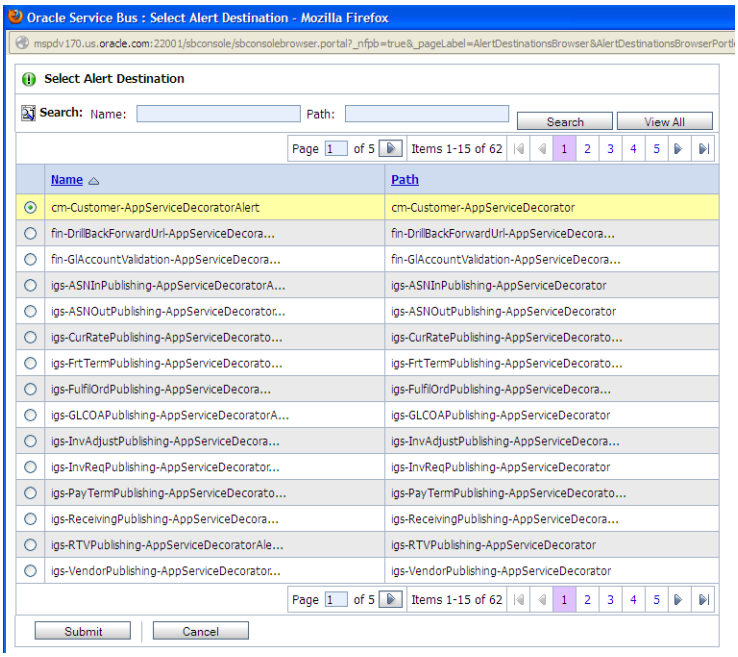
6. Click **Add** at the bottom to add a new rule.



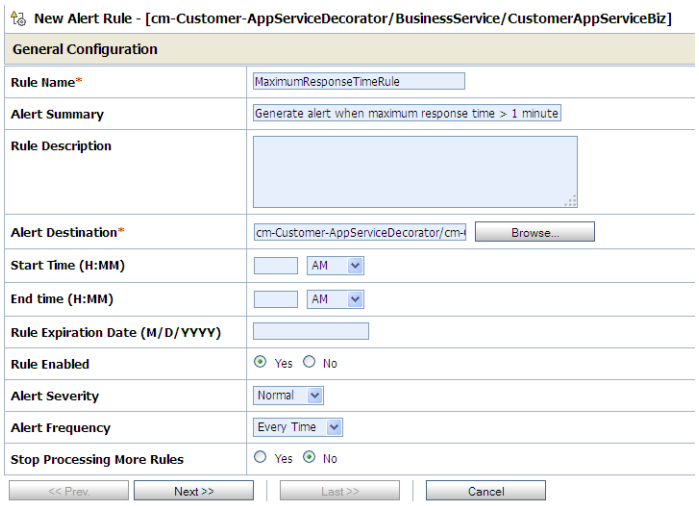
7. Enter appropriate values for Rule Name and Alert Summary fields. It is recommended to have a good summary of why this alert rule should be generated. Having proper description of all fields will be useful when looking at rules in RIC console and it will help better in diagnosing the issues.



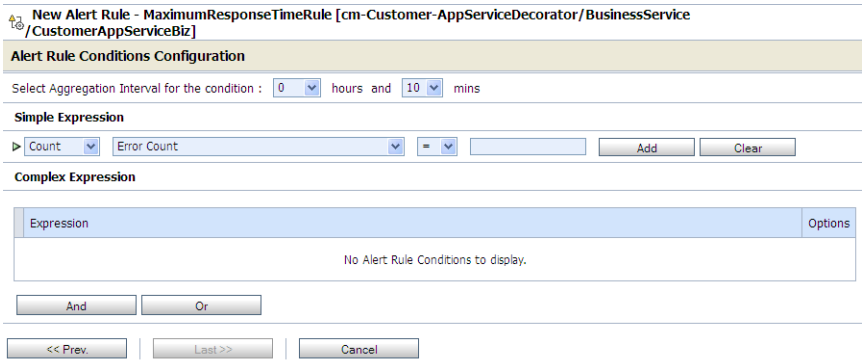
8. For Alert Destination, click the **Browse** button. It shows the list of all alert destinations in the OSB server. You can select the default alert destination or create a new destination and choose that.



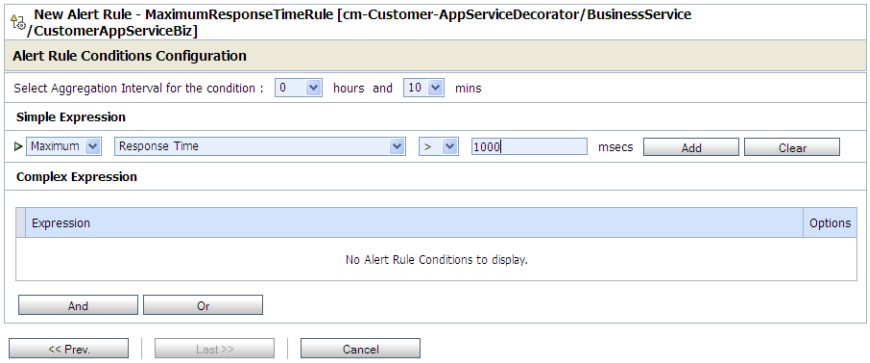
9. Click **Submit** to select the alert destination.



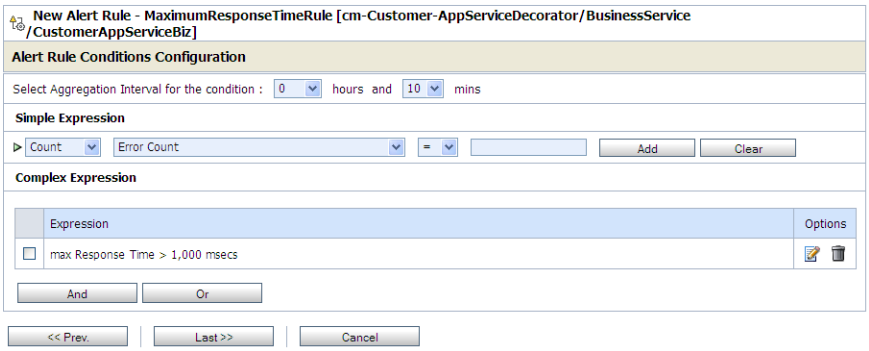
10. Click **Next**. In the next page you can build the expression which defines the criteria for alert rule.



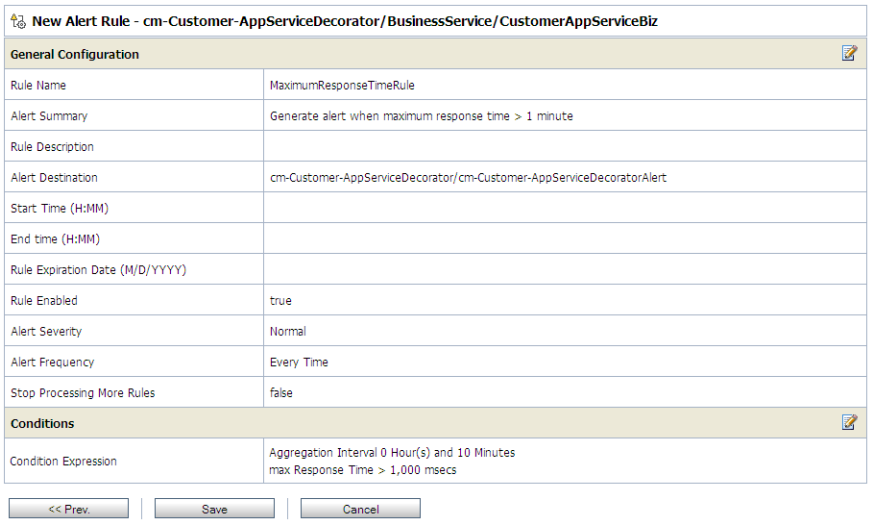
11. Now in the Simple Expression area, you can build the expression for which you want to generate SLA alert. In this example, we will build an expression to generate an SLA alert whenever the response time of business service is more than one second.



12. Click **Add**.



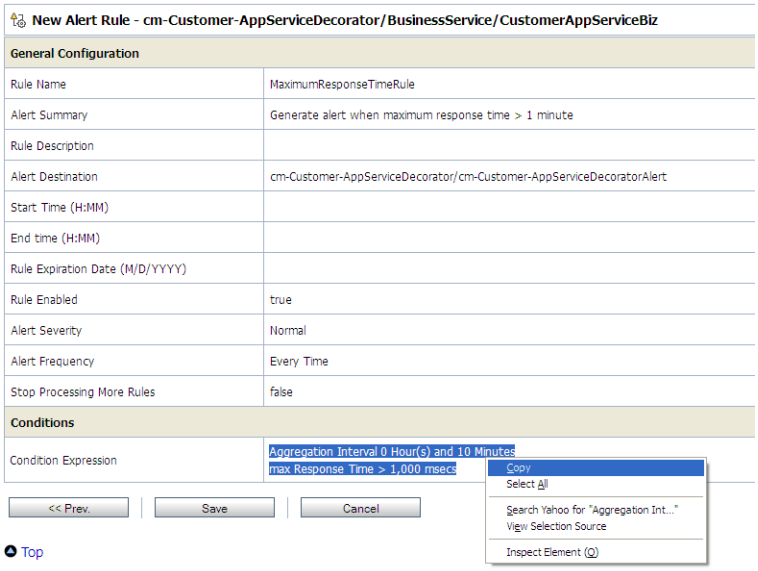
13. You can build more complex expression using And and Or buttons to build the rule. After the rule expression is built, click **Last**.



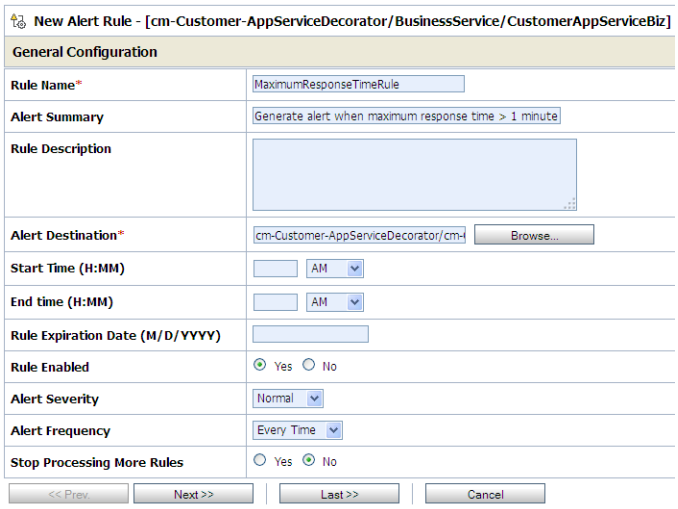
14. In the above page, you will see all the details about the new rule. Note that the Rule Description field is empty. We recommend that the condition expression is copied into the description field as well, so that when the SLA alert is displayed in the RIC, the exact condition of causing the alert is also displayed. The condition expression is not available in RIC, but the description field is available. Therefore

we should have a good description for the alert. Follow the steps to copy the condition expression into the description field.

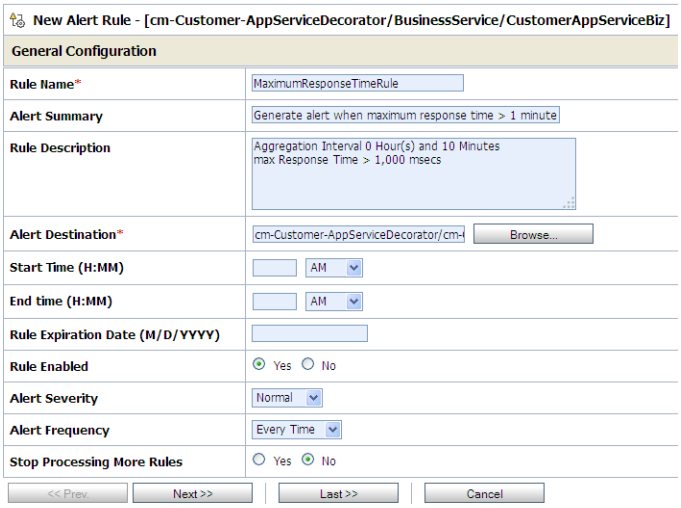
15. Right-click the expression string and select copy.



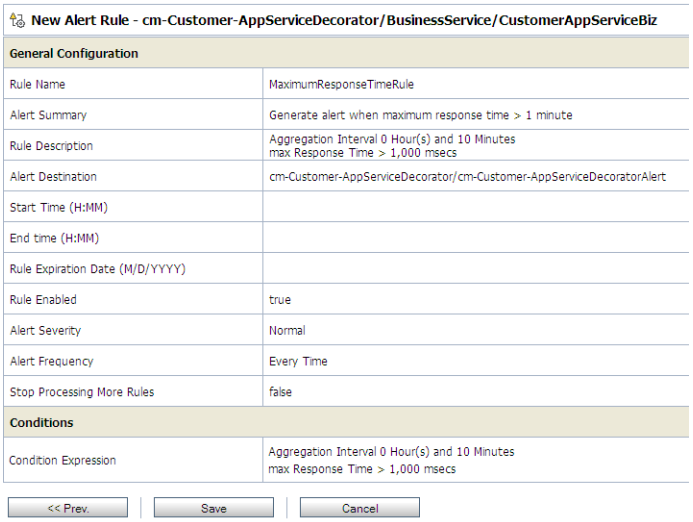
16. Click **Prev** button twice.



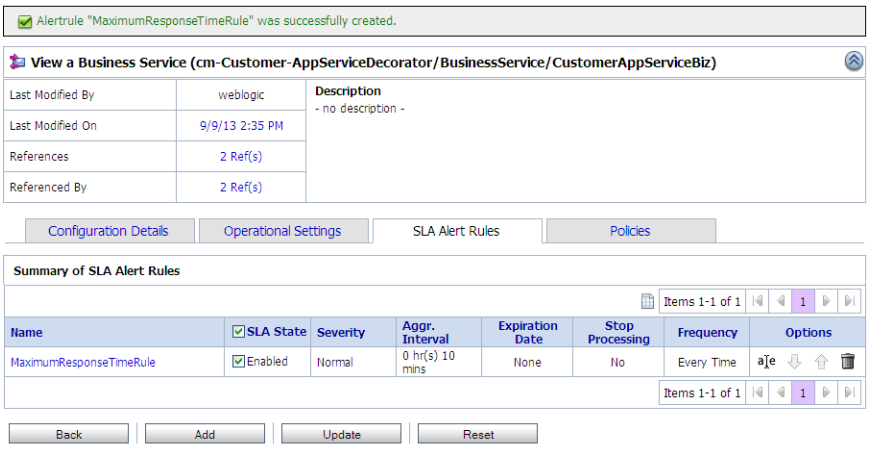
17. Paste the condition expression into the Rule Description field.



18. Click **Last>>** to go to last page.



19. Click **Save**.



20. Click **Activate** and then **Submit** to commit the changes in server.

This completes the steps to create new SLA alert rule.

> **Note:** SLA alerts are operational settings and can be added and modified only from OSB console. If the decorator jar is re-deployed on the server, remember that all the operational settings and SLA alerts will be lost. After deploying the new decorators, the SLA alerts will need to be created again.

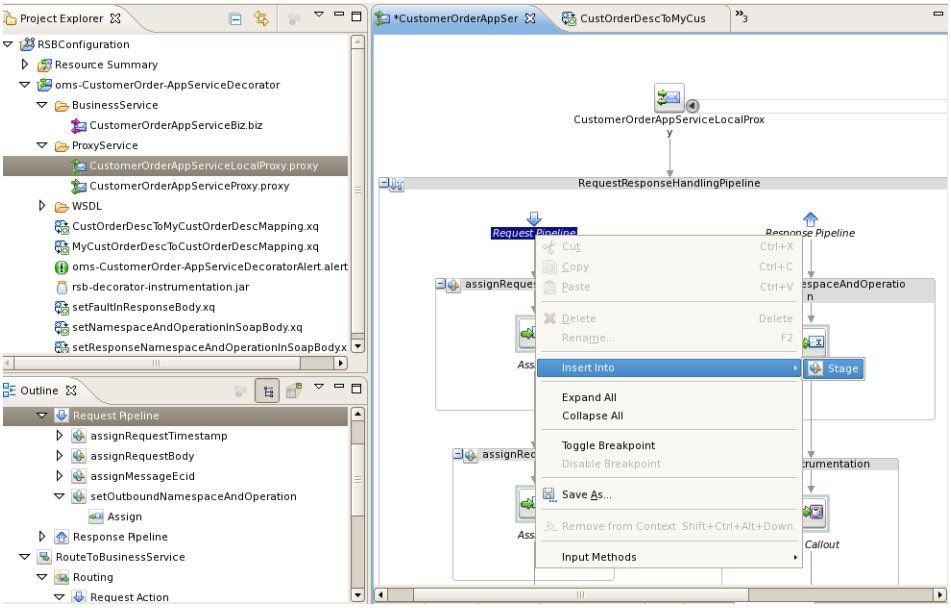
How to Add New Pipeline/Business Alert

Pipeline alerts are also called Business Alerts in RSB context. The reason for calling them business alerts is that they are used mostly to identify unusual business conditions or errors. For example, a customer may want to see an alert whenever a request is made with a very large amount.

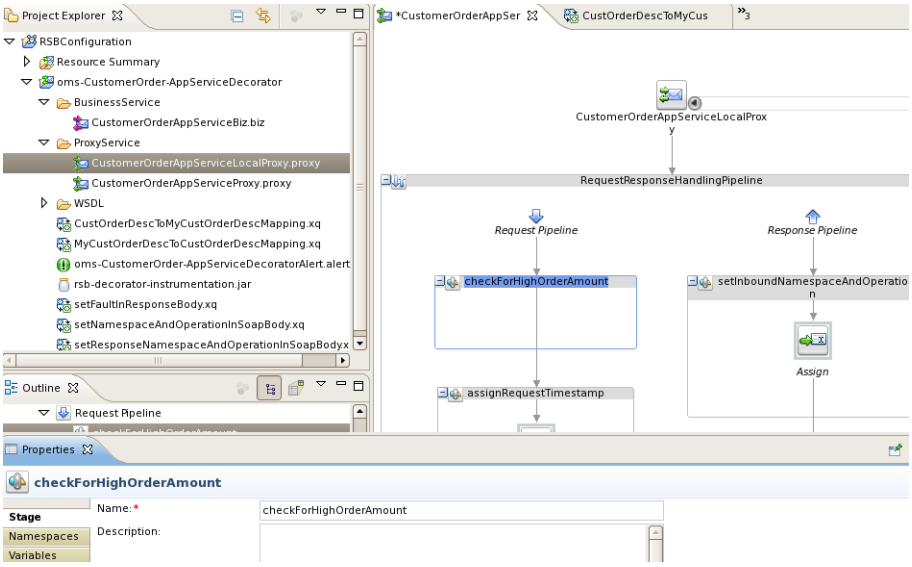
Pipeline alerts can be added in proxy services only. In RSB decorators, the message flow is defined in local proxy services. Therefore any new pipeline alerts should be added in local proxy service.

In this example, we will take an oms-CustomerOrderService where a business alert needs to be raised when grand total > 500000. The following steps are for adding a new pipeline alert:

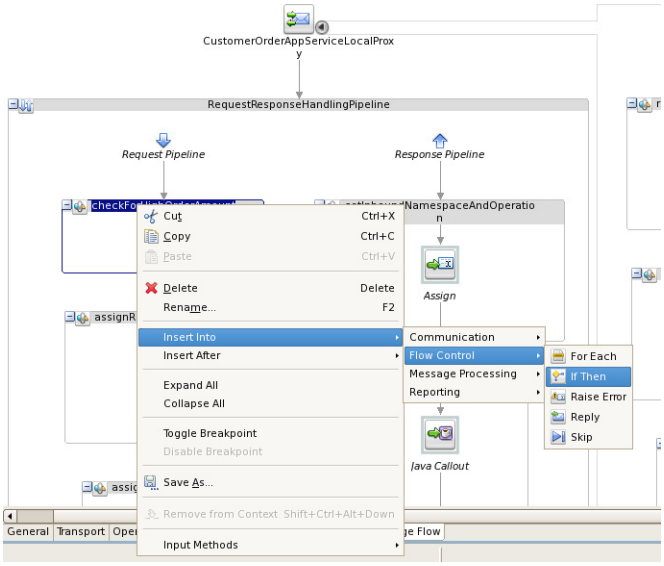
1. Right-click **Request Pipeline**, select **Insert Into >Stage** to add a new stage as shown below:



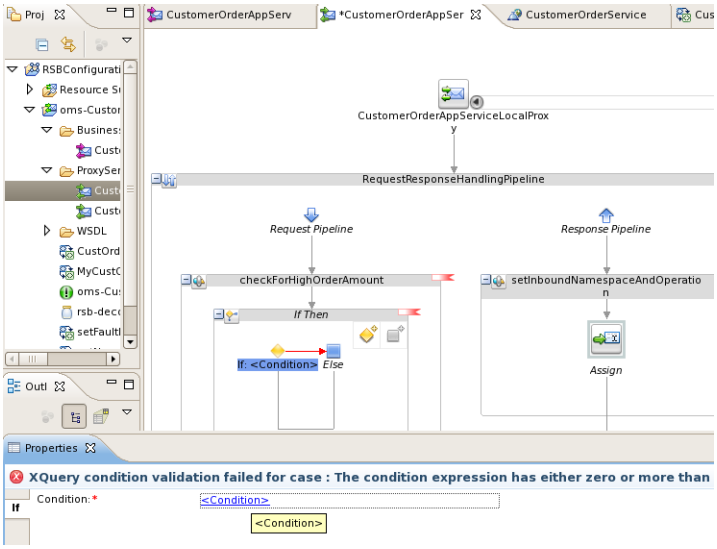
2. Enter appropriate name for the stage.



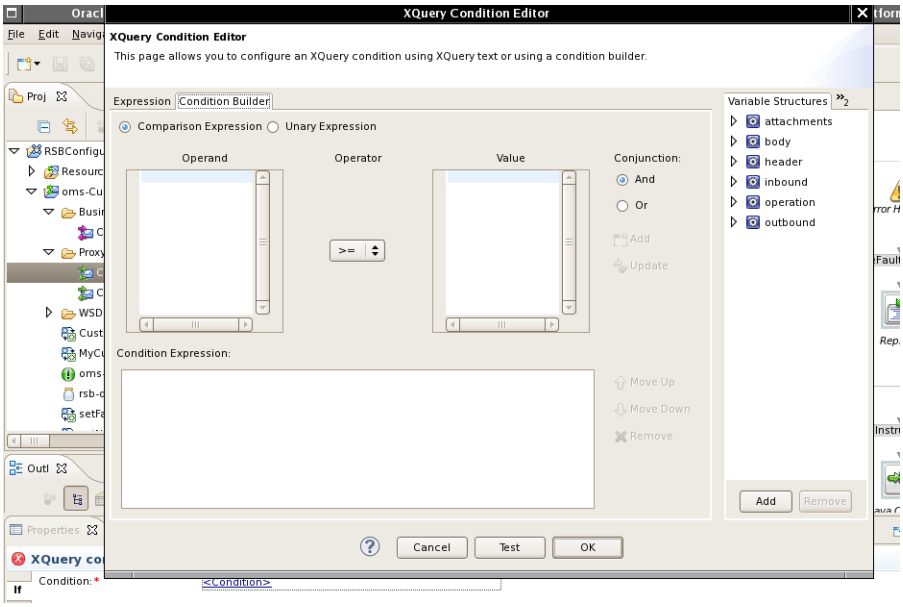
3. In the stage, add a new If Then flow, right click the stage added above, select **Insert Into> Flow Control > If Then** as shown below:



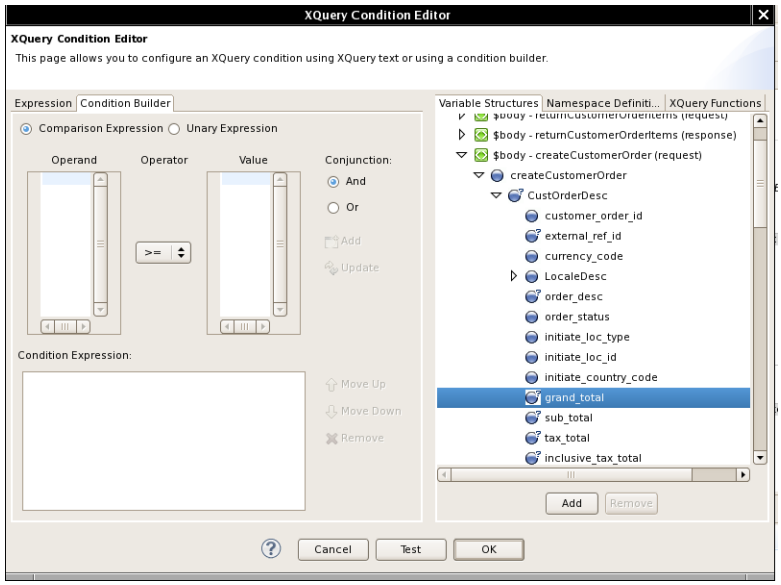
4. The If Then flow is added as shown below:



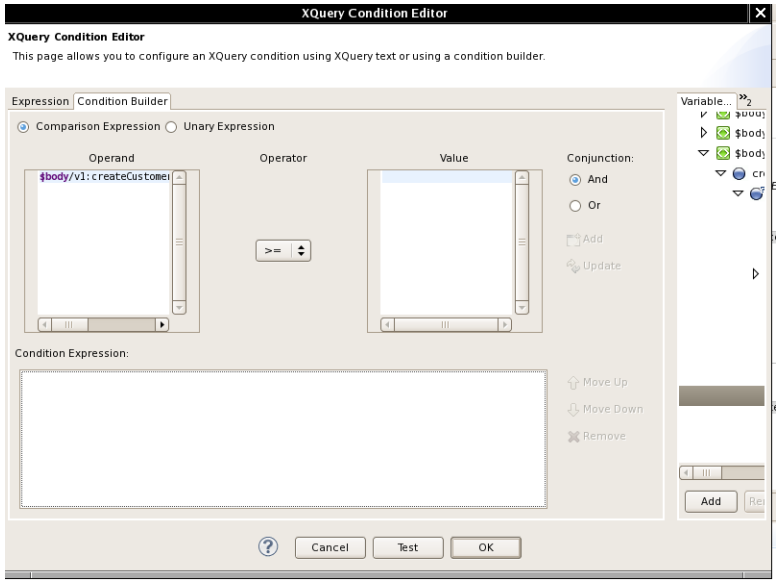
5. For the first If condition, click the **<Condition>** link and access **Condition Builder** tab.



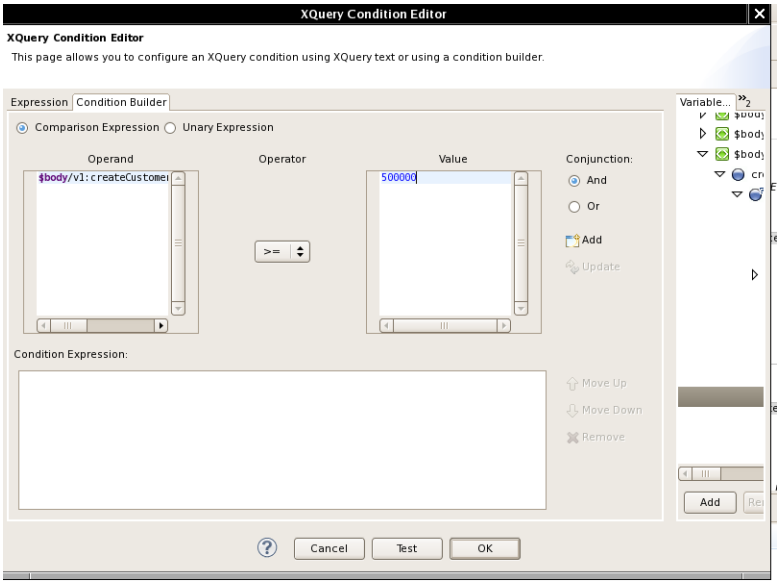
6. In this screen you can build the expression for the alert rule. Select the request schema element for the operation:



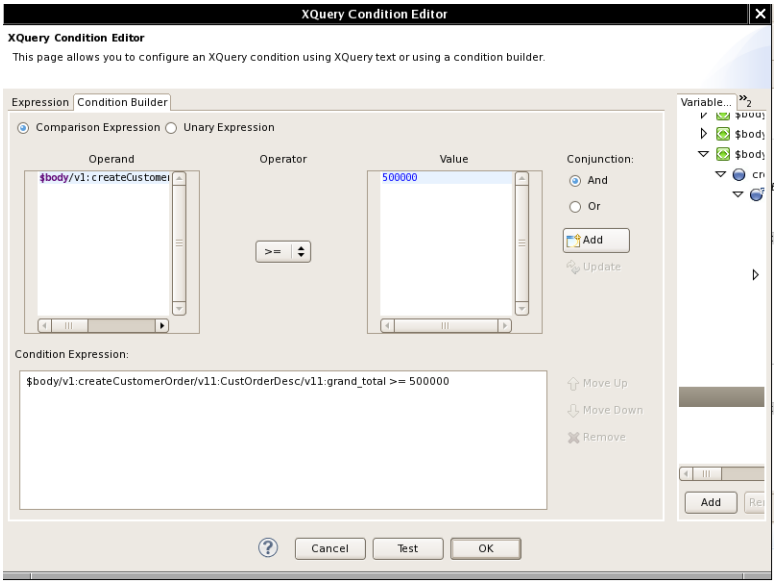
7. Drag the element to Operand window:



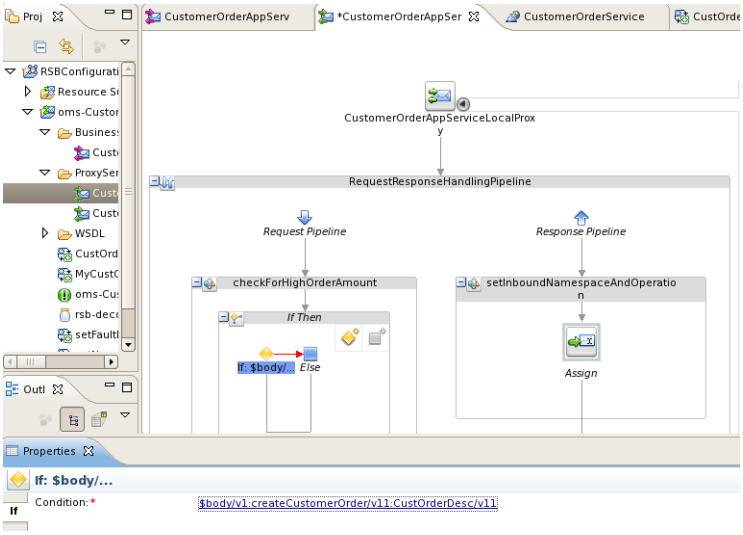
8. Select the Operator and enter a Value for comparison.



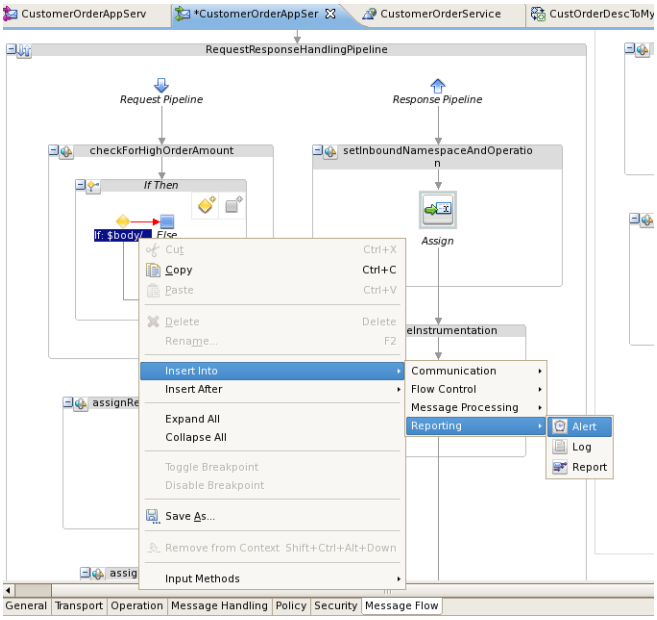
9. Click **Add** to add the condition to Condition Expression window.



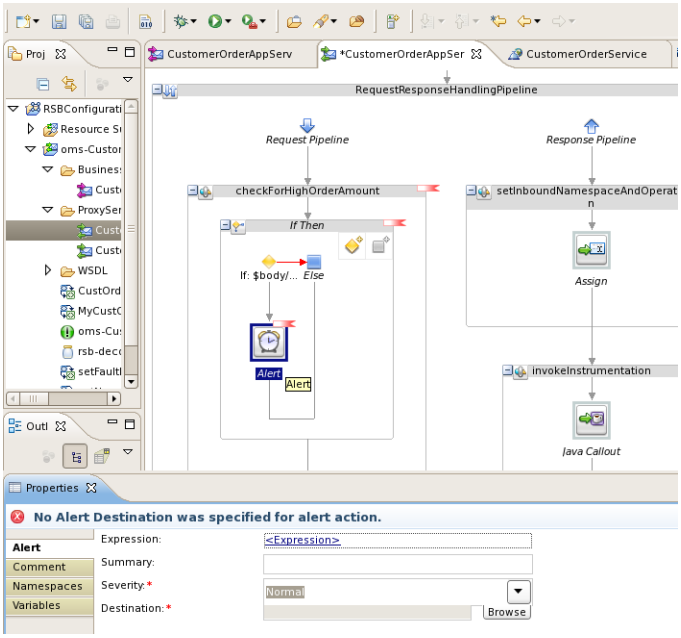
10. You can build more complex expression using And and Or buttons. We will keep it simple here. After building the condition expression, click **OK**.



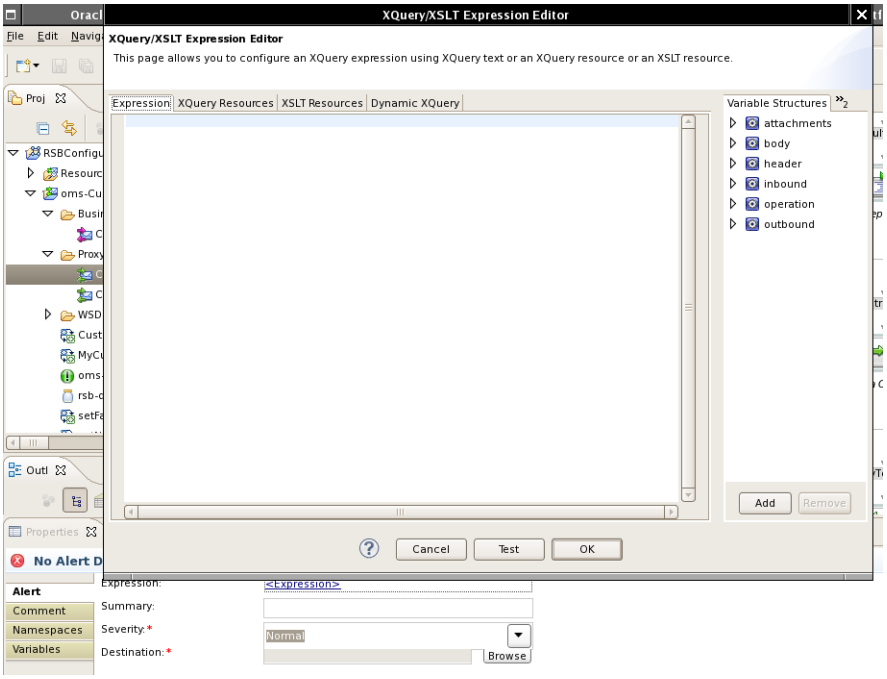
11. Now we need to add Alert action for this If condition as shown below:



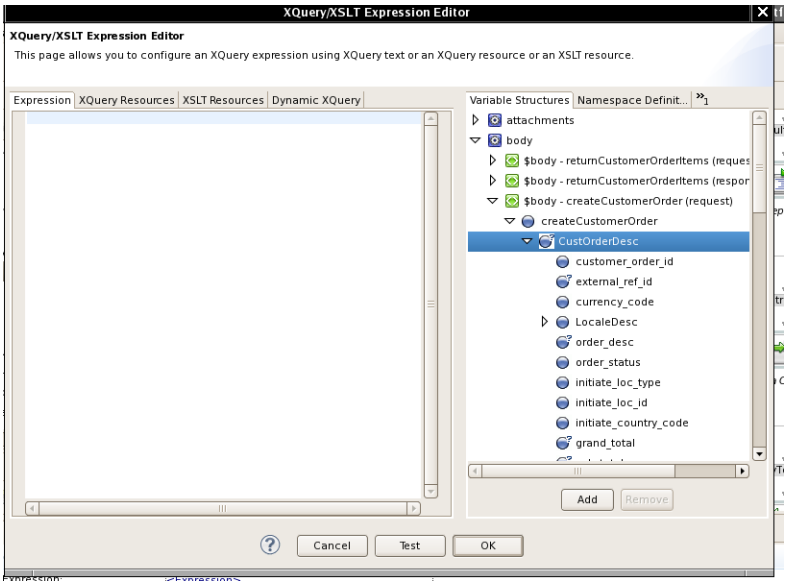
12. The alert action gets added, right click the If condition added in the above steps, select **Insert Into > Reporting > Alert** as shown:



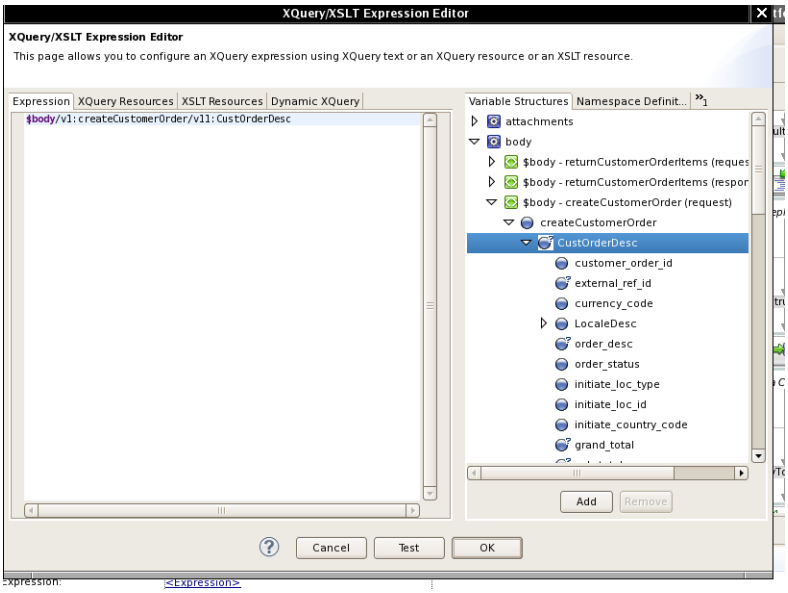
13. In the **Expression** field, you can enter the xml that you want to see in alert description when alert is generated. For example, you may want to see the whole SOAP body which caused the alert to be generated or a subset of the SOAP body. Click the **<Expression>** link to select the XML.



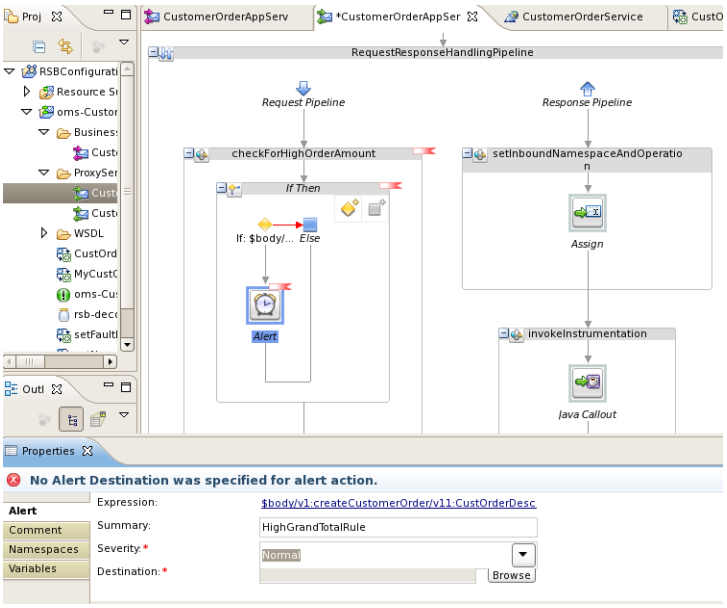
14. In this example, we will add CustOrderDesc element to the expression.



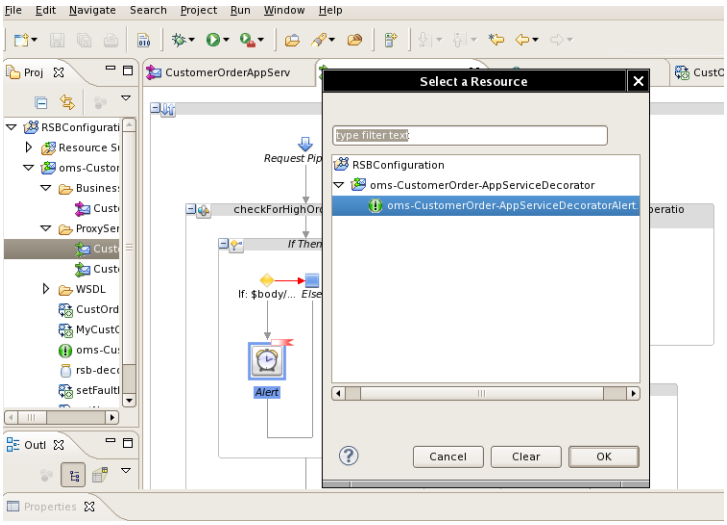
15. Drag CustOrderDesc to the Expression window.



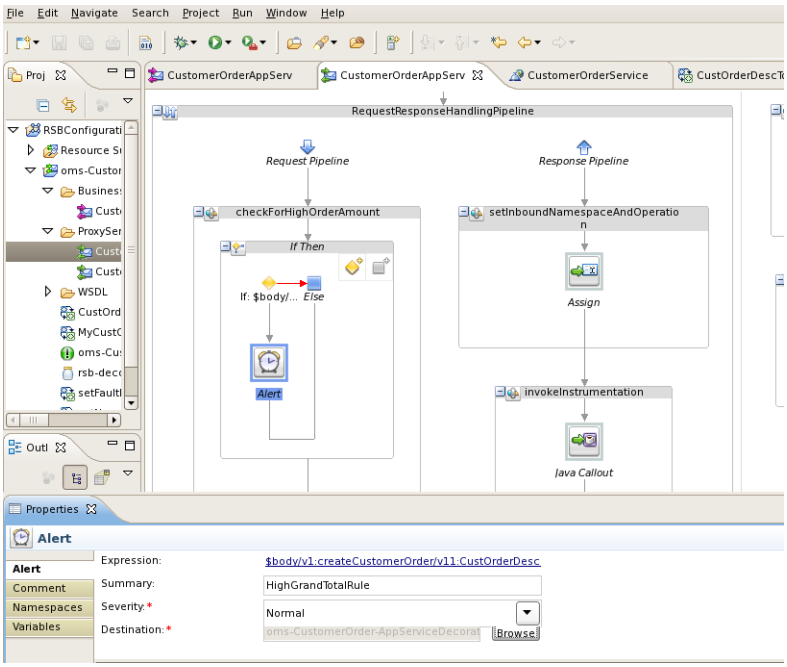
16. Click **OK**. In the Summary field, enter an appropriate name for the alert.



17. Select Destination for the Alert. You should select the destination that was created by default in this project. Click **Browse**.



18. Click **OK** to close the dialog box.



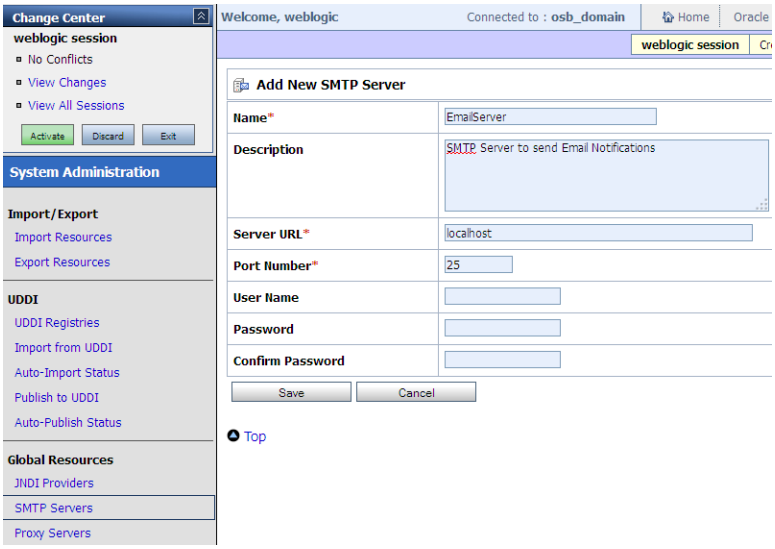
This finishes up the steps for adding new business alert to a message flow in a decorator.

How to add E-mail Notification for Alerts

The default alert destination created by RSB only sends alerts on OSB reporting provider JMS. The alert destination can also be configured to send email notifications; this will be useful to get immediate notifications for SLA alerts. For generating email notifications first step is to create SMTP server configuration in OSB server. You need to have a SMTP server running and URL, port number information available. Following are the steps to create SMTP server configuration using OSB console:

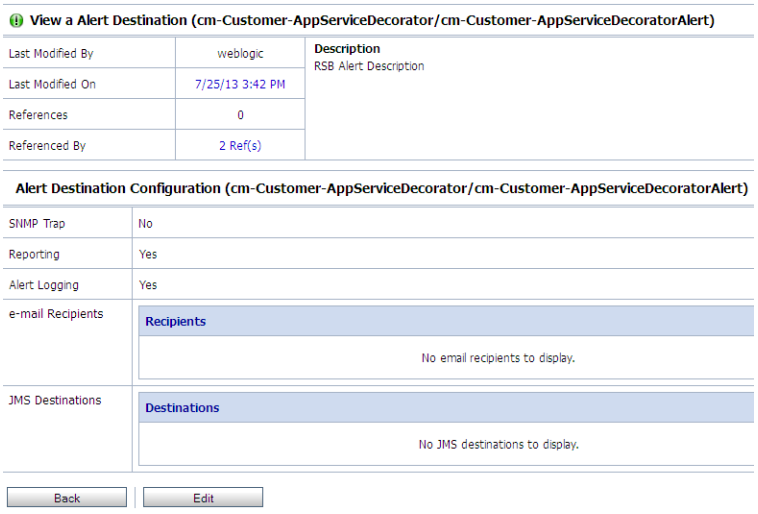
1. Create a new session in OSB console.
2. Go to **System Administration > SMTP Servers** page.
3. Click **Add**.
4. Provide a name for the server.
5. Provide URL of the SMTP server. If the SMTP server is running on localhost, then the URL will be localhost.
6. Enter SMTP port number, generally it is 25.
7. If it is secured, then provide username/password. Generally it is not required when running on localhost.
8. Click **Save**. Click **Activate** and **Submit** to commit changes to the server.

9. This completes the steps for creating SMTP server configuration. Following is a screenshot of this:

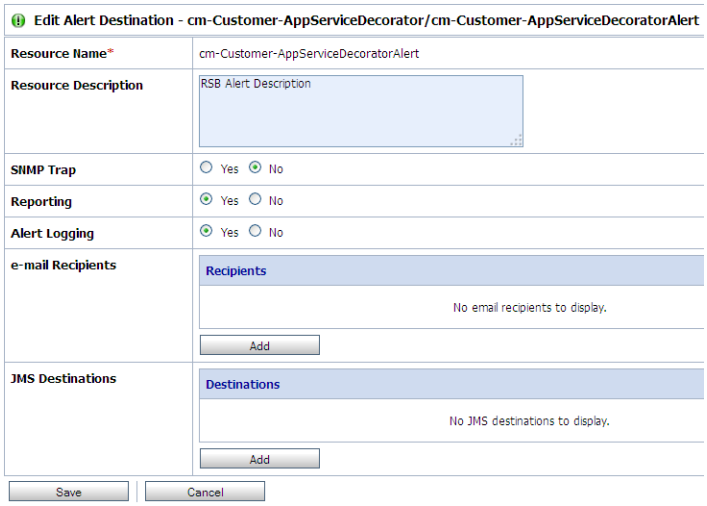


After creating SMTP server configuration, we need to update alert destination to use the SMTP server for sending notifications.

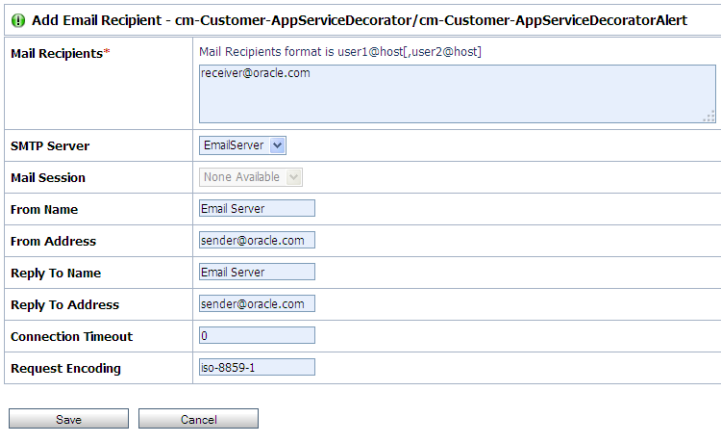
1. Create a new session in OSB console.
2. Go to **Project Explorer** tab and browse to the project for which you want to modify the alert destination. In this example, we will update cm-Customer-AppServiceDecorator project. When you click the project; it shows the list of files in that project. The default alert destination follows the naming convention as <appName>-<ServiceName>-AppServiceDecoratorAlert. So the file name here will be cm-Customer-AppServiceDecoratorAlert.
3. Click the alert destination to go to alert destination configuration page.



4. Click **Edit**.



5. In the e-mail Recipients section, click Add button. In the next page, you need to provide details about senders and receivers of e-mail notifications as shown below:



   - Mail Recipients: This needs the email addresses of the persons who should receive email notification.
   - SMTP Server: Select the name of the SMTP server that was created earlier.
   - From Name: The name of the person on whose behalf the notification is sent.
   - From Address: Email address of the person on whose behalf the notification is sent.
   - Reply To Name: Name of the person which should show in reply-to field of the email.
   - Reply To Address: Email address which should show in reply-to field of email.

6. Click **Save** after entering all the values. Click **Activate and Submit** to commit changes to the server. This completes the steps required for setting up email notifications for alerts.

5

Introduction to Injector Service

Injector Service is a mechanism for external web services to subscribe data published in RIB topics. In the absence of this method, external applications will always have to subscribe directly to RIB JMS topics and parse the messages. With help of the injector service, RIB can now invoke external web services to send messages to those applications.

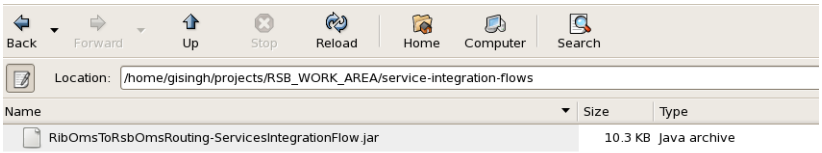
Injector Service Implementation in RSB

RSB has a service integration flow which is based on RIB injector service. The purpose of this service integration flow is to route messages from RIB-OMS application to RSB decorator services. This service integration flow is an OSB project and it is available in RsbServiceIntegrationFlowPak14.0.0ForRibOmsToRsbOmsRouting_eng_ga.zip PAK. The OSB jar packaged inside the PAK is RibOmsToRsbOmsRouting-ServicesIntegrationFlow.jar.

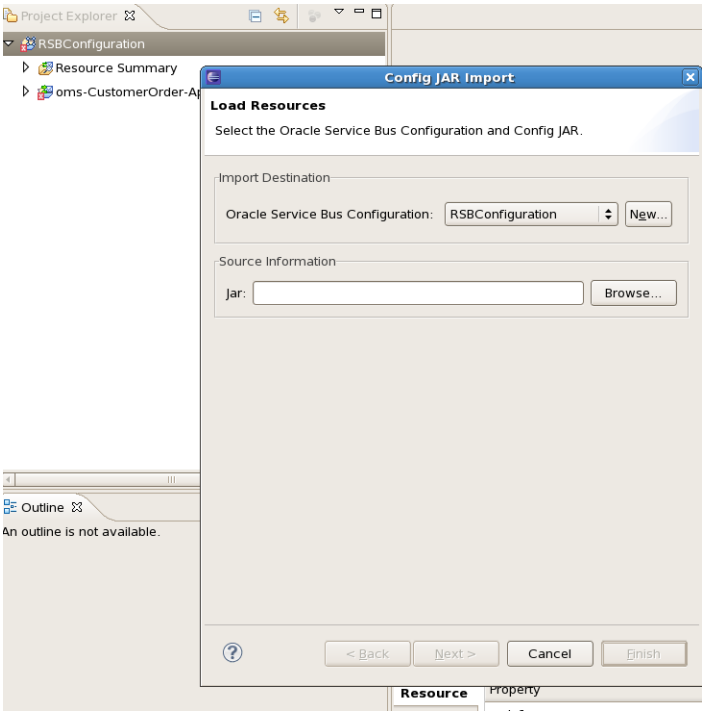
This OSB jar contains a Proxy Service which is based on Injector Service WSDL. The name of the proxy service is RibOmsToRsbOmsRoutingService. The WSDL contains an operation named as injectMessage(). This operation requires four parameters: message family, message type, business object ID and payload. When RIB-OMS application receives a message on one of its topics, it builds the request message with appropriate values for the parameters and invokes injectMessage() method of the RibOmsToRsbOmsRoutingService proxy service. Business Object Id is an optional parameter and it may be null but rest of the parameters are required.

How to import RSB-OMS routing service in OEPE

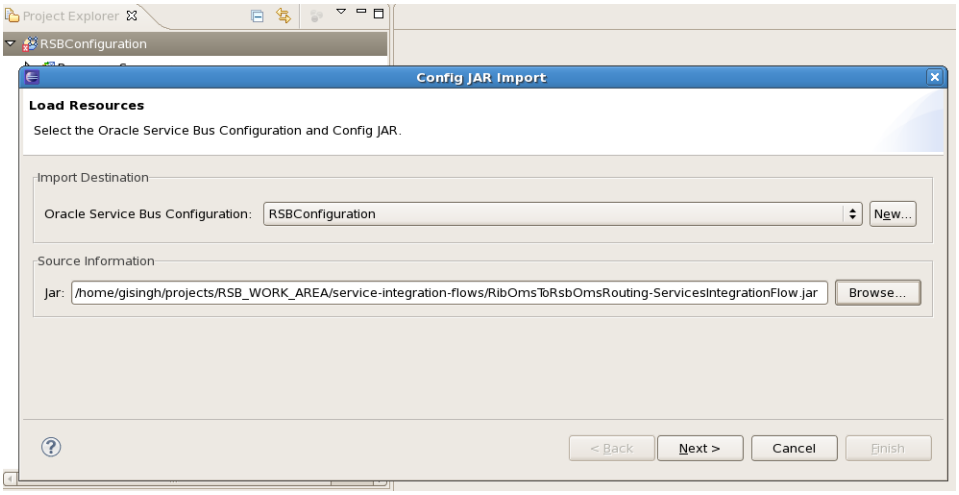
1. Copy the RibOmsToRsbOmsRouting-ServicesIntegrationFlow.jar to RSB_WORK_AREA/service-integration-flows folder. The directory structure looks like this:



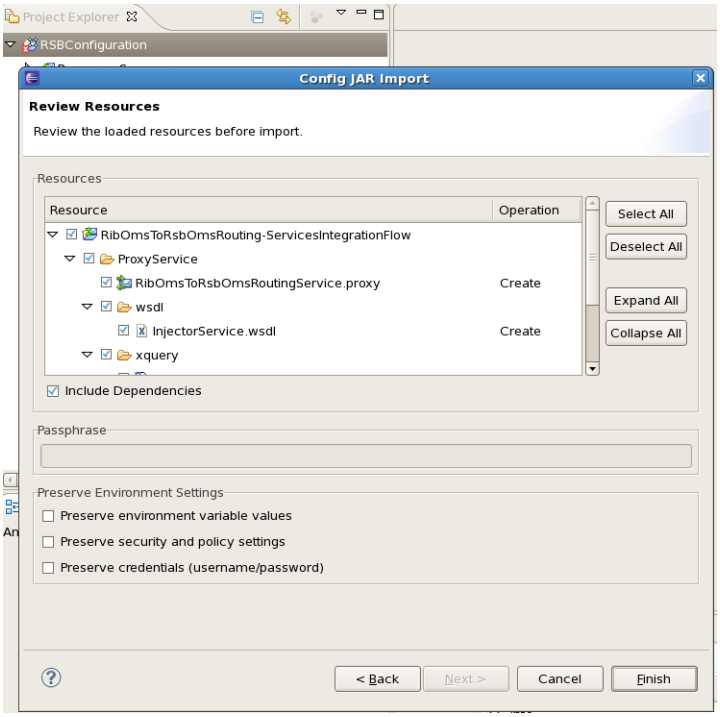
2. Now we need to import this jar in OEPE. To import the jar, right-click RSBConfiguration and select **Import > Oracle Service Bus - Configuration Jar**.



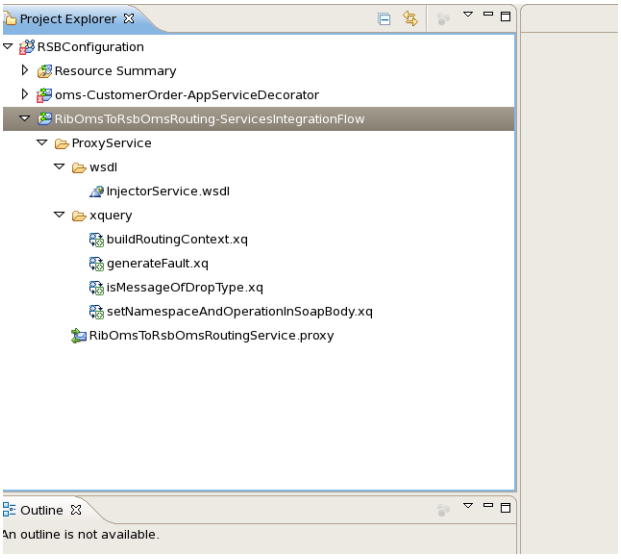
3. Click **Browse** and select the jar file.



4. Click **Next** button. It will show all the files available in the jar.

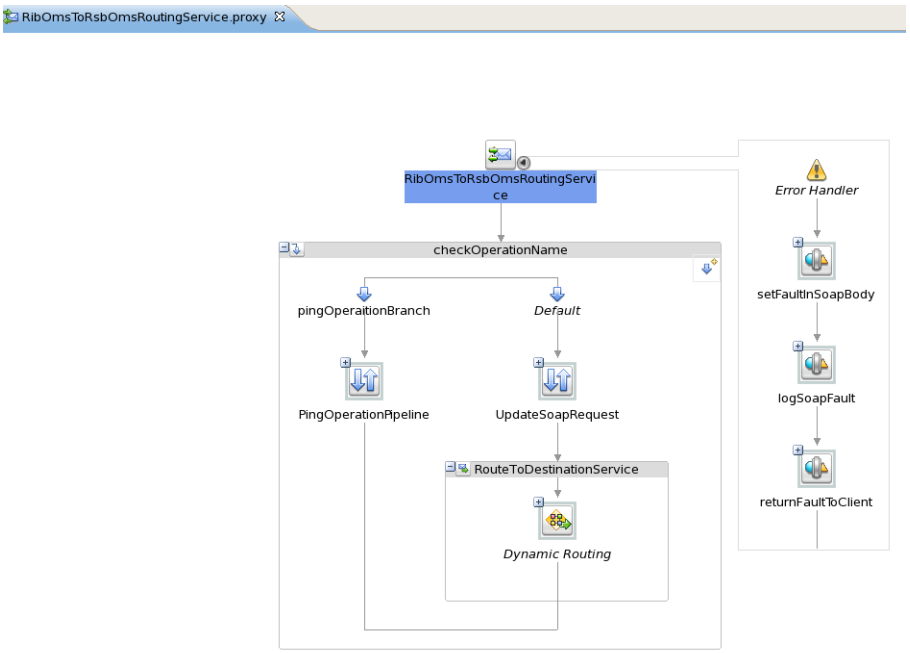


5. Click **Finish**. The jar is imported as an OSB project and the project structure looks like the following:

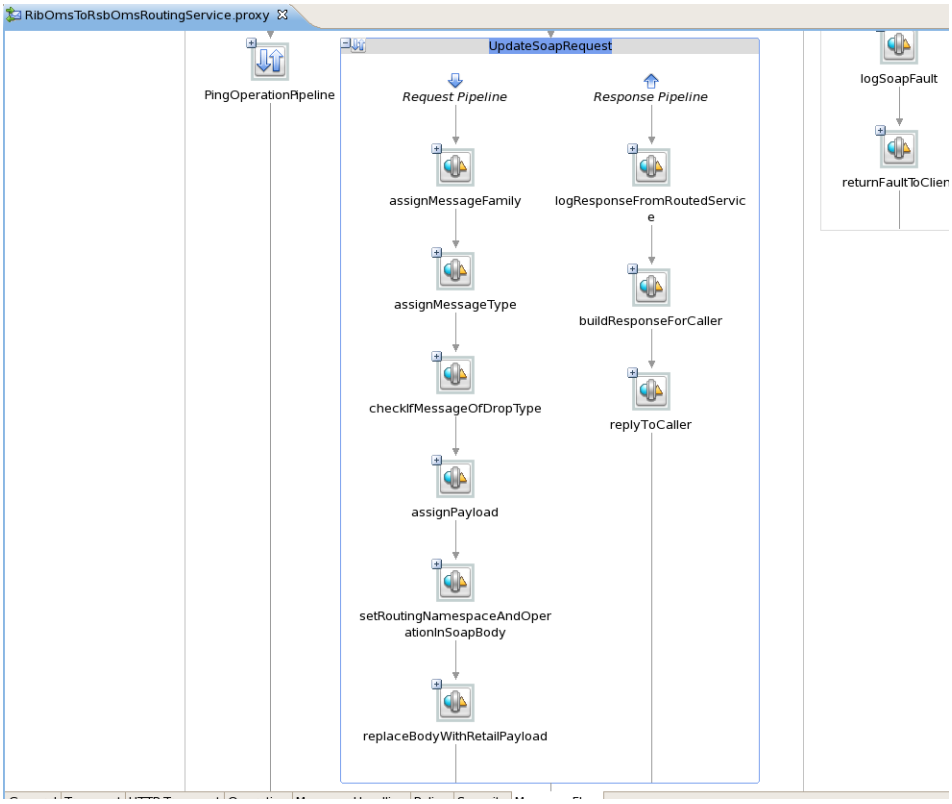


Message Flow in RSB-OMS Routing Service

The following diagrams show the message flow in the proxy service of the routing project. The whole message flow does not fit in once screenshot, so the first picture shows message flow at a high-level; it shows all the pipelines that define the message flow.



The next screenshot is of the updateSoapRequest pipeline-pair where most of the message processing takes place.



When RibOmsToRsbOmsRoutingService proxy service receives the request XML, it executes the message flow. The sequence of activities in the flow is explained below:

- The Message flow has a conditional branch, which checks for operation name in the SOAP request.
- The first condition is to check if operation invoked is ping. If operation name is ping, then the proxy service does not need to do further processing, it just builds a success response for ping method and returns the response to the client.
- If operation name is not ping, then the message goes to a Pipeline Pair which is named as updateSoapRequest. In request flow of that pipeline pair, these are the things that take place:
  1. Find the message family from the request and assign to a variable.
  2. Find the message type from the request and assign to a variable.
  3. Execute an xquery to check if the message needs to be dropped, that is, the proxy service should do nothing and return an appropriate response to the client. So if the message is of drop type, then it skips rest of the pipeline and returns an appropriate response to the client. Table 5-1 shows the combination of message family and message type which are dropped by the proxy service.
  4. If the message is not of drop type, then it extracts the payload from the request and saves it to a variable.
  5. Execute an xquery to set namespace and operation name in the outgoing request for the target decorator service.
  6. Build a new SOAP body with the payload that was stored in a variable.
- The proxy service uses dynamic routing to route the request to target decorator service. The dynamic routing action is based on an xquery file. This xquery builds

route message context which is used by dynamic routing to route to the appropriate service. The table 5-2 shows the mappings between message family, message type which are routed to decorator service URIs.

- Message is routed to the proxy service of the target decorator project.
- When the target decorator returns the response, then the response pipeline is executed. In the response pipeline, the response returned from decorator is logged. Finally a string with success response is returned to the proxy service client.
- If an error occurs in the RibOmsToRsbOmsRoutingService message flow, then the Service Error handler pipeline is executed. In this pipeline, an XQuery is used to build appropriate fault message. The fault is returned to the client with failure status.

The following table contains the combination of Message Family and Message Type that are dropped by routing service:

Table 5–1

Message Family	Message Type
pendreturn	pendretcre
pendreturn	pendretmod
pendreturn	pendretdel
pendreturn	pendretdtlcre
pendreturn	pendretdtlmod
pendreturn	pendretdtldel
asnout	asnoutmod
receiving	appointcre
receiving	appointdel
receiving	receiptcre
receiving	receiptmod
receiving	appointdtlcre
receiving	appointdtldel
receiving	appointdtlmod
receiving	appointhdrmod

The following table contains the list of decorator service URI to which messages with combination of Message Family, Message type are routed:

Table 5–2

Message Family	Message Type	Decorator Service URI
pendreturn	rtrnrcptnotify	oms-OrderReturn-AppServiceDecorator/ProxyService/OrderReturnAppServiceLocalProxy
pendreturn	rtrncomplete	oms-OrderReturn-AppServiceDecorator/ProxyService/OrderReturnAppServiceLocalProxy

Table 5–2 (Cont.)

Message Family	Message Type	Decorator Service URI
sostatus	sostatuscre	oms-StockOrderStatus-AppServiceDecorator/ProxyService/StockOrderStatusAppServiceLocalProxy
asnout	asnoutcre	oms-AdvancedShipmentNotification-AppServiceDecorator/ProxyService/AdvancedShipmentNotificationAppServiceLocalProxy
fulfilordcfm	fulfilordcfmcre	oms-FulfillOrderConfirm-AppServiceDecorator/ProxyService/FulfillOrderConfirmAppServiceLocalProxy
fulfilordcfmcnc	fulfilordcfmcnccre	oms-FulfillOrderCancelConfirm-AppServiceDecorator/ProxyService/FulfillOrderCancelConfirmAppServiceLocalProxy
asnin	asnincre	oms-VendorShipmentNotification-AppServiceDecorator/ProxyService/VendorShipmentNotificationAppServiceLocalProxy
asnin	asninmod	oms-VendorShipmentNotification-AppServiceDecorator/ProxyService/VendorShipmentNotificationAppServiceLocalProxy
asnin	asnindel	oms-VendorShipmentNotification-AppServiceDecorator/ProxyService/VendorShipmentNotificationAppServiceLocalProxy
receiving	receiptordcre	oms-CustomerOrder-AppServiceDecorator/ProxyService/CustomerOrderAppServiceLocalProxy

How to add new routing flow in RSB-OMS Routing Service

The table 5-2 contains the list of all decorator URIs services to which RibOmsToRsbOmsRoutingService routes the messages. It is also possible to add routing to a new decorator service by modifying XQuery files in the OSB project. This document covers only the RSB side of changes. For adding message flow for a new message family and message type from RIB, there are changes required in RIB side too. Please refer to RIB documents for changes in the RIB side. This document assumes that RIB-OMS application can invoke the RibOmsToRsbOmsRoutingService for a new message family and message type and now RibOmsToRsbOmsRoutingService needs to route those messages to the appropriate decorator. Follow the steps to achieve the same:

1. If the message needs to be just dropped by RibOmsToRsbOmsRoutingService and not to be processed further, the only change required in the project is to add the new message family and type in the isMessageOfDropType xquery file.
2. If the message need not be dropped and should be routed to a decorator, then there are no changes required in isMessageOfDropType file. In this case, open the buildRoutingContext xquery file. This xquery builds the path to the target decorator service.
3. For routing to a decorator service, the request message also needs to contain the operation name and namespace for the target service. To set new operation name and namespace in the request message, open the setNamespaceAndOperationInSoapBody xquery file and add appropriate code.

The changes to the three xquery files are all that is needed in OSB project. There is a properties file in rsb-home that also needs to be modified. This properties file is used by RSB builder tool to update RSB artifacts appropriately.

A
Appendix

The following code snippet shows the content of setOutboundNamespaceAndOperationInSoap xquery file:

```
xquery version "1.0" encoding "UTF-8";
(:: pragma  parameter="$soapBody" type="xs:anyType" ::)
(:: pragma  type="xs:anyType" ::)

declare namespace xf =
"http://tempuri.org/oms-CustomerOrder-AppServiceDecorator/xquery/setOutboundNamesp
aceAndOperationInSoap/";

declare function xf:setOutboundNamespaceAndOperationInSoap($soapBody as
element(*))
as element(*) {

    let $namespace := fn-bea:serialize(fn:namespace-uri($soapBody/*[1]))
    let $operation := local-name($soapBody/*[1])

let $destNamespace :=

'http://www.oracle.com/retail/oms/integration/services/CustomerOrderService/v1'

let $destOperation :=
    if($operation= "queryMyCustomerOrderResponse")  then
        'queryCustomerOrderResponse'

    else if($operation= "createMyCustomerOrderResponse")  then
        'createCustomerOrderResponse'

    else
        $operation

let $updatedNsBody :=
replace(fn-bea:serialize($soapBody),$namespace,$destNamespace)

let $updatedOperationBody :=
replace(fn-bea:serialize($updatedNsBody),$operation,$destOperation)

return fn-bea:inlinedXML($updatedOperationBody)
};

declare variable $soapBody as element(*) external;

xf:setOutboundNamespaceAndOperationInSoap($soapBody)
```

B

Appendix

The following code snippet shows the content of setInboundNamespaceAndOperationInSoap xquery file:

```
xquery version "1.0" encoding "UTF-8";
(:: pragma  parameter="$soapBody" type="xs:anyType" ::)
(:: pragma  type="xs:anyType" ::)

declare namespace xf =
"http://tempuri.org/oms-CustomerOrder-AppServiceDecorator/xquery/setInboundNamespa
ceAndOperationInBody/";

declare function xf:setInboundNamespaceAndOperationInBody($soapBody as element(*))
as element(*) {
let $namespace := fn-bea:serialize(fn:namespace-uri($soapBody/*[1]))
    let $operation := local-name($soapBody/*[1])

let $destNamespace
:='http://www.oracle.com/retail/oms/integration/services/CustomerOrderService/v1'


let $destOperation :=
    if($operation = "queryMyCustomerOrderResponse")  then
        'queryCustomerOrderResponse'

    else if($operation = "createMyCustomerOrderResponse")  then
        'createCustomerOrderResponse'

    else
        $operation


let $updatedNsBody :=
replace(fn-bea:serialize($soapBody),$namespace,$destNamespace)

let $updatedOperationBody :=
replace(fn-bea:serialize($updatedNsBody),$operation,$destOperation)

return fn-bea:inlinedXML($updatedOperationBody)
};

declare variable $soapBody as element(*) external;

xf:setInboundNamespaceAndOperationInBody($soapBody)
```