

Oracle® Retail Service Backbone

Developers Guide

15.0

E67904-01

December 2015

Oracle® Retail Service Backbone Developers Guide, 15.0

E67904-01

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Primary Author: Carrie Federer

Contributing Author: Anshuman Accanoor

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

- (i) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.
- (ii) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.
- (iii) the software component known as **Access Via**™ licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.
- (iv) the software component known as **Adobe Flex**™ licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.

You acknowledge and confirm that Oracle grants you use of only the object code of the VAR Applications. Oracle will not deliver source code to the VAR Applications to you. Notwithstanding any other term or condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR

Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all 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.

Contents

Send Us Your Comments	vii
Preface	ix
Audience	ix
Documentation Accessibility	ix
Related Documents	ix
Customer Support	x
Review Patch Documentation	x
Improved Process for Oracle Retail Documentation Corrections	x
Oracle Retail Documentation on the Oracle Technology Network	xi
Conventions	xi
1 Getting Started with the RSB Developer Guide	
Introduction	1-1
Types of Integrations Addressed by RSB	1-1
Technical Architecture	1-2
2 Building RSB Integration Flows	
Development Tools	2-1
OSB Console vs JDeveloper	2-1
Installing JDeveloper	2-1
Introduction to RSB Decorator jar Files	2-2
Introduction to RSB Service Integration Flow jar Files	2-2
How to Setup RSB Workbench	2-3
JDeveloper Workspace	2-3
Development Lifecycle	2-4
How to Import RSB Decorator jar into JDeveloper	2-5
Creating a New Service Bus Application	2-5
Components of RSB Decorator Project	2-7
Business Services	2-8
Local and Remote Proxy Services	2-8
WSDL files	2-8
Alert Destination	2-9
Instrumentation Jar File	2-9
Fault XQuery File	2-9

RSB Decorator Message Flow	2-9
Message Flow in Remote Proxy Service	2-10
Message flow in Local Proxy Service	2-11
How to Export RSB Decorator Project.....	2-11

3 Integration with Third-Party Application Services

Types of Customization	3-1
How to Import WSDL into RSB Decorator Project.....	3-1
How to Map Namespaces and Operation Names	3-7
How to do Payload Transformation	3-27

4 Introduction to Alerts

Pipeline/Business Alerts	4-1
SLA Alerts.....	4-1
Default Alerts in RSB Decorator Projects.....	4-2
How to add new SLA alert.....	4-2
How to Add New Pipeline/Business Alert.....	4-12
How to add E-mail Notification for Alerts	4-19

5 Introduction to Injector Service

Injector Service Implementation in RSB	5-1
How to import RSB-OMS routing service in JDeveloper	5-1
Message Flow in RSB-OMS Routing Service	5-3
How to add new routing flow in RSB-OMS Routing Service.....	5-6

A Appendix

B Appendix

Send Us Your Comments

Oracle® Retail Service Backbone Developers Guide, 15.0.

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 that have purchased support have access to electronic support through My Oracle Support. For information, visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit

<http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Retail Service Backbone 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*
- *Oracle Retail Service Backbone Installation Guide*

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, 15.0) or a later patch release (for example, 15.0.1). 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.
<i>italic</i>	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.

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

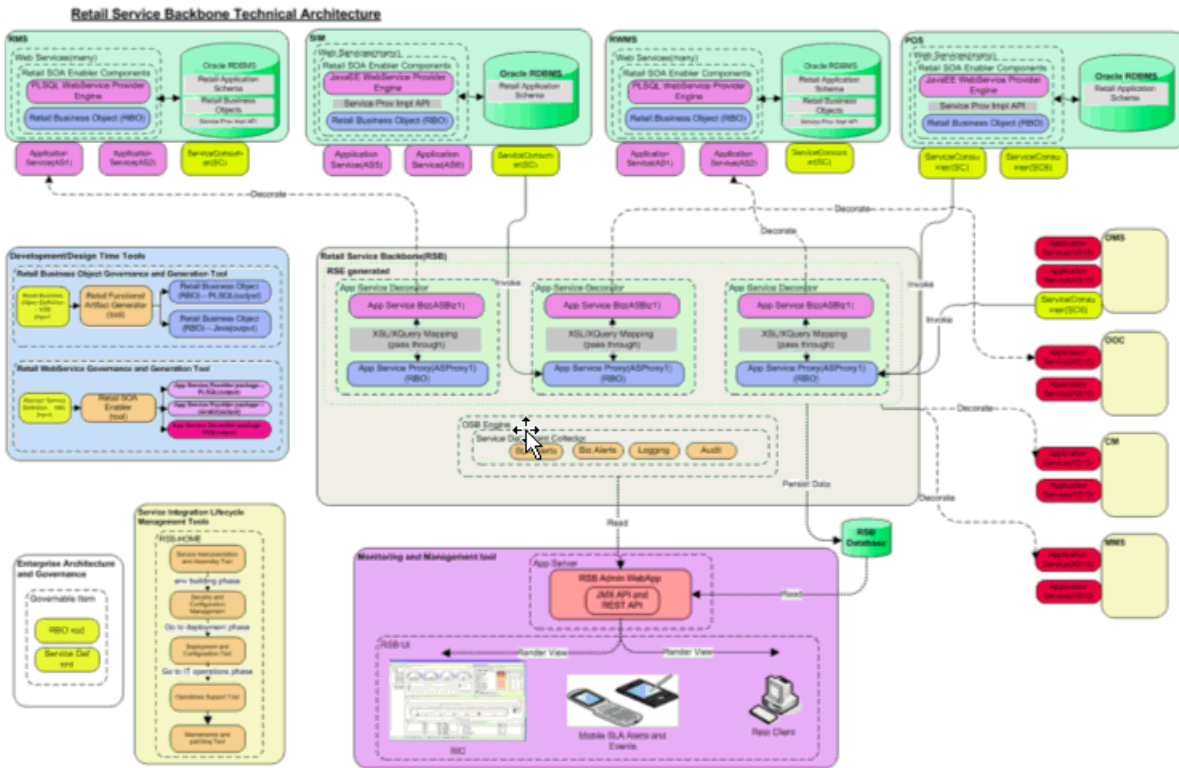
Oracle Retail uses three types of integration patterns:

- Request Reply
- Fire and Forget
- Bulk Data

Technical Architecture



RSB Deployment Architecture



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 JDeveloper.

OSB Console vs JDeveloper

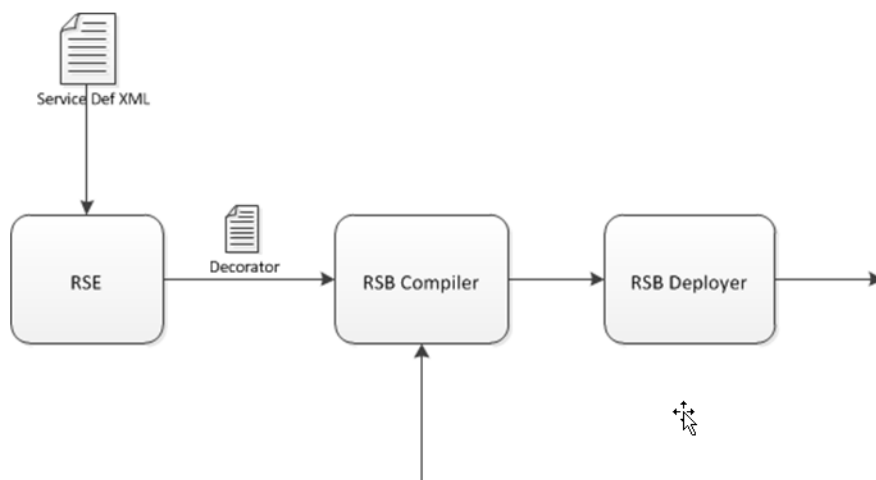
There are two ways to make programmatic changes to web services hosted in OSB server: OSB console and JDeveloper. However, OSB console is an operational tool and is not recommended as a programming tool, even though OSB console provides the feature to make programmatic changes. The recommended approach is to use JDeveloper 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 JDeveloper

Use this JDeveloper for your OSB/RSB programming tasks.

JDeveloper does not need to be installed in the same machine where RSB builder tool is located. JDeveloper 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.

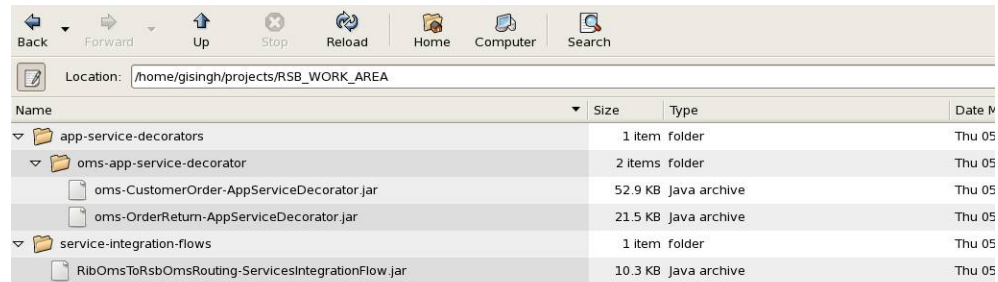
RsbServiceIntegrationFlowPak15.0.0ForRibOmsToRsbOmsRouting_eng_ga.zip is the only PAK available for this release. This PAK contains a proxy service which routes the data coming from RIB-OMS to various RSB decorator services.

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 JDeveloper has been installed. In this document we will refer to that location as RSB_WORK_AREA.

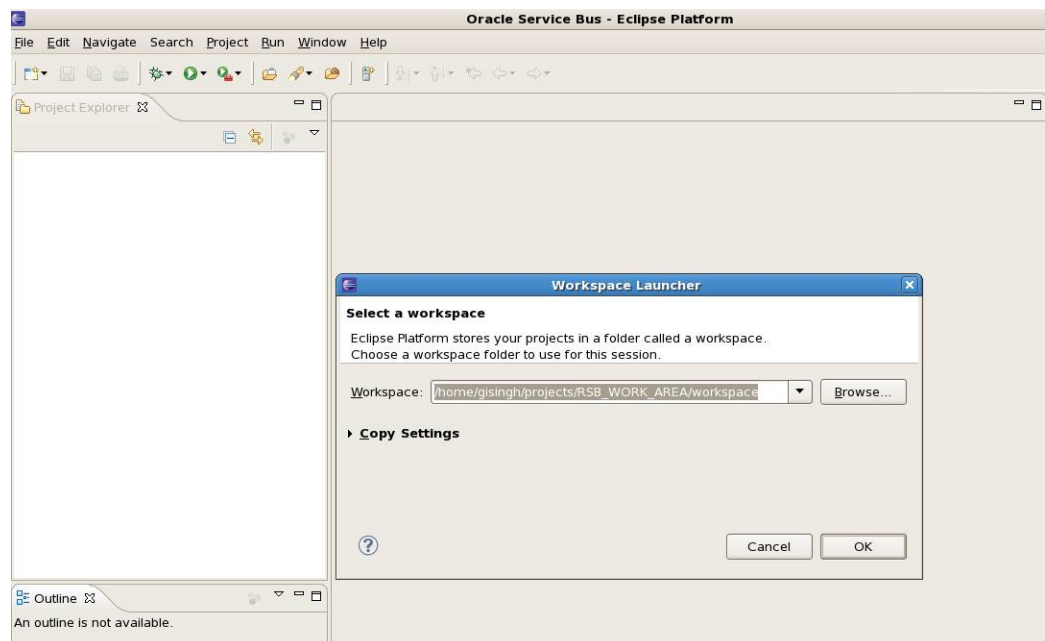
RSB has two types of OSB projects which can be customized: app service decorators and service integration flows. Therefore, it is recommended to have a directory for each type of project in RSB_WORK_AREA. The following is a screenshot of the recommended directory structure:



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.

JDeveloper Workspace

When an OSB jar is imported in JDeveloper, it extracts the jar inside the JDeveloper 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.



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 JDeveloper where the JDeveloper workspace is OSB_WORK_SPACE. The steps for importing jar have been shown below.
3. JDeveloper 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 JDeveloper

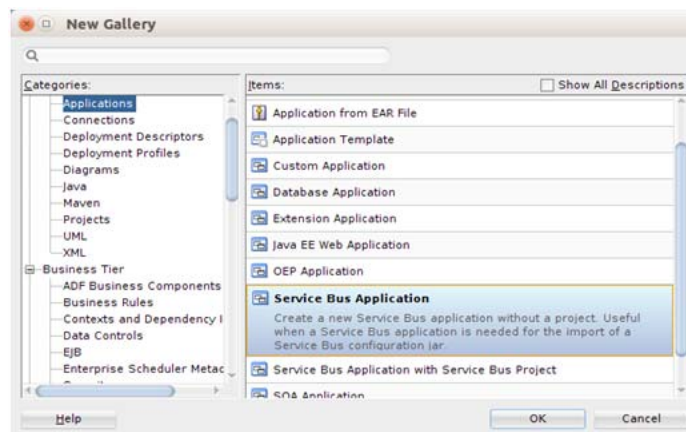
This section provides a step-by-step guide for how to import RSB decorator jar into JDeveloper workspace.

Creating a New Service Bus Application

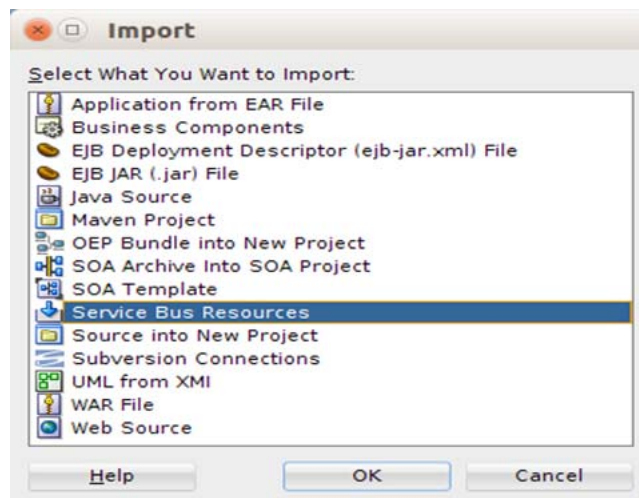
To create a new Service Bus Application, do the following:

1. Select **File>New>Application>Service Bus Application**.
2. Provide Application Name and click **Finish** to create a new Service Bus Application.

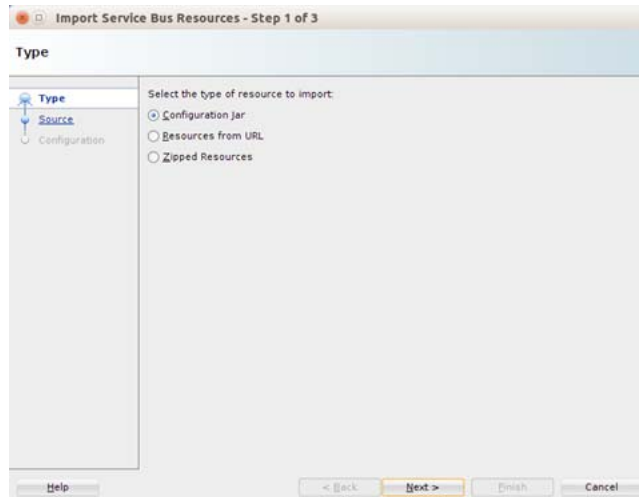
This creates an OSB Configuration Project, in which you can import RSB decorator or service integration flow jar files.



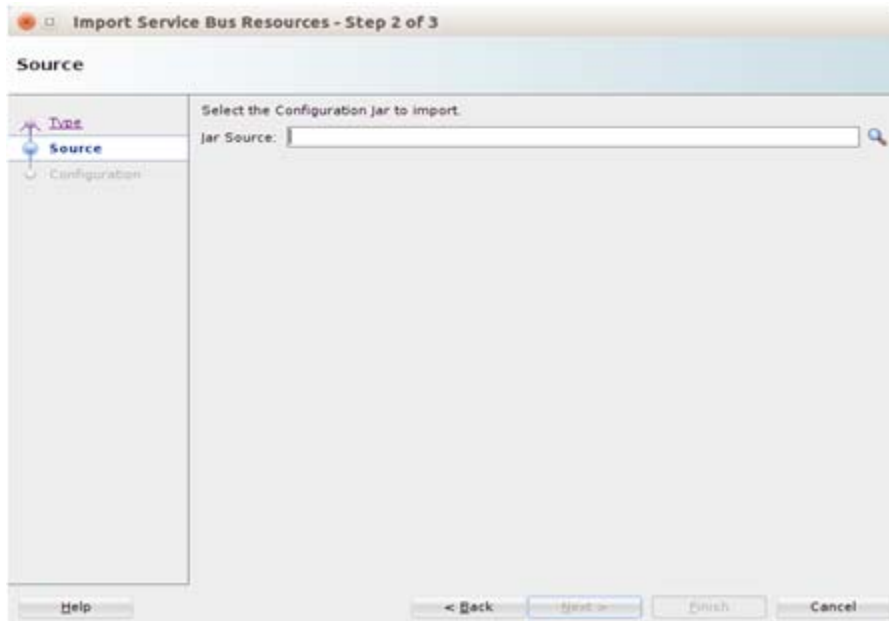
3. To import a decorator jar, select **File>Import**. The following screen is displayed:



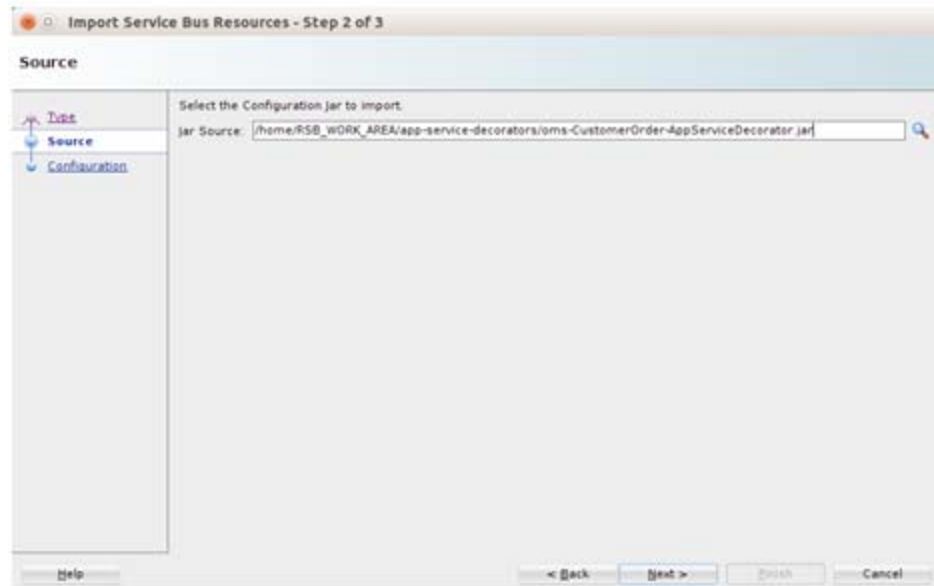
4. Select **Service Bus Resources** and click **OK**.



5. Select Configuration jar and click **Next**.
6. Specify the path to the jar file that you want to import and click **Next**.



7. Click **Finish** to import decorator jar.



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 JDeveloper. 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.

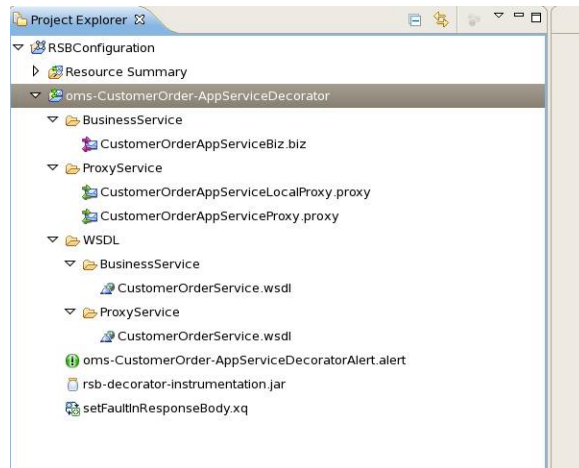
Note: While making changes to OSB projects, it may be cumbersome to copy the jar to rsb-home/service-assembly-home every-time and go through compilation and deployment phases to test the changes. Instead, you can test the OSB project by deploying the jar in a development OSB server environment and test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

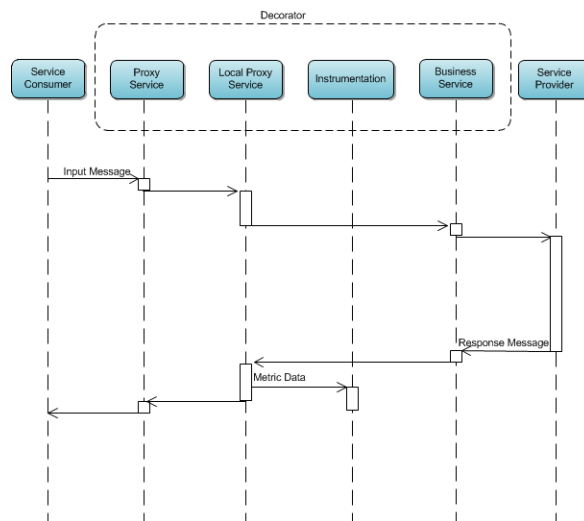
Instrumentation Jar File

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.

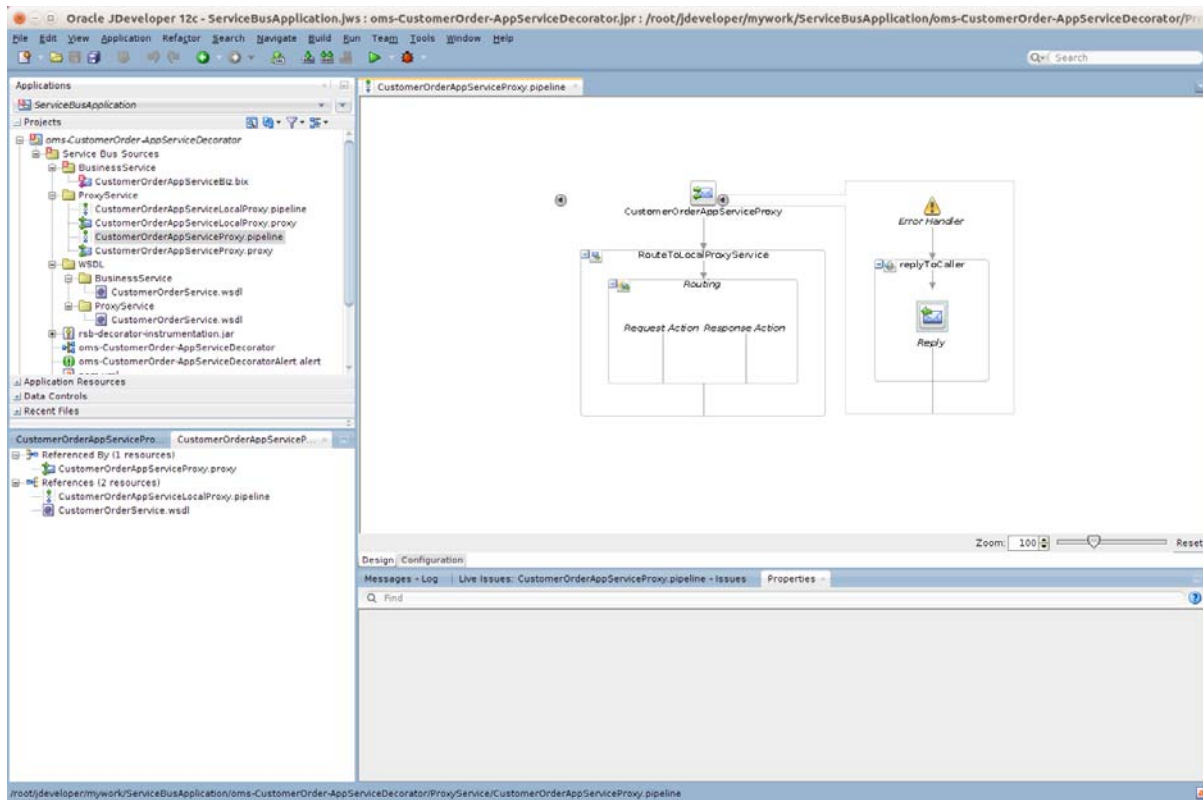
RSB Decorator Message Flow



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

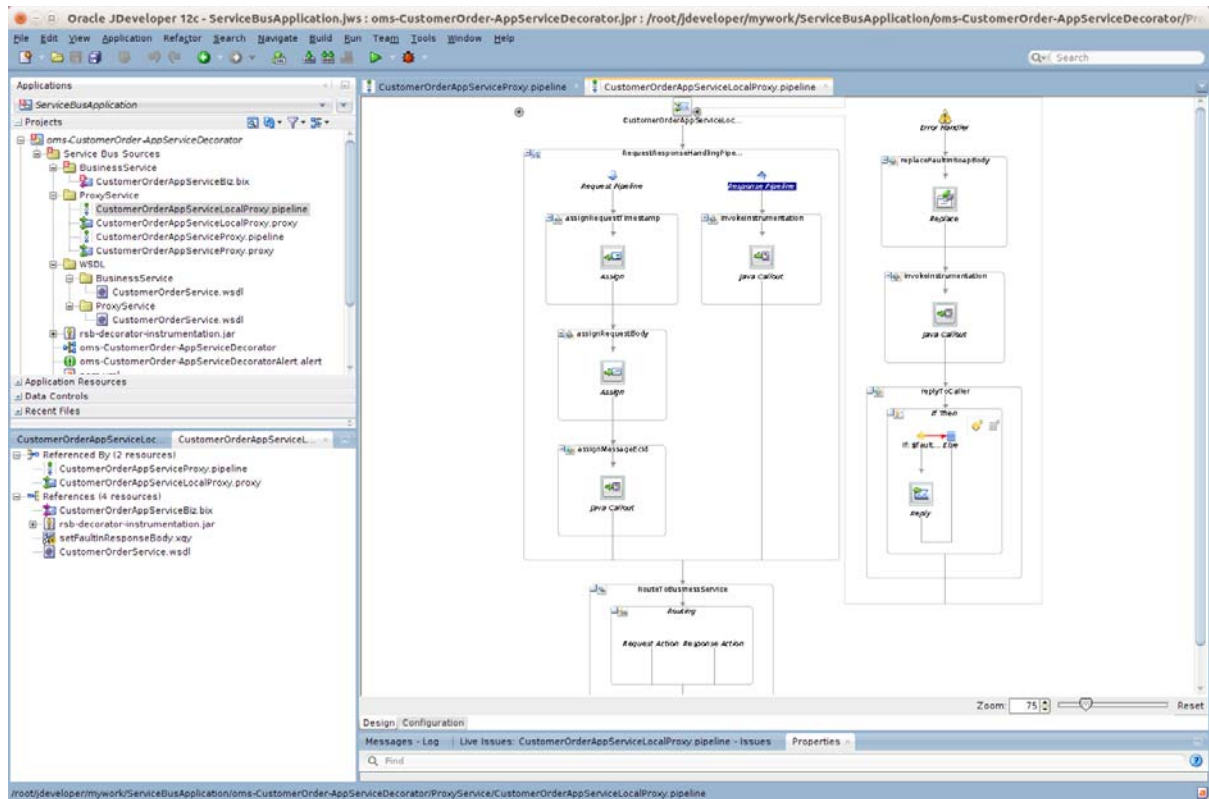
Message Flow in Remote Proxy Service

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:

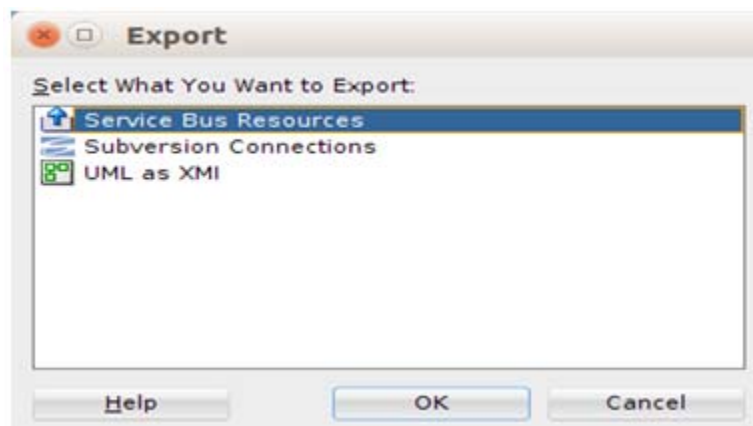


How to Export RSB Decorator Project

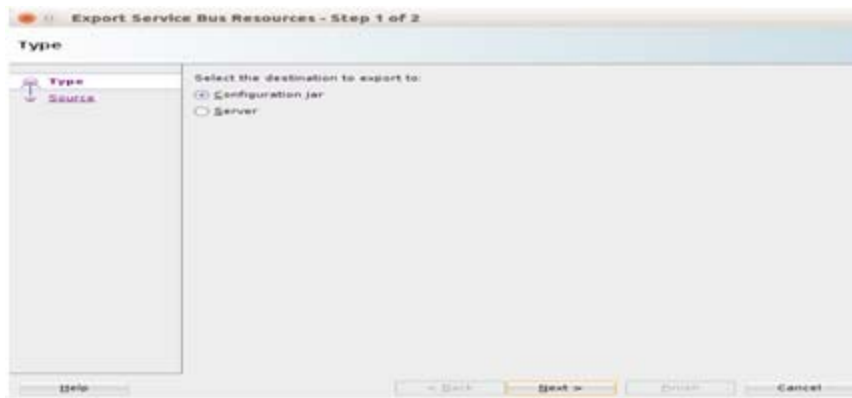
Once you have completed changes in a decorator project, you can export it back to the jar and deploy and test.

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

1. Select **File > Export > Service Bus Resources** and click **OK**.



2. Select **Configuration jar** and click **Next**.



3. Select resource, provide jar file location and click **Finish**.



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

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

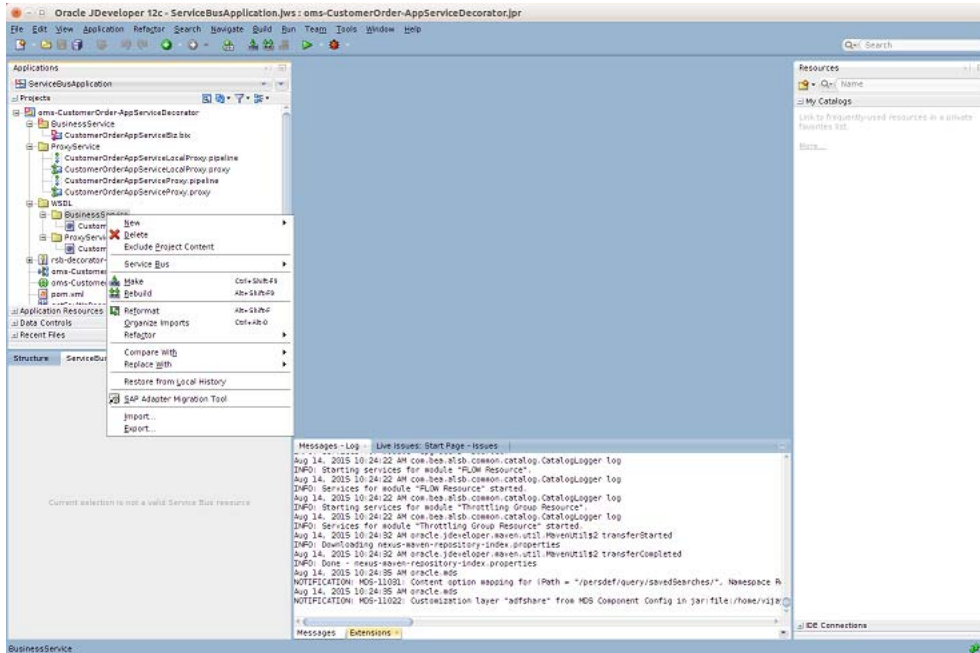
The web services you want to integrate to RSB is likely to be different from what the corresponding provided decorators expect. These differences can be broadly classified into two categories:

- Payload is different
- Service and operation names are different

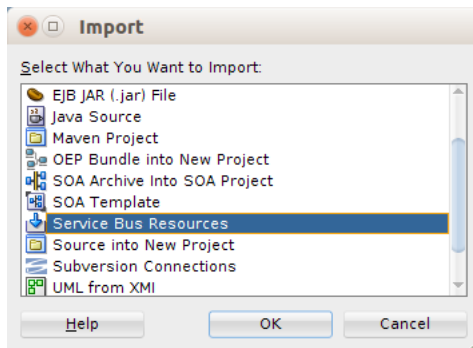
How to Import WSDL into RSB Decorator Project

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:

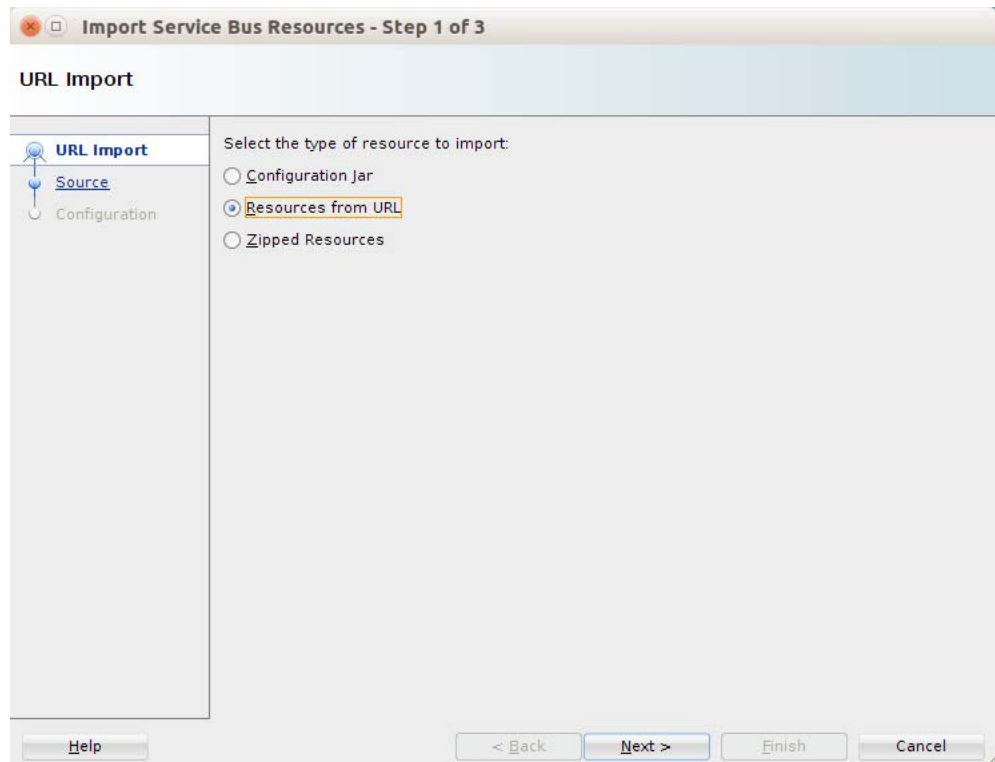
1. Right-click **WSDL** > **BusinessService**.



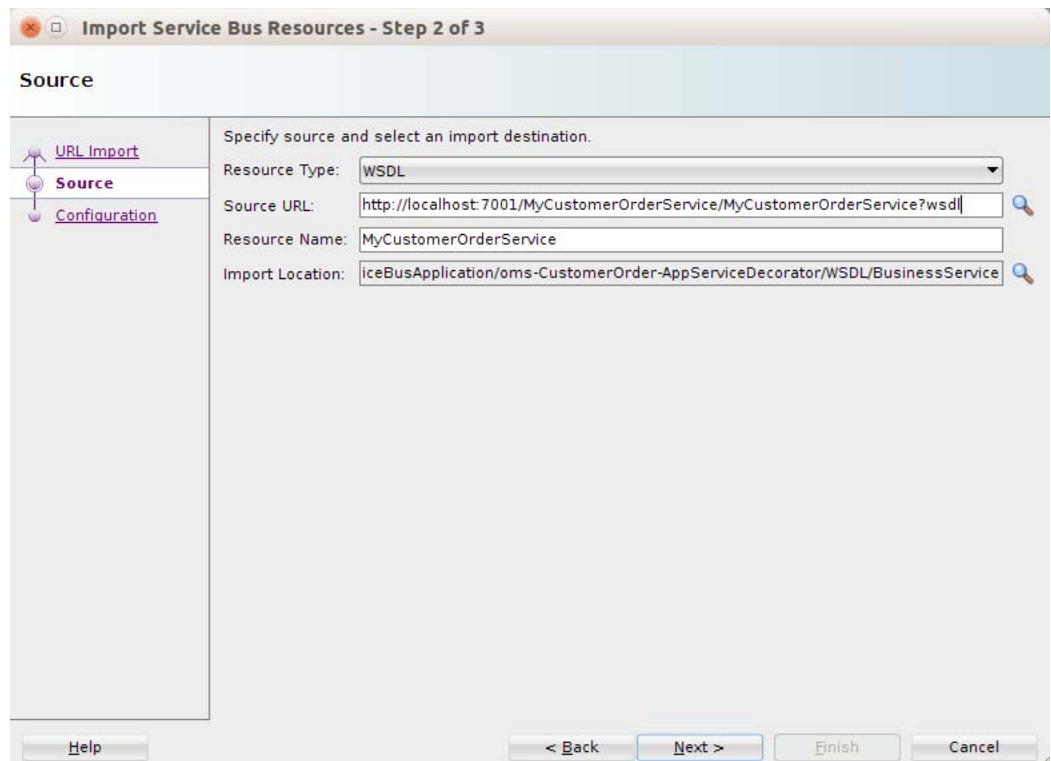
2. Select Service Bus Resources and click OK.



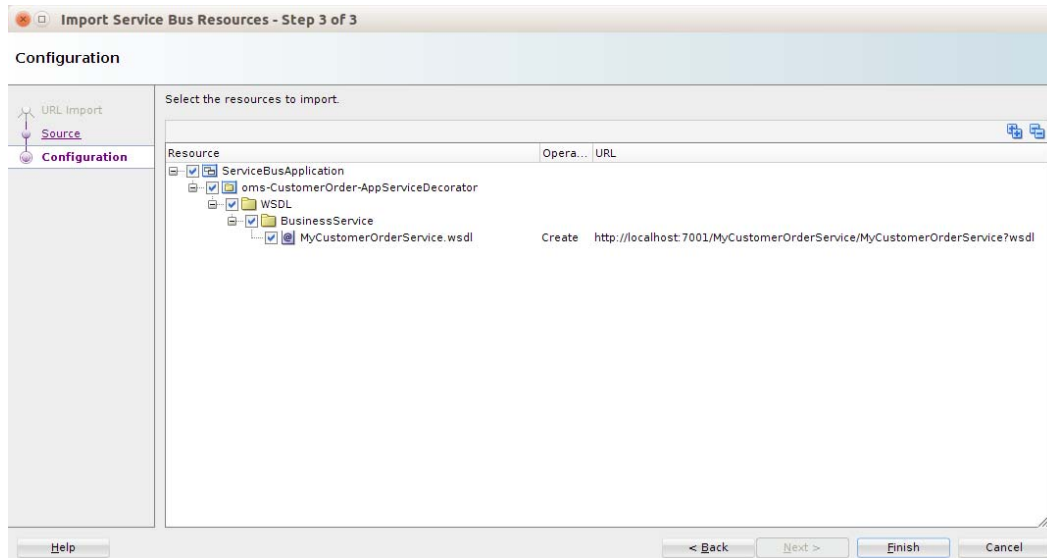
3. Select Resources from URL and click Next.



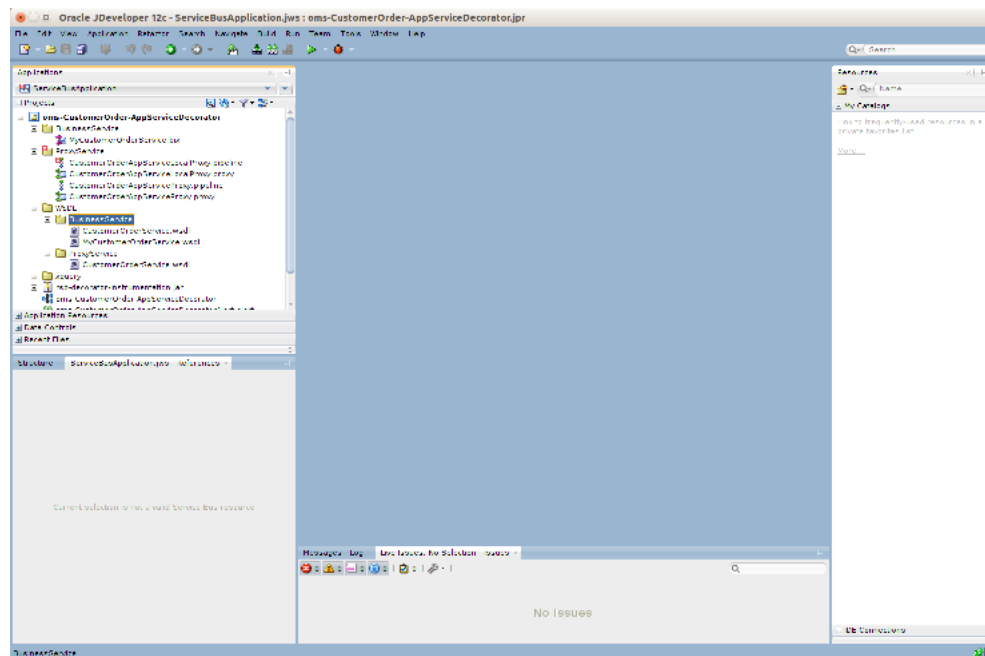
4. Select Resource Type as WSDL. Also provide the URL of the WSDL. Alternatively, if you have the WSDL downloaded to your local machine, you can browse to that location and select the WSDL file. Here open the WSDL and verify the WSDL can be successfully accessed.



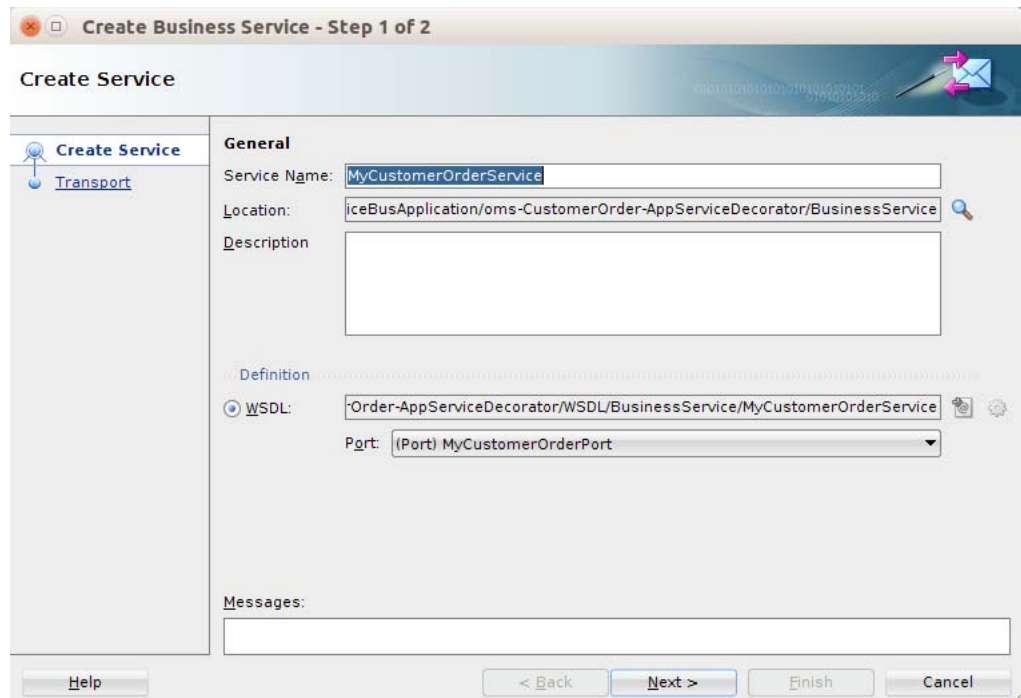
5. Click Next.



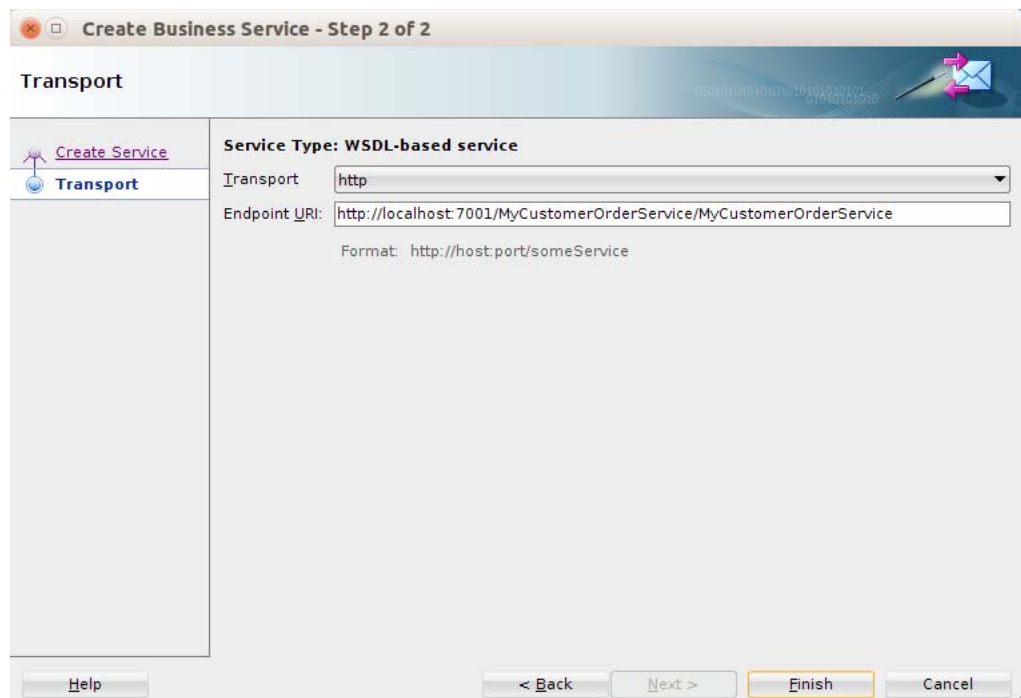
6. Click **Finish**. The new WSDL file is added to the project.



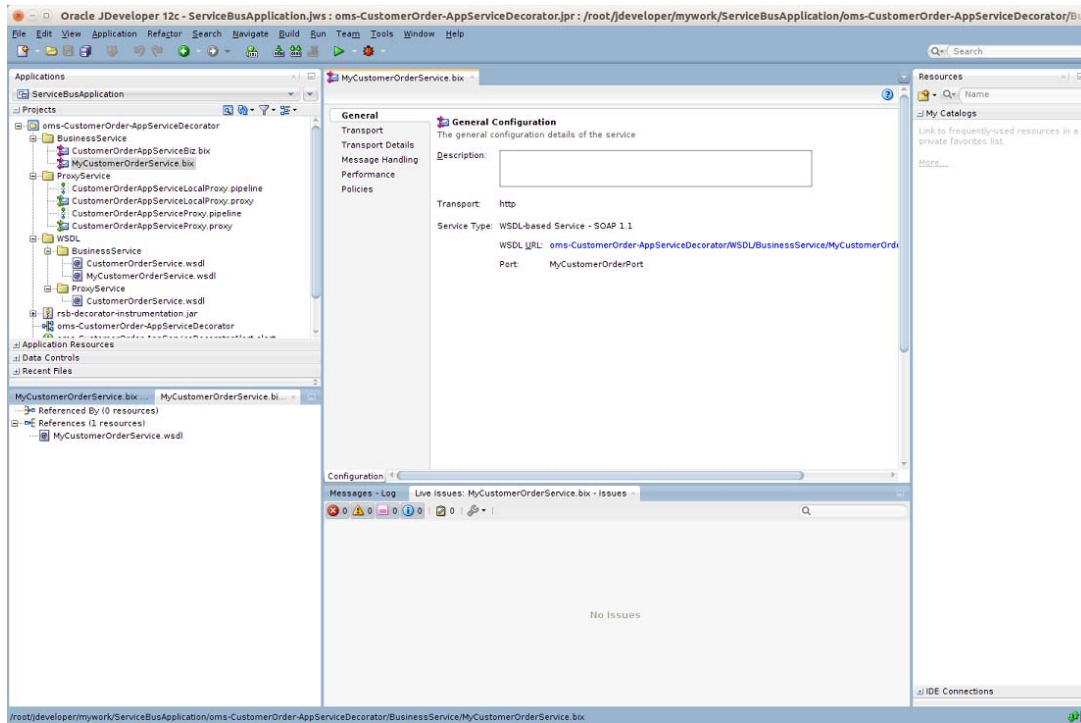
7. Select new WSDL, right click and select **Service Bus > Generate Business Service** to modify the business service.



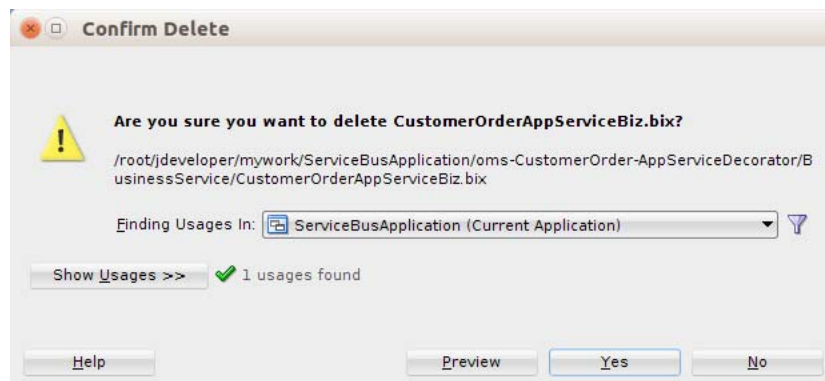
8. Change the location to the Business Service folder and click **Next**.



9. Click **Finish** to generate business service from new WSDL.



10. Select previous business service "CustomerOrderAppService.biz", right click and select **Delete**.

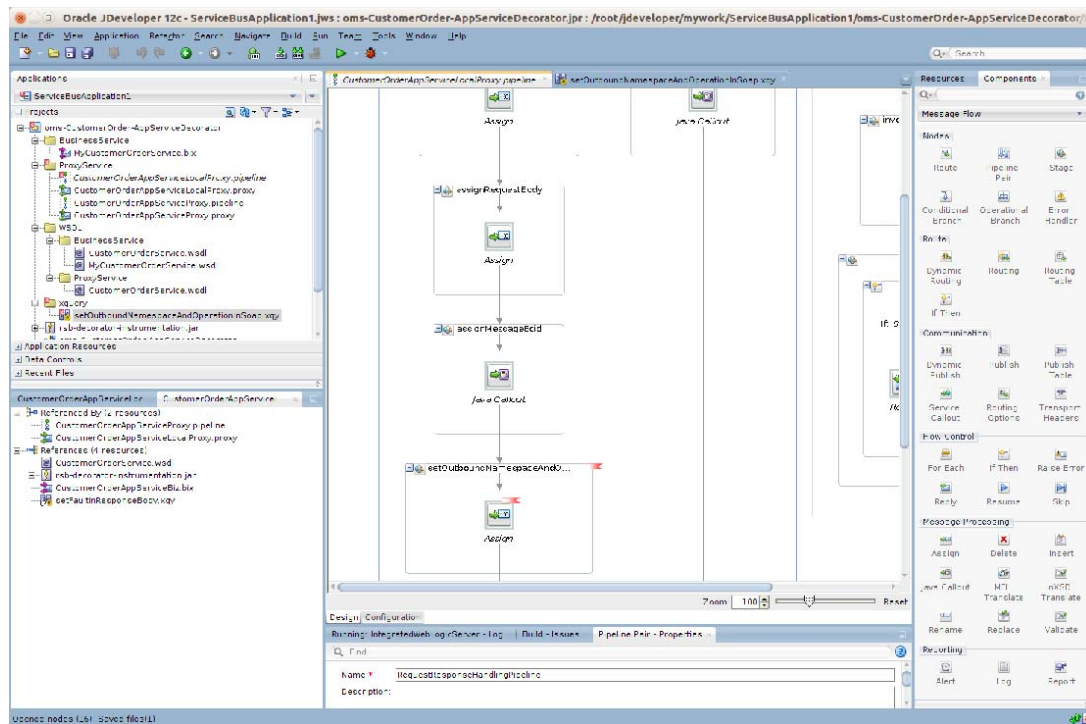


11. Click **Yes** to delete the previous business service.

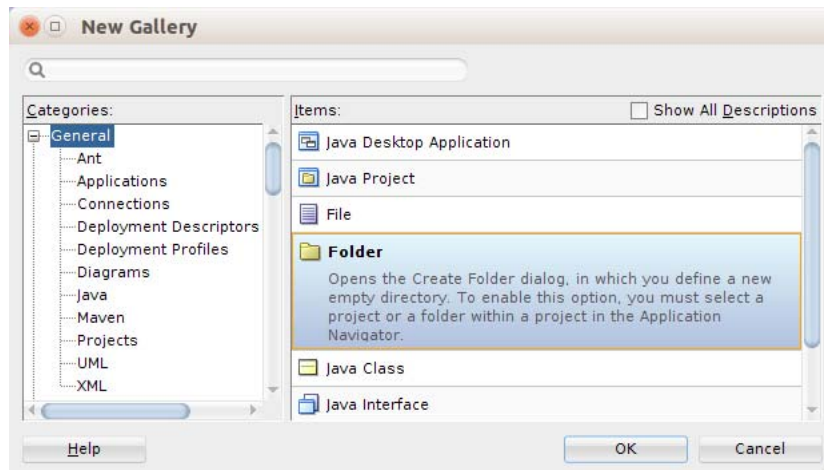
How to Map Namespaces and Operation Names

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.

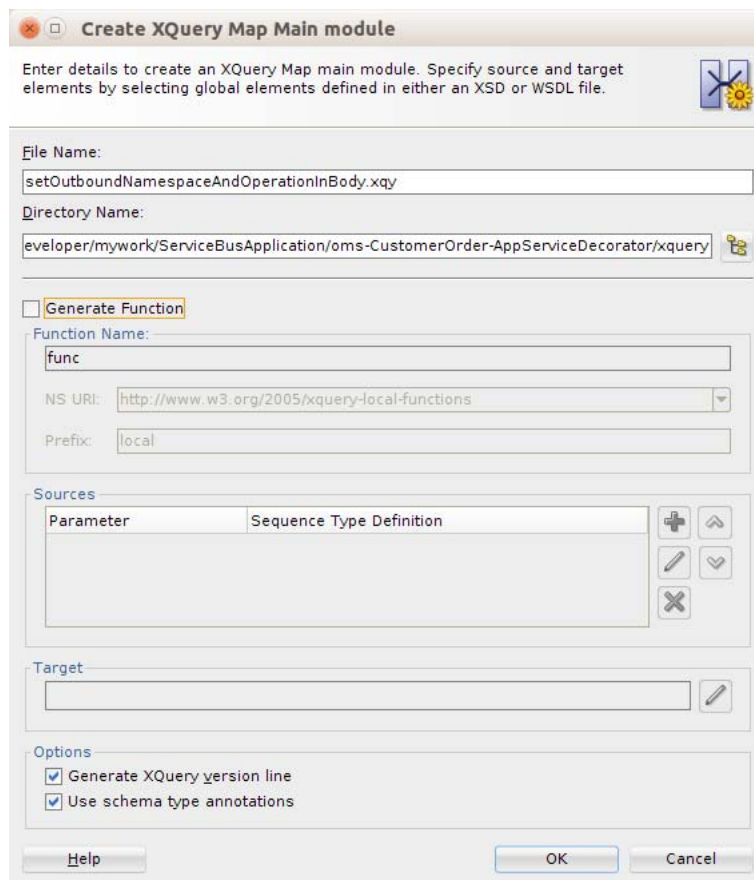
1. The first step is to add a new stage in request pipeline of local proxy service message flow. To add a new stage, drag Stage component from Components window below assignMessageEid stage. Enter the name of the stage as "setOutboundNamespaceAndOperation".



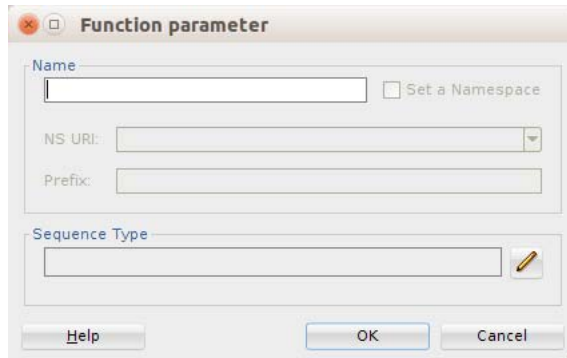
2. To create a new xquery file, you need to create a folder "xquery" where all the xquery files will be saved. To create the folder, right click the project name and select New > From Gallery > Folder. For Enter or select the parent folder, verify the AppServiceDecorator folder is selected (for example, omsc-CustomerOrder-AppServiceDecorator). Enter xquery as the folder name.



3. Right click the xquery folder and select New > Xquery File ver 1.0. Enter the name "setOutboundNamespaceAndOperationInBody".

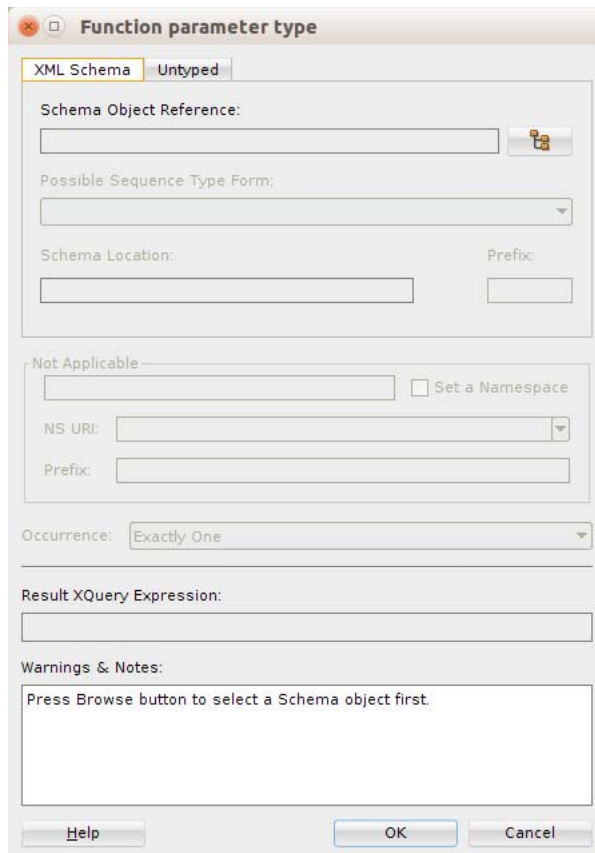


4. Select Generate Function check box, and provide Function Name "setOutboundNamespaceAndOperationInSoap". Provide NS URI (for example, http://tempuri.org/oms-CustomerOrder-AppServiceDecorator/xquery/setOutboundNamespaceAndOperationInSoap/) and Prefix as xf. Click add in the Sources section.



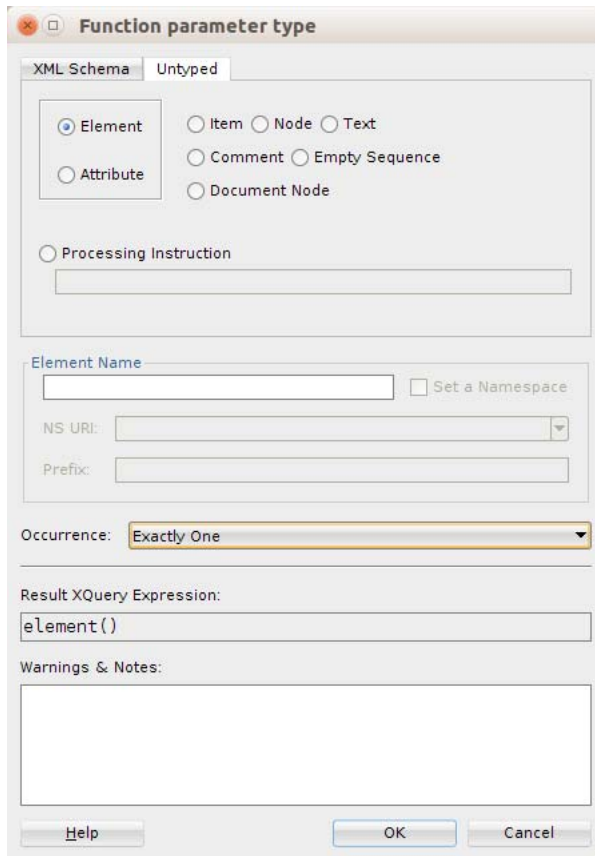
The dialog box is titled "Function parameter". It contains a "Name" text field, a "Set a Namespace" checkbox, an "NS URI" dropdown menu, a "Prefix" text field, and a "Sequence Type" text field with an edit icon. At the bottom are "Help", "OK", and "Cancel" buttons.

5. Enter Name as "soapBody" and click edit in Sequence Type section.



The dialog box is titled "Function parameter type" and has two tabs: "XML Schema" (selected) and "Untyped". It includes a "Schema Object Reference" field with a browse icon, a "Possible Sequence Type Form" dropdown, "Schema Location" and "Prefix" text fields, a "Not Applicable" section with a "Set a Namespace" checkbox, "NS URI" dropdown, and "Prefix" text field, an "Occurrence" dropdown set to "Exactly One", a "Result XQuery Expression" text field, and a "Warnings & Notes" area containing the message: "Press Browse button to select a Schema object first." At the bottom are "Help", "OK", and "Cancel" buttons.

6. Select Untyped tab, select Element and click **OK**.



7. In the **Function parameter type** window, click **OK**.

Enter details to create an XQuery Map main module. Specify source and target elements by selecting global elements defined in either an XSD or WSDL file.

File Name: untitled1.xqy

Directory Name: veloper/mywork/ServiceBusApplication1/oms-CustomerOrder-AppServiceDecorator/xquery

Generate Function

Function Name: setOutboundNamespaceAndOperationInSoap

NS URI: er-AppServiceDecorator/xquery/setOutboundNamespaceAndOperationInSoap/

Prefix: xfi

Parameter	Sequence Type Definition
\$soapBody	element()

Target:

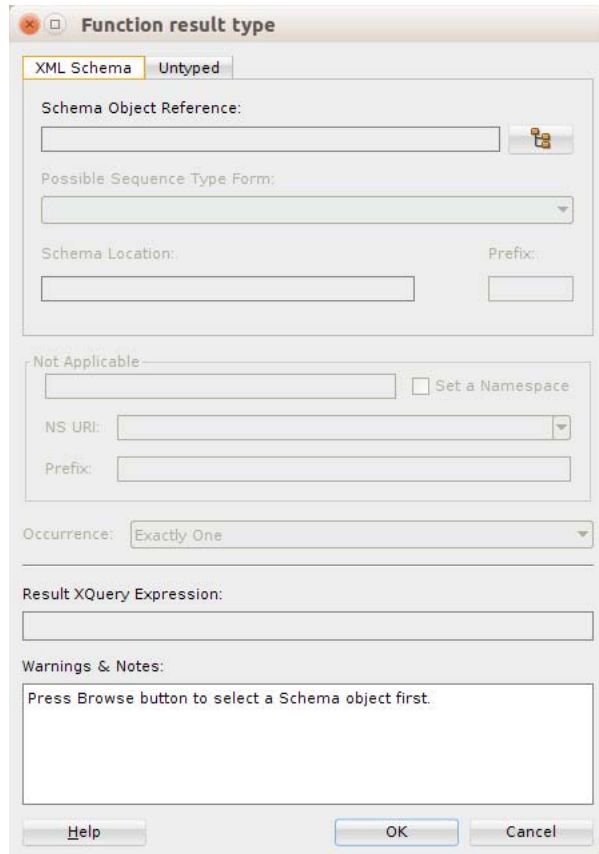
Options

Generate XQuery version line

Use schema type annotations

Help OK Cancel

8. In the **Create Xquery Map Main module** window, edit the target by clicking edit.



9. In the Untyped tab, select the Element option and click OK.

Function result type

XML Schema: Untyped

Element Attribute Item Node Text Comment Empty Sequence Document Node

Processing Instruction

Element Name: Set a Namespace

NS URI:

Prefix:

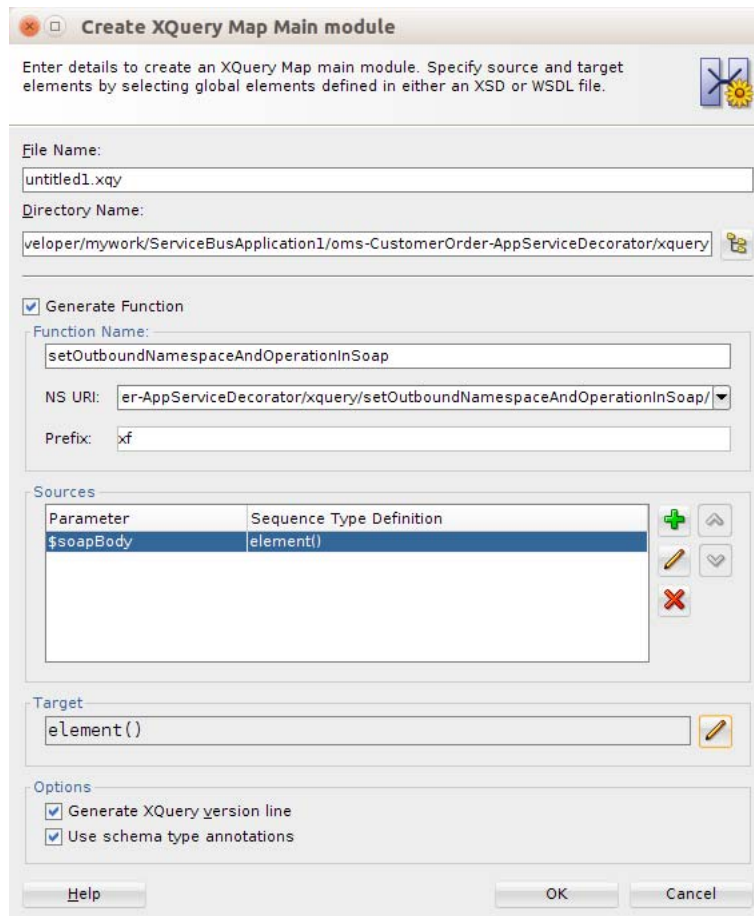
Occurrence: Exactly One

Result XQuery Expression:

Warnings & Notes:

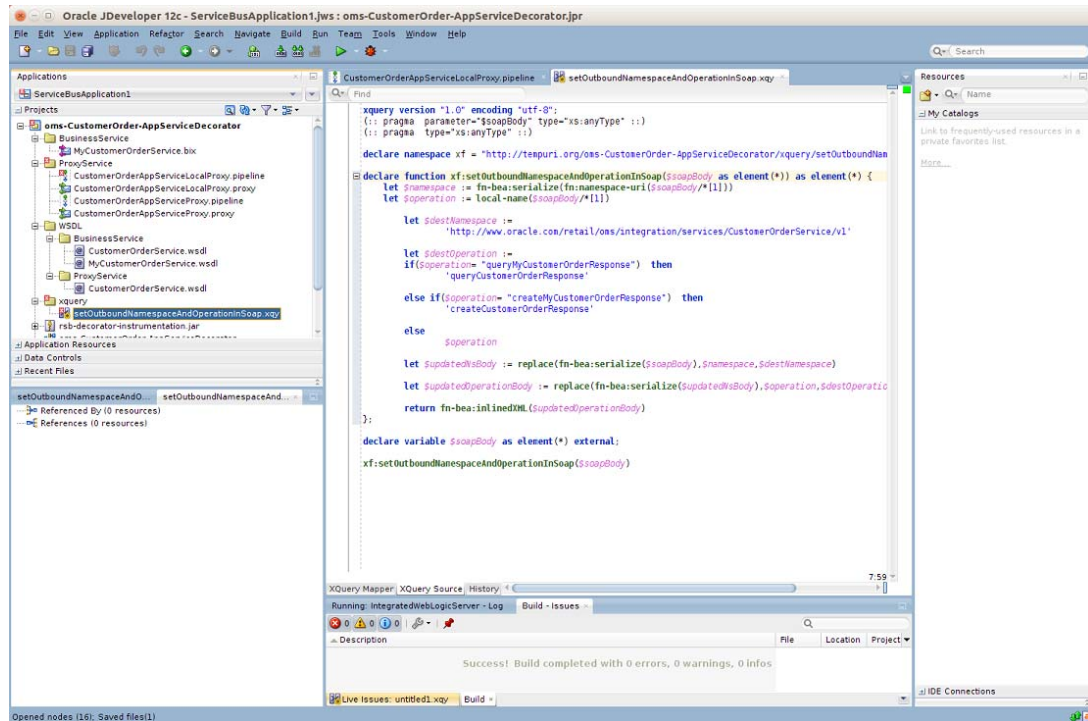
Help OK Cancel

10. Click **OK** on the **Create XQuery Map** module window.



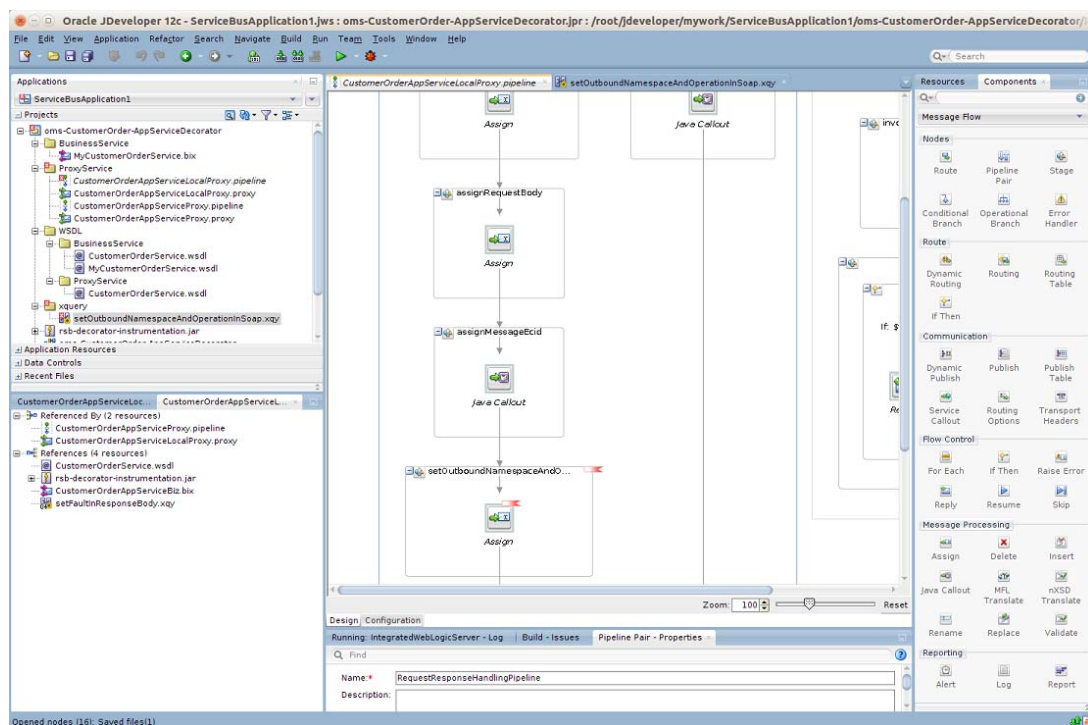
11. Click **OK**. In the source view of the file, enter the code as shown in the screenshot below. In this code, the variable \$namespace contains the namespace of the incoming request xml and \$operation contains the operation name in the incoming request. Further, we check for each incoming operation 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.

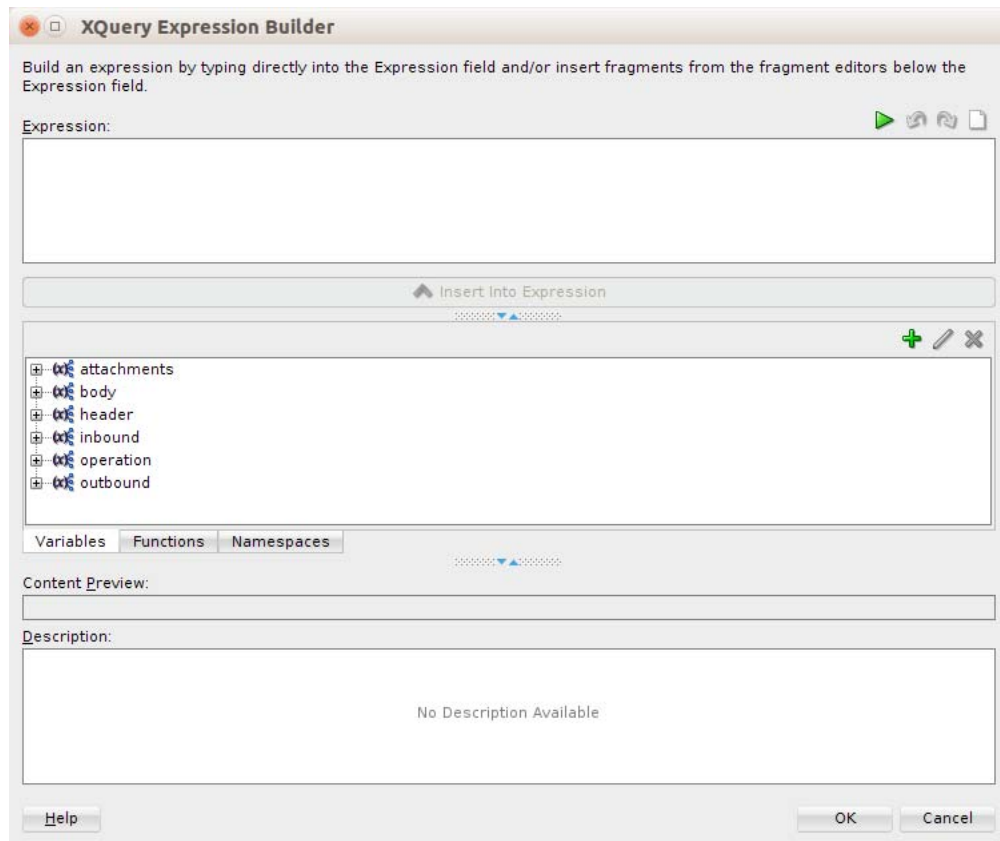


12. Return to the stage in message flow and add an Assign action. Steps to add Assign action are:

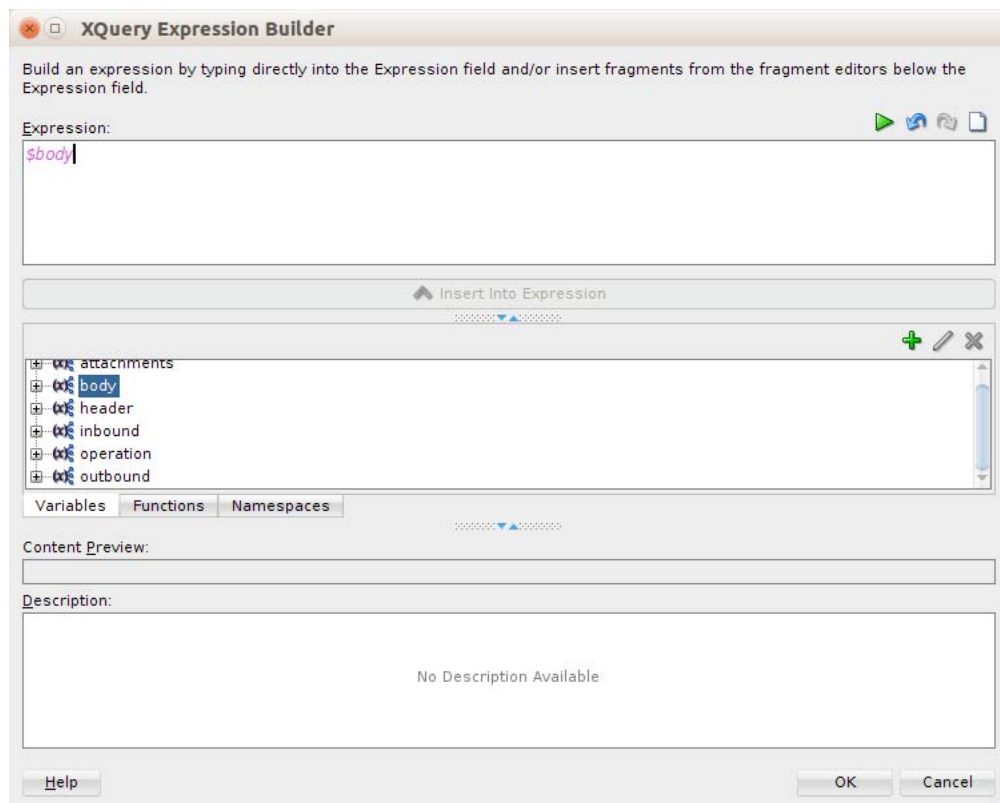
Drag assign component from Message Processing section of Components window into "setOutboundNamespaceAndOperation" stage



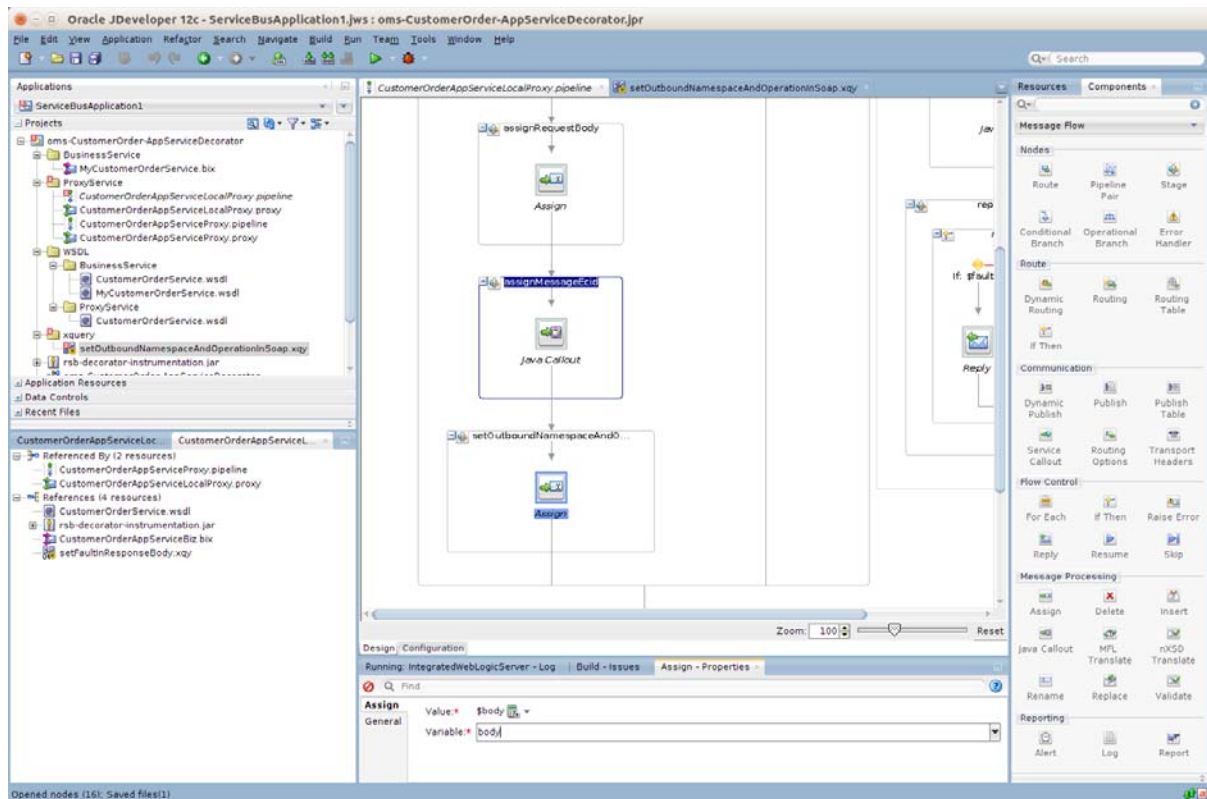
13. In the Expression field of Assign action, click the <Expression> link and go to XQuery Expression Builder window:



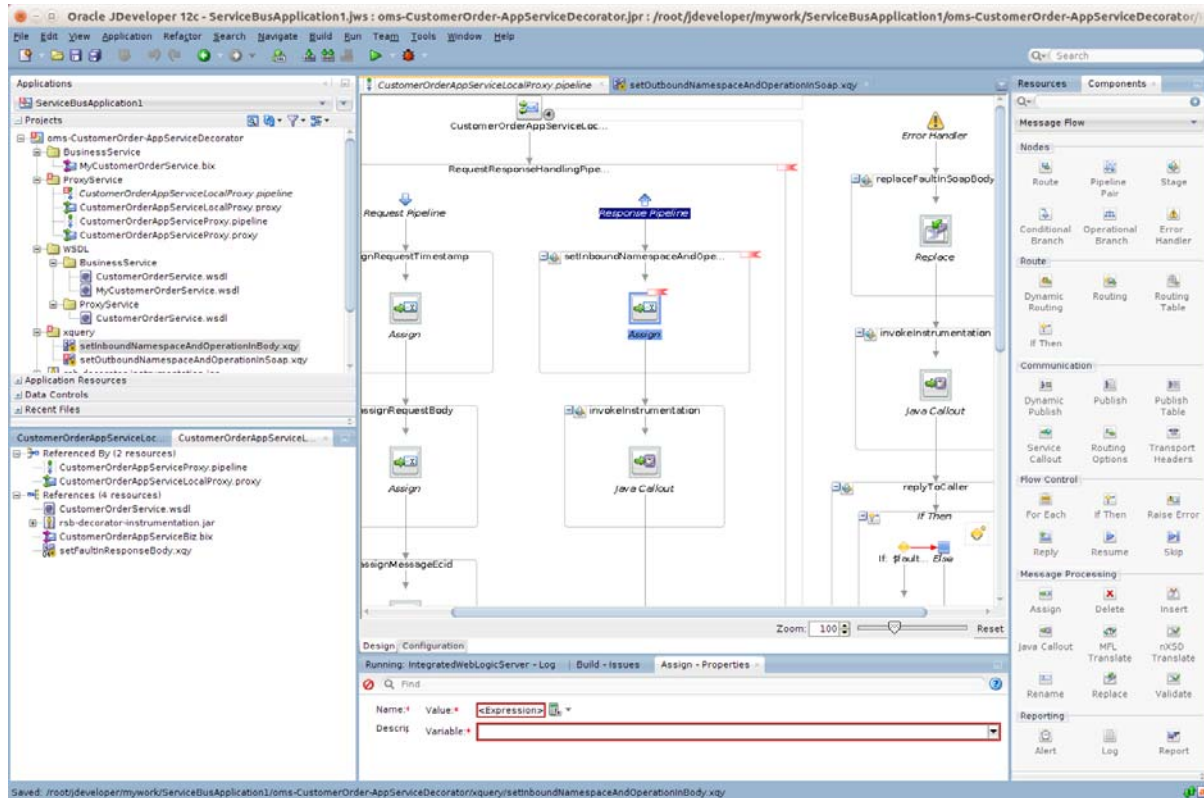
14. Enter \$body in the Expression field and click OK.



15. In the Variable field of Assign action, enter the value as body.



16. 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. Drag assign component to the response pipeline.



17. Enter the stage name as **setInboundNamespaceAndOperation**.

18. Now we need to create an xquery file to do the mapping. Right click the xquery folder and select **New > XQuery File version1.0**.

Create XQuery Map Main module

Enter details to create an XQuery Map main module. Specify source and target elements by selecting global elements defined in either an XSD or WSDL file.

File Name:
setInboundNamespaceAndOperationInBody.xqy

Directory Name:
veloper/mywork/ServiceBusApplication1/oms-CustomerOrder-AppServiceDecorator/xquery

Generate Function

Function Name:
setInboundNamespaceAndOperationInBody

NS URI:
rder-AppServiceDecorator/xquery/setInboundNamespaceAndOperationInBody/

Prefix:
xf

Sources

Parameter	Sequence Type Definition

Target

Options

Generate XQuery version line

Use schema type annotations

Help OK Cancel

19. Enter file name as **setInboundNamespaceAndOperationInBody**. Select Generate Function checkbox. Enter NS URI (for example, <http://tempuri.org/oms-CustomerOrder-AppServiceDecorator/xquery/setInboundNamespaceAndOperationInBody/>) and prefix (for example, xf). Click add in Sources section.

Function parameter

Name

Set a Namespace

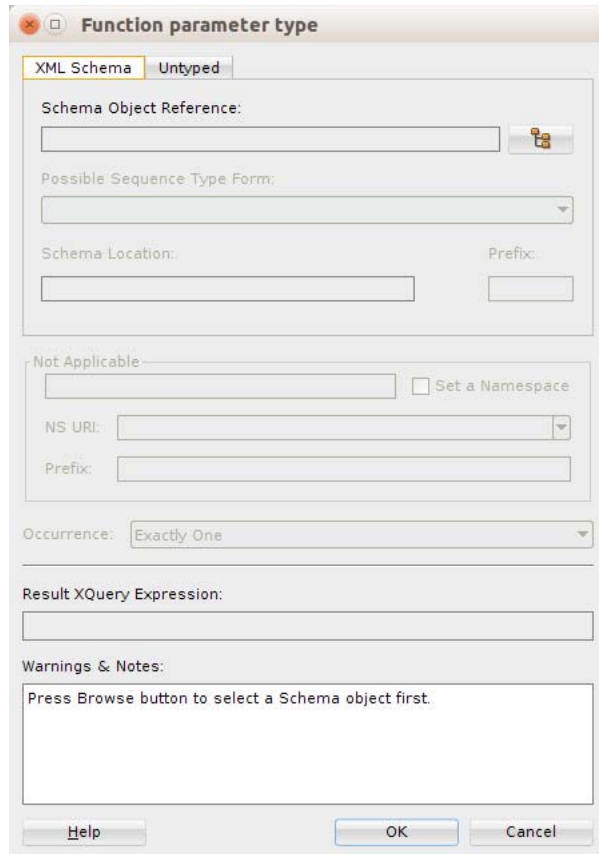
NS URI:

Prefix:

Sequence Type

Help OK Cancel

20. Enter "soapBody" in the Name field. Click edit in Sequence Type section:



21. Navigate to the Untyped tab, select Element and click **OK**.

Create XQuery Map Main module

Enter details to create an XQuery Map main module. Specify source and target elements by selecting global elements defined in either an XSD or WSDL file.

File Name:
setInboundNamespaceAndOperationInBody.xqy

Directory Name:
veloper/mywork/ServiceBusApplication1/oms-CustomerOrder-AppServiceDecorator/xquery

Generate Function

Function Name:
setInboundNamespaceAndOperationInBody

NS URI: rder-AppServiceDecorator/xquery/setInboundNamespaceAndOperationInBody/

Prefix: xf

Sources

Parameter	Sequence Type Definition
-----------	--------------------------

Target:

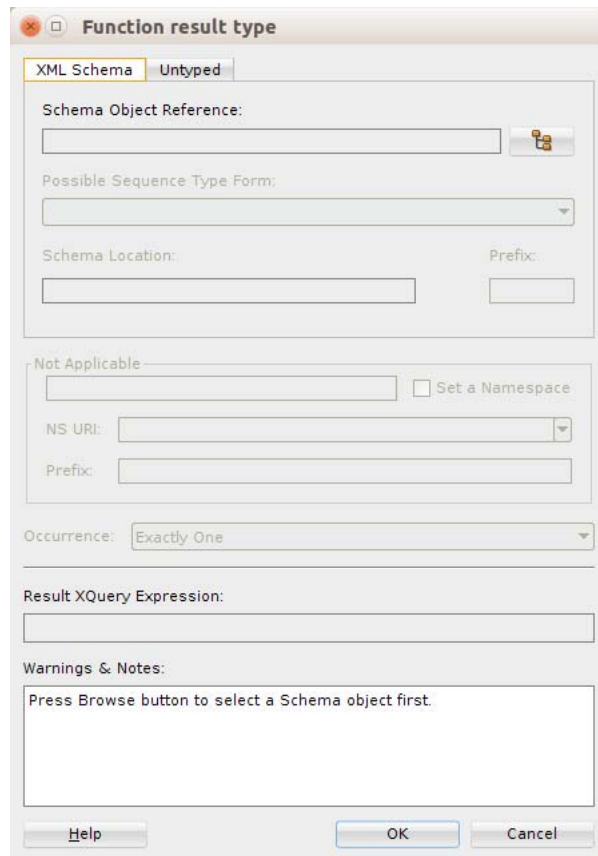
Options

Generate XQuery version line

Use schema type annotations

Help OK Cancel

22. Click edit in the Target section.



The image shows a dialog box titled "Function result type" with a close button (X) and a maximize button (square) in the top-left corner. The dialog has two tabs: "XML Schema" (selected) and "Untyped".

Under the "XML Schema" tab, there are several sections:

- Schema Object Reference:** A text input field followed by a "Browse" button (represented by a folder icon).
- Possible Sequence Type Form:** A dropdown menu.
- Schema Location:** A text input field.
- Prefix:** A text input field.

Below these fields is a section labeled "Not Applicable" with a text input field and a checkbox labeled "Set a Namespace".

Underneath is another section with "NS URI:" and "Prefix:" labels, each followed by a dropdown menu.

Below that is an "Occurrence:" dropdown menu set to "Exactly One".

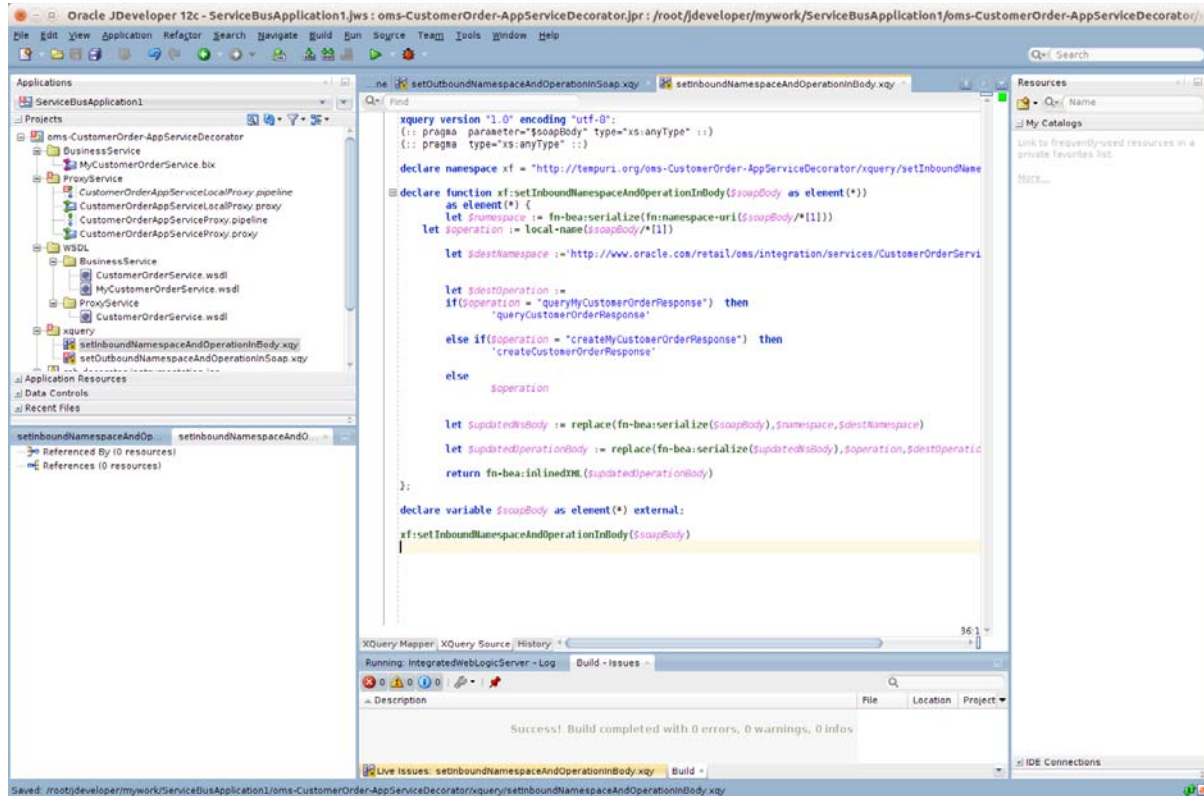
Next is a "Result XQuery Expression:" label followed by a text input field.

At the bottom is a "Warnings & Notes:" section with a text area containing the message: "Press Browse button to select a Schema object first."

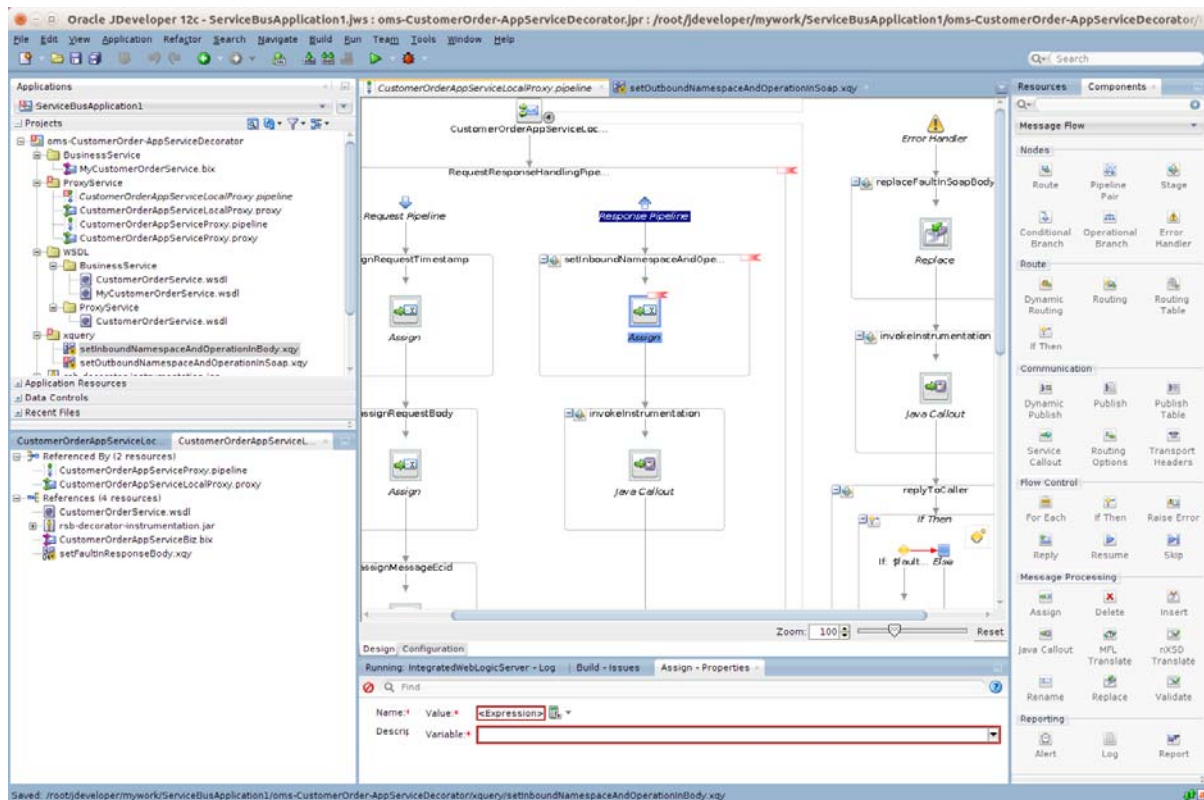
At the very bottom of the dialog are three buttons: "Help", "OK", and "Cancel".

23. Navigate to the Untyped tab, select Element, and click **OK**.

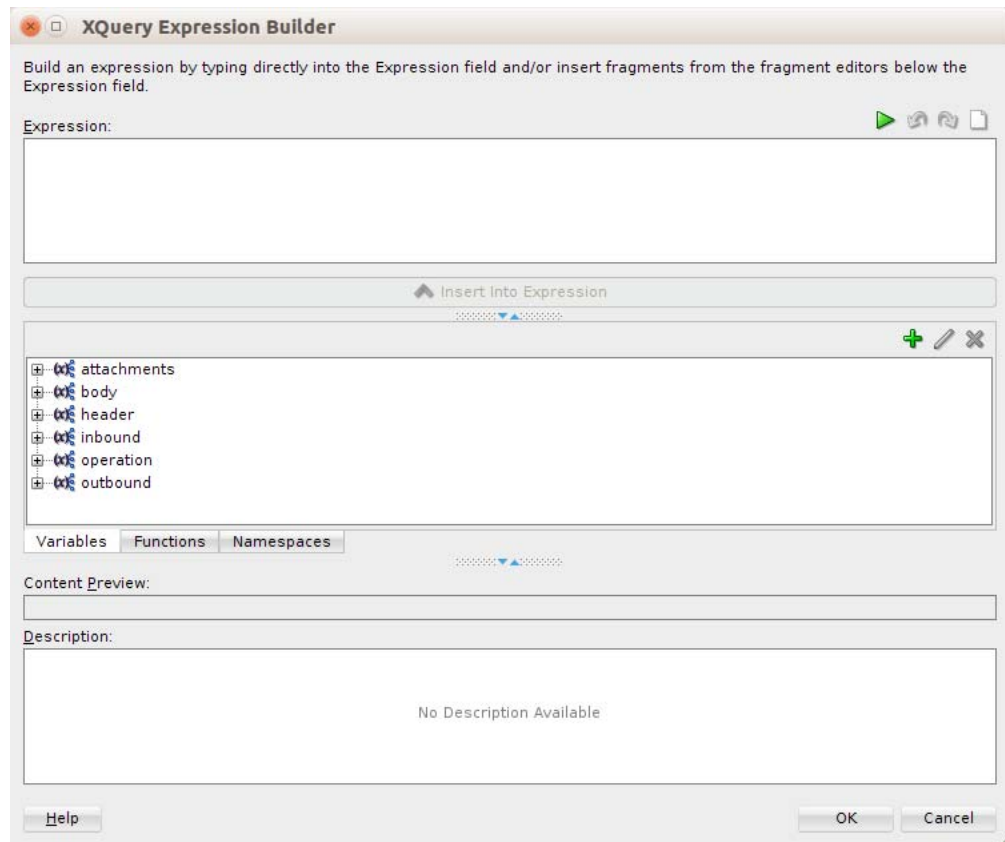
24. Navigate to the Source tab of the xquery file. Enter the code as shown below. In this code, the variable `$namespace` contains the namespace of the response xml and `$operation` contains the operation name in the response. Further, we check for each operation name and assign the corresponding proxy 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.



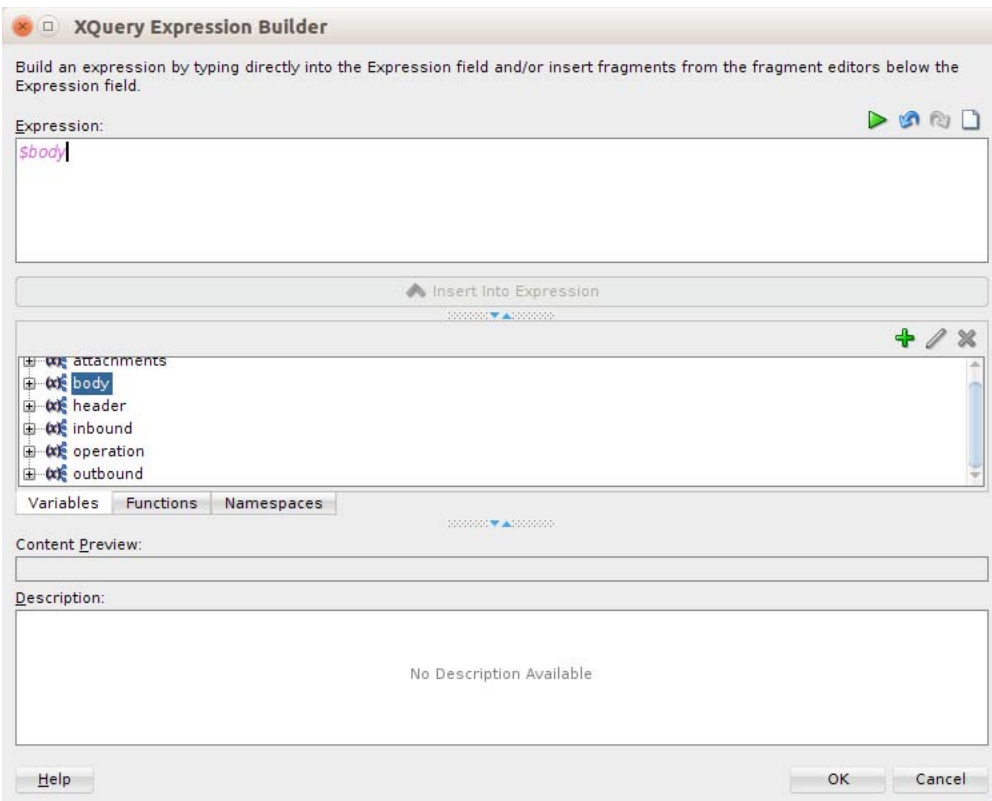
25. In the message flow, in the `setInboundNamespaceAndOperation` stage add an Assign action by dragging Assign component from Message Processing section of Components window.



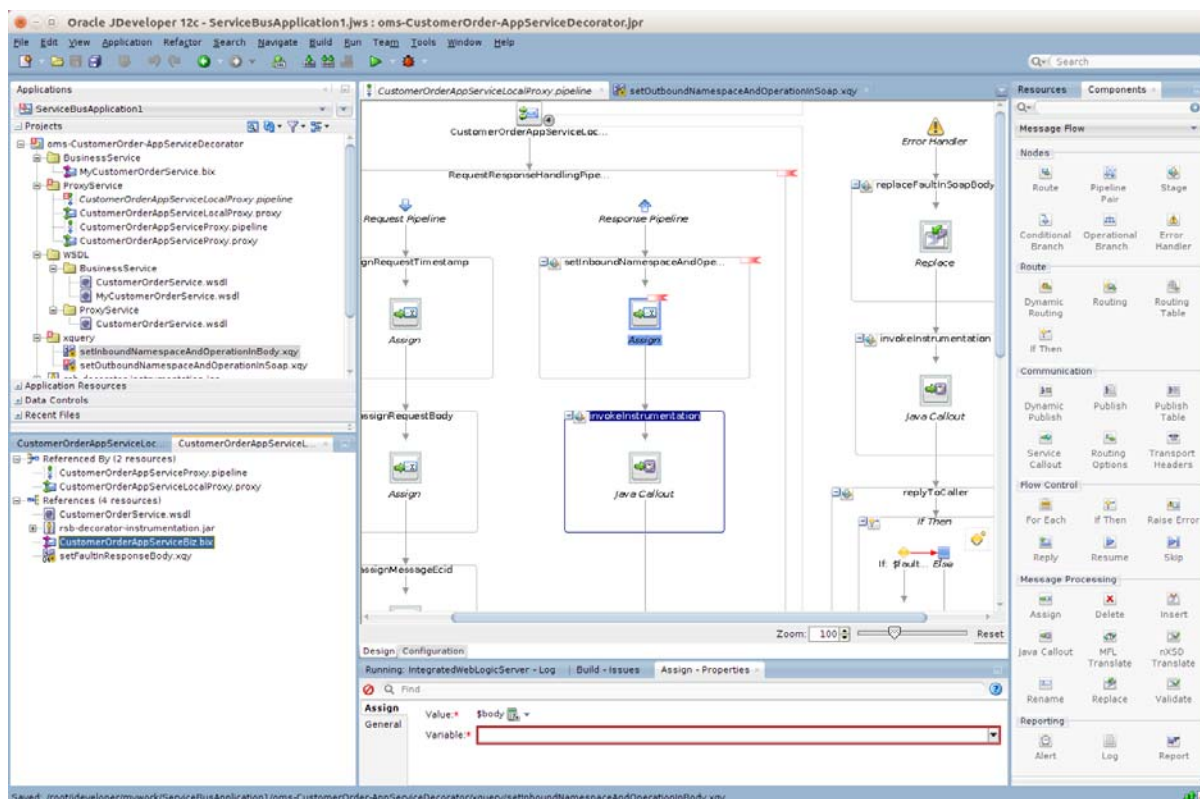
26. In the Properties window of Assign action, click the **Expression** link to open the Expression builder.



27. Enter \$body in Expression field and click OK.



28. Click OK and type "body" in the variable field.



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 XQuery File Ver 1.0**.

Create XQuery Map Main module

Enter details to create an XQuery Map main module. Specify source and target elements by selecting global elements defined in either an XSD or WSDL file.

File Name: CustOrderDescToMyCustOrderDescMapping.xqy

Directory Name: veloper/mywork/ServiceBusApplication1/oms-CustomerOrder-AppServiceDecorator/xquery

Generate Function

Function Name: CustOrderDescToMyCustOrderDescMapping

NS URI: rder-AppServiceDecorator/xquery/setinboundNamespaceAndOperationInBody/

Prefix: xf1

Sources

Parameter	Sequence Type Definition

Target

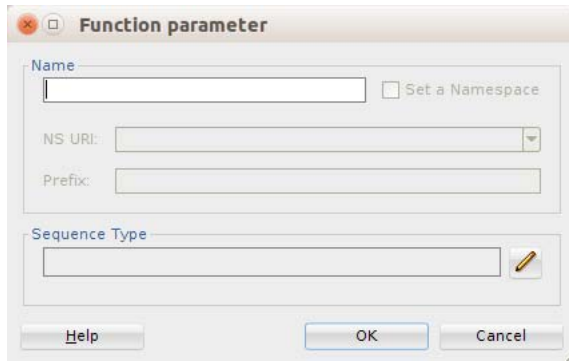
Options

Generate XQuery version line

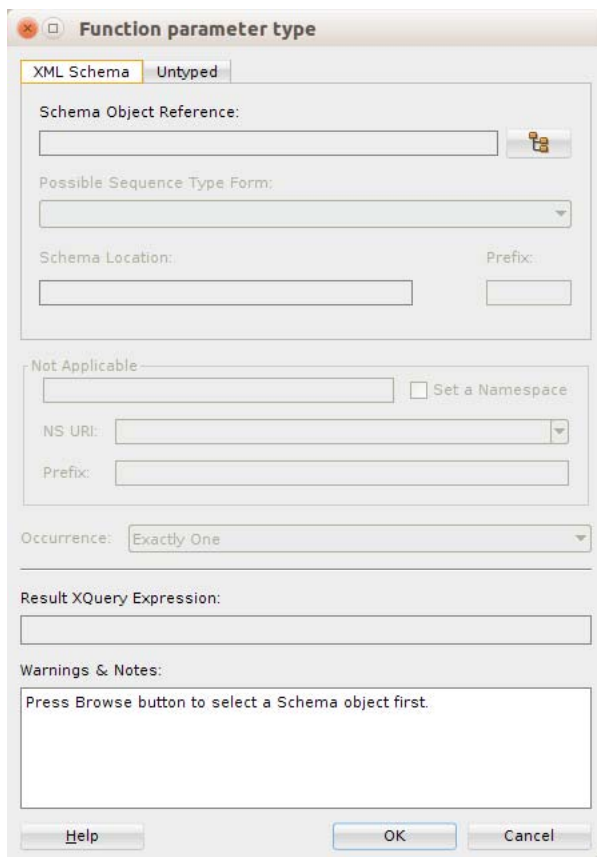
Use schema type annotations

Help OK Cancel

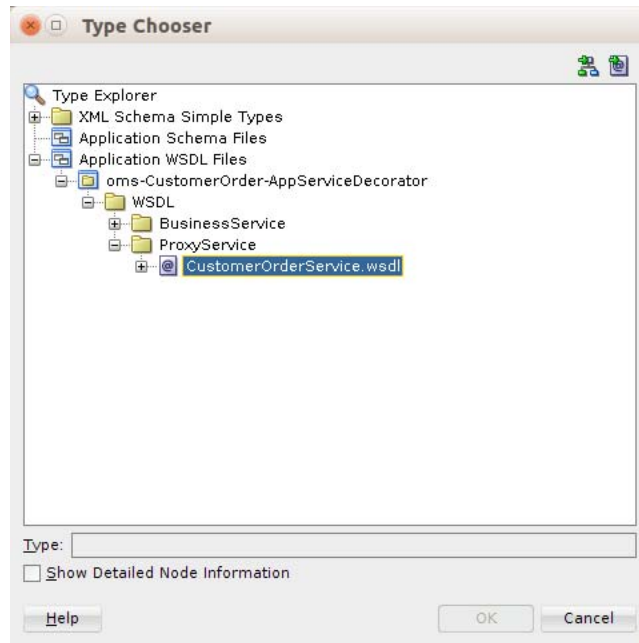
2. Enter file name as CustOrderDescToMyCustOrderDescMapping. Select Generate Function. Enter the NS URI and Prefix.
3. In the Sources section, click add.



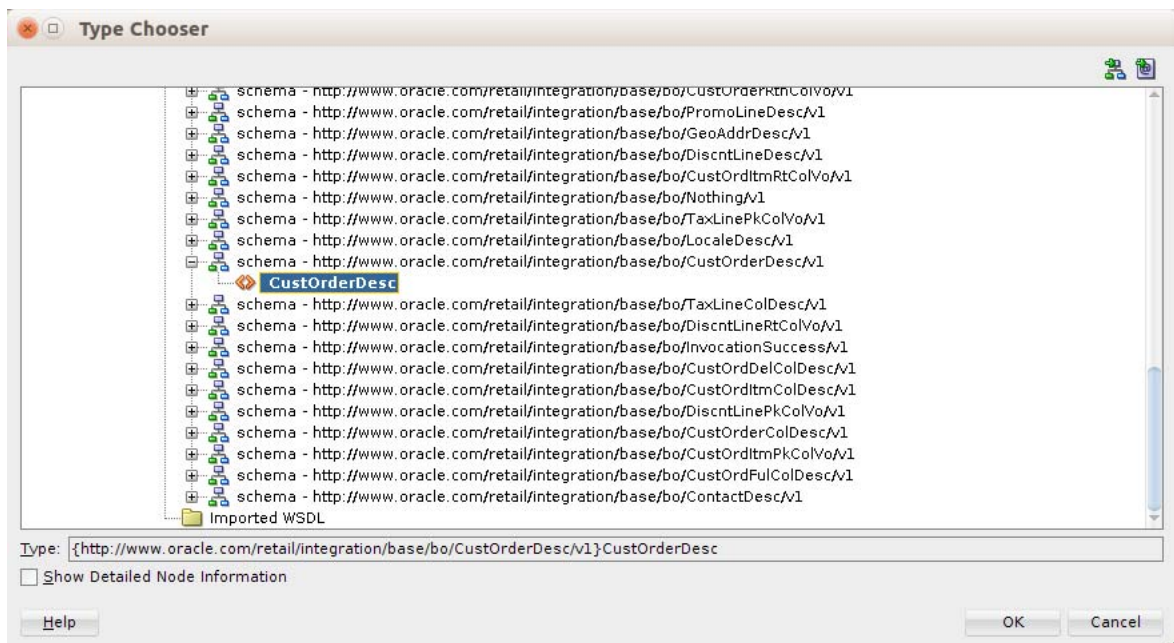
4. Enter "payload" in the Name field. Click edit in the Sequence Type section.



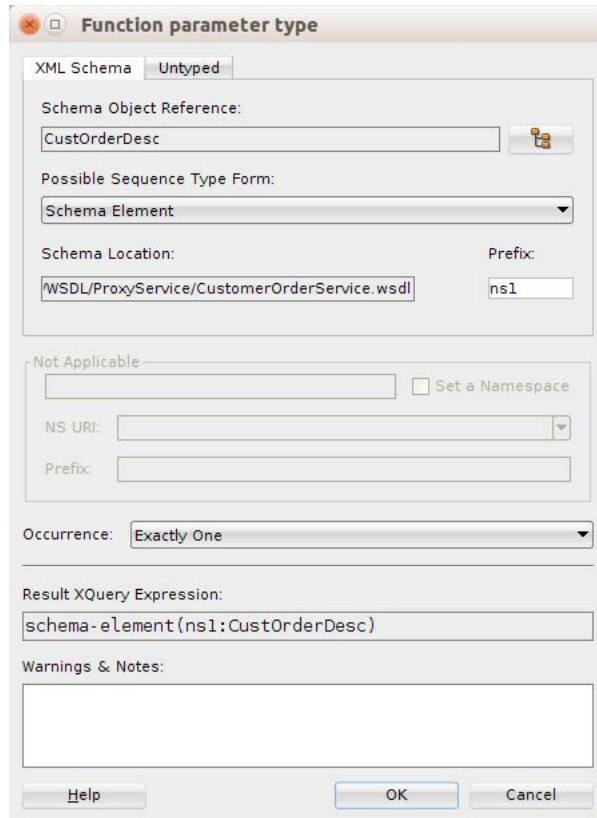
5. Click browse to view schema object.



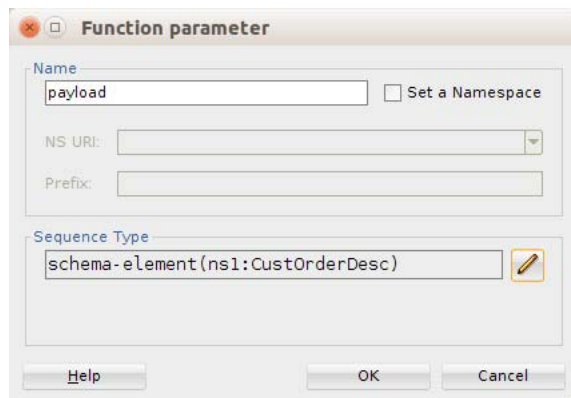
6. Select CustOrderDesc element from ProxyService WSDL which is CustomerOrderService.wsdl and click **OK**.



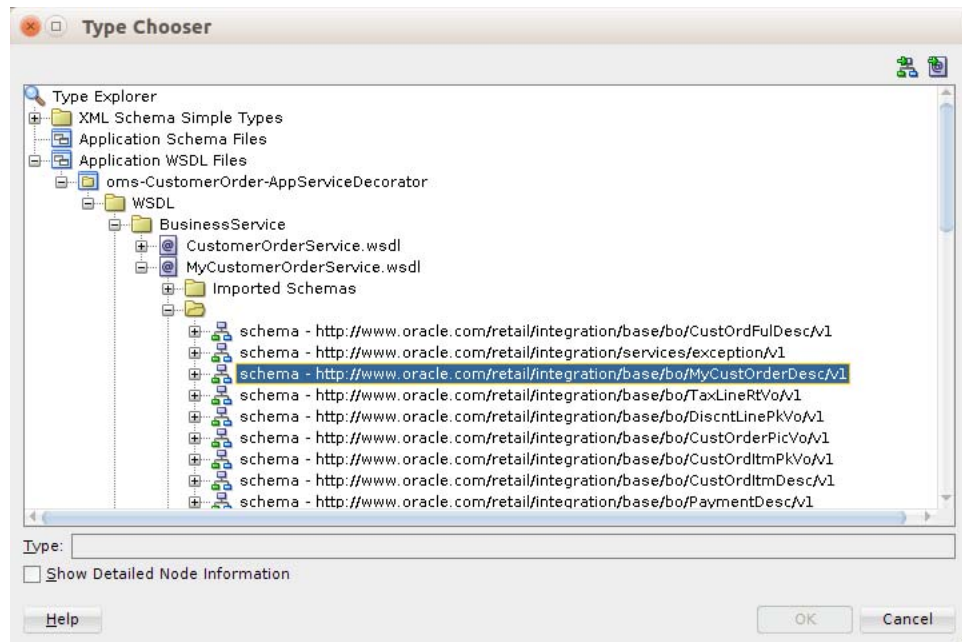
7. On the Type Chooser window, click **OK**.
8. On the Function parameter type window, click **OK**.



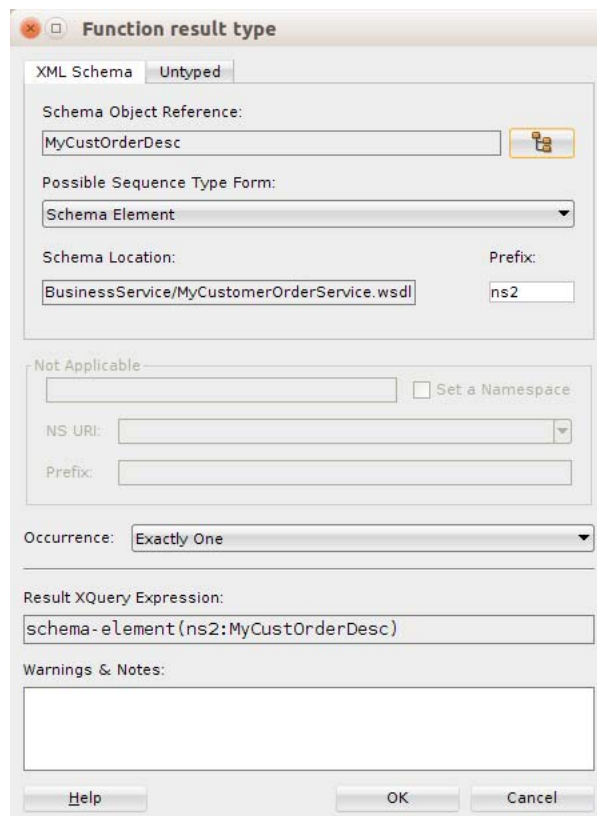
9. Click edit in the Target section, in the Function Parameter Window.



10. Select MyCustOrderDesc element and click OK.



11. On the Function result type window, click OK.



12. On the Create XQuery Map Main module window, click OK.

Enter details to create an XQuery Map main module. Specify source and target elements by selecting global elements defined in either an XSD or WSDL file.

File Name:
CustOrderDescToMyCustOrderDescMapping.xqy

Directory Name:
veloper/mywork/ServiceBusApplication1/oms-CustomerOrder-AppServiceDecorator/xquery

Generate Function

Function Name:
CustOrderDescToMyCustOrderDescMapping

NS URI: rder-AppServiceDecorator/xquery/setInboundNamespaceAndOperationInBody/

Prefix: xf1

Sources

Parameter	Sequence Type Definition
\$payload	schema-element(ns1:CustOrderDesc)

Target
schema-eLement (ns2:MyCustOrderDesc)

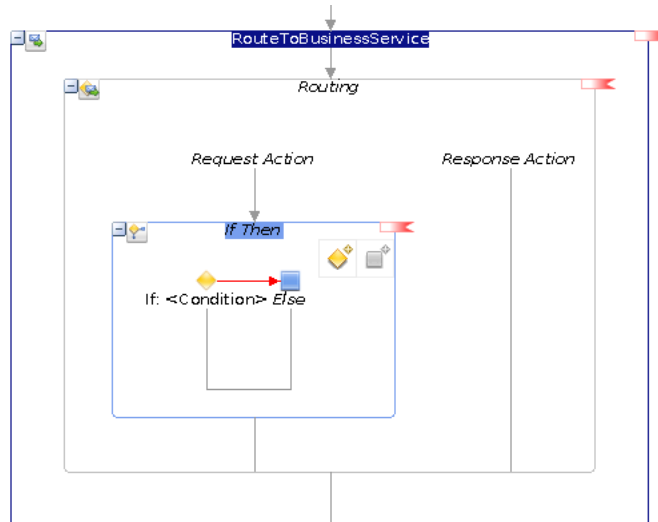
Options

Generate XQuery version line

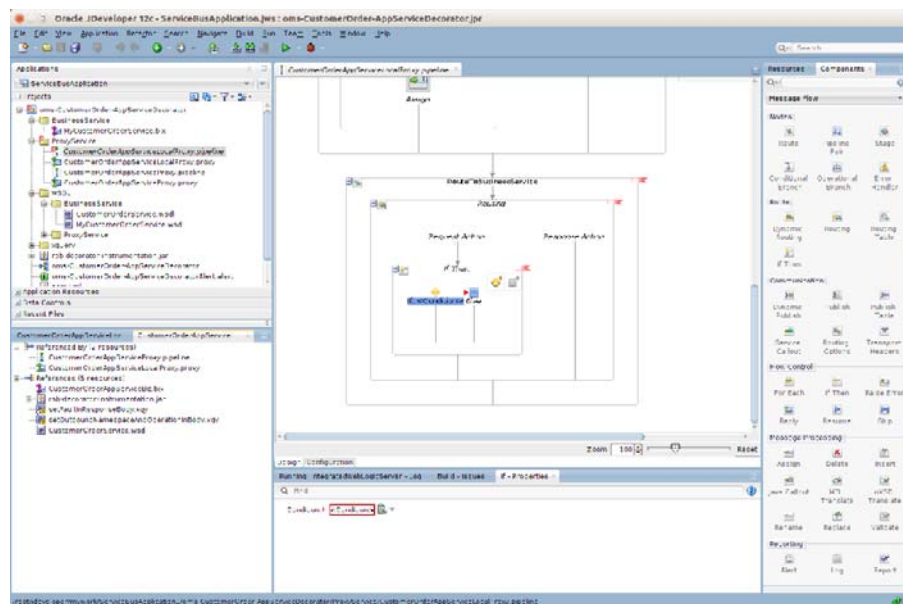
Use schema type annotations

Help OK Cancel

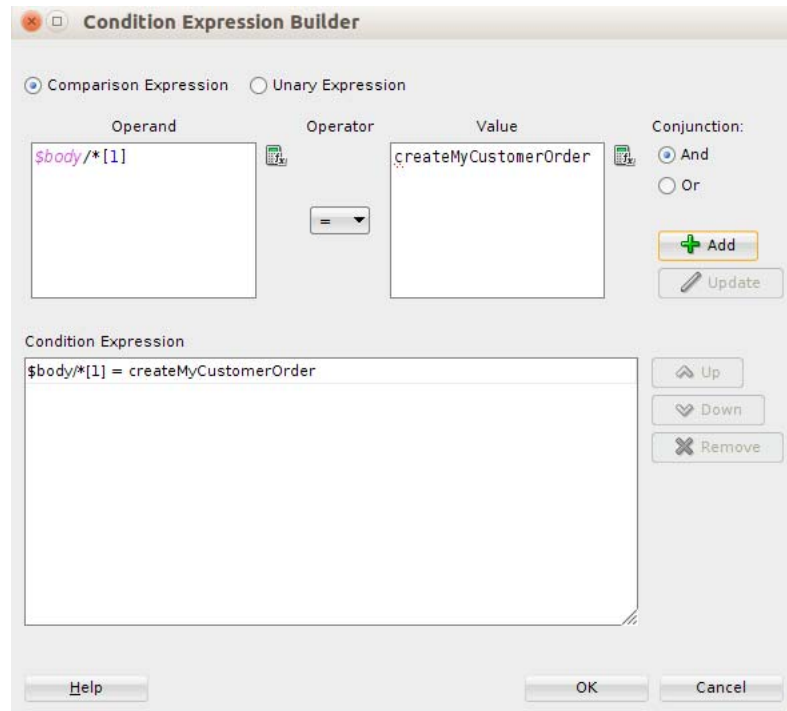
13. Click **OK**.
14. For the fields which are not auto-mapped, we need to map them manually. Drag and connect those fields one by one. You may have to write xquery functions for complex mapping. For more information, see the XQuery documentation.
15. Once the mapping in xquery file is complete, navigate to the Routing node and in the Request pipeline, drag the "If Then" component, from Components window, to the request pipeline.



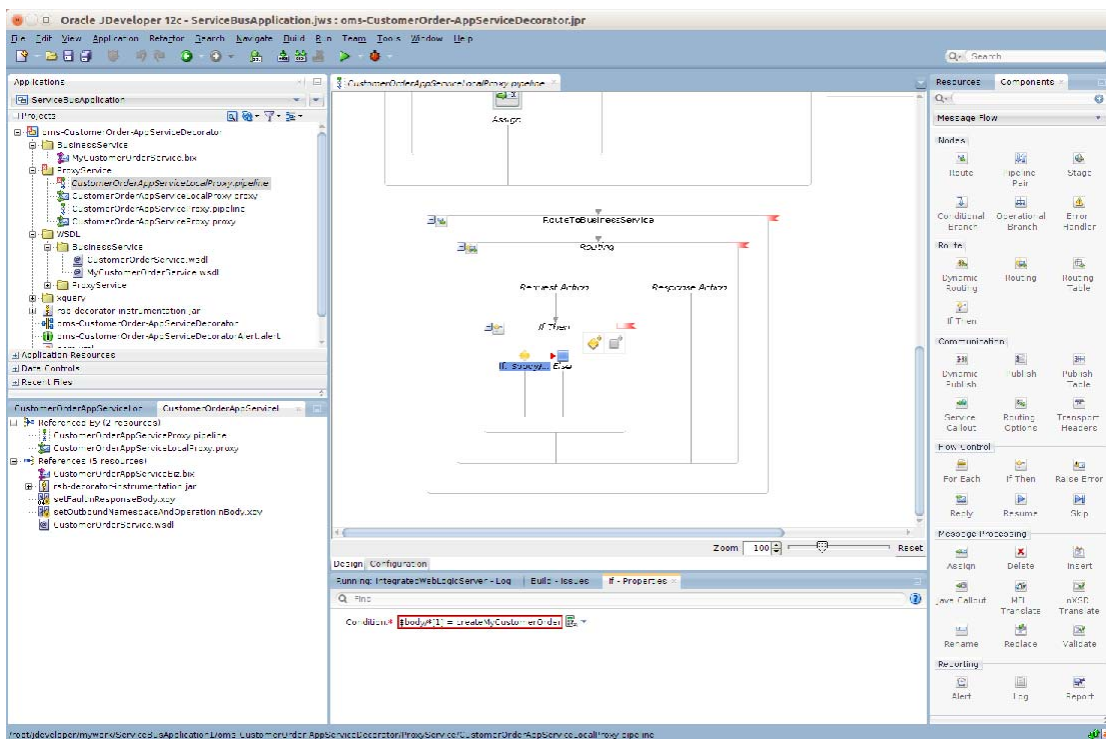
16. Navigate to the Properties window for the first "If" condition, in the If-Properties window.



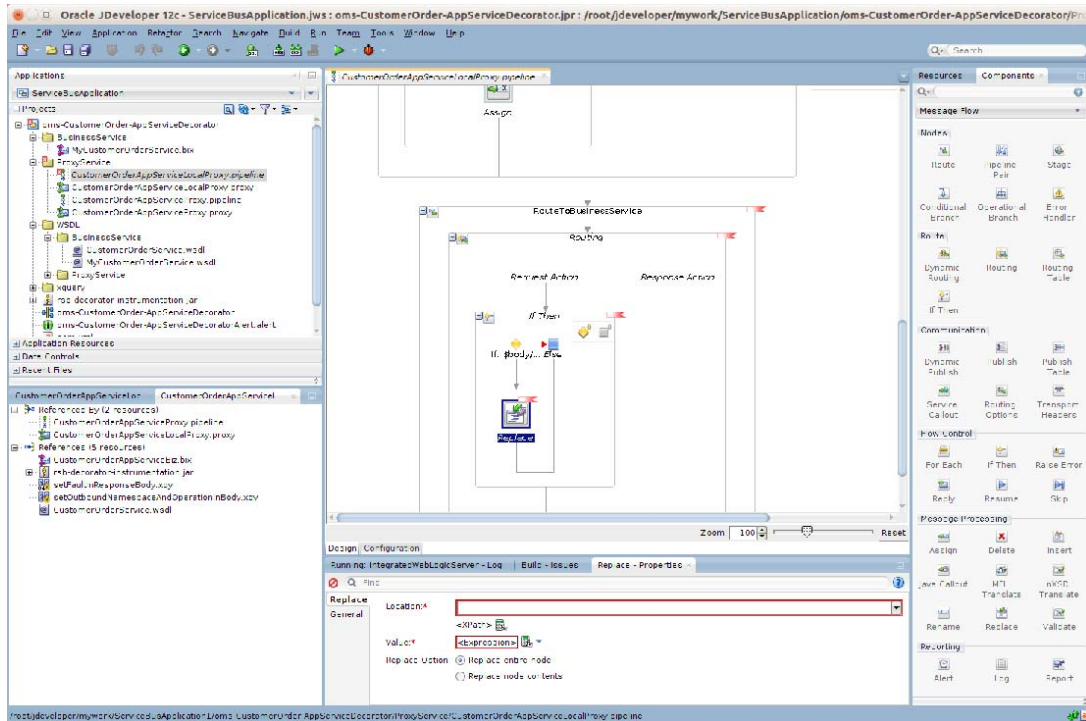
17. Click the <Condition> link and navigate to the Condition Expression Builder window. Here we need to build the condition for payload mapping. We will check for operation name to build the condition. Enter local-name(\$body/*[1]) in the Operand field, which gives the operation name in request xml.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



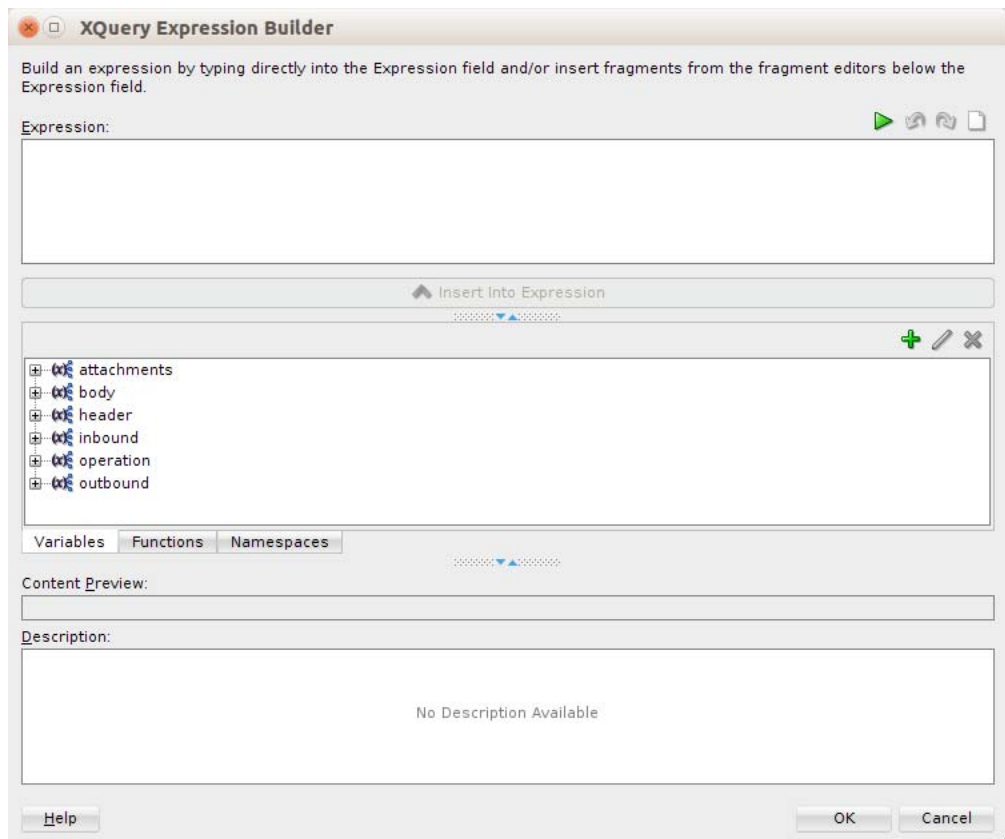
18. Click OK to add condition to the Condition Expression.



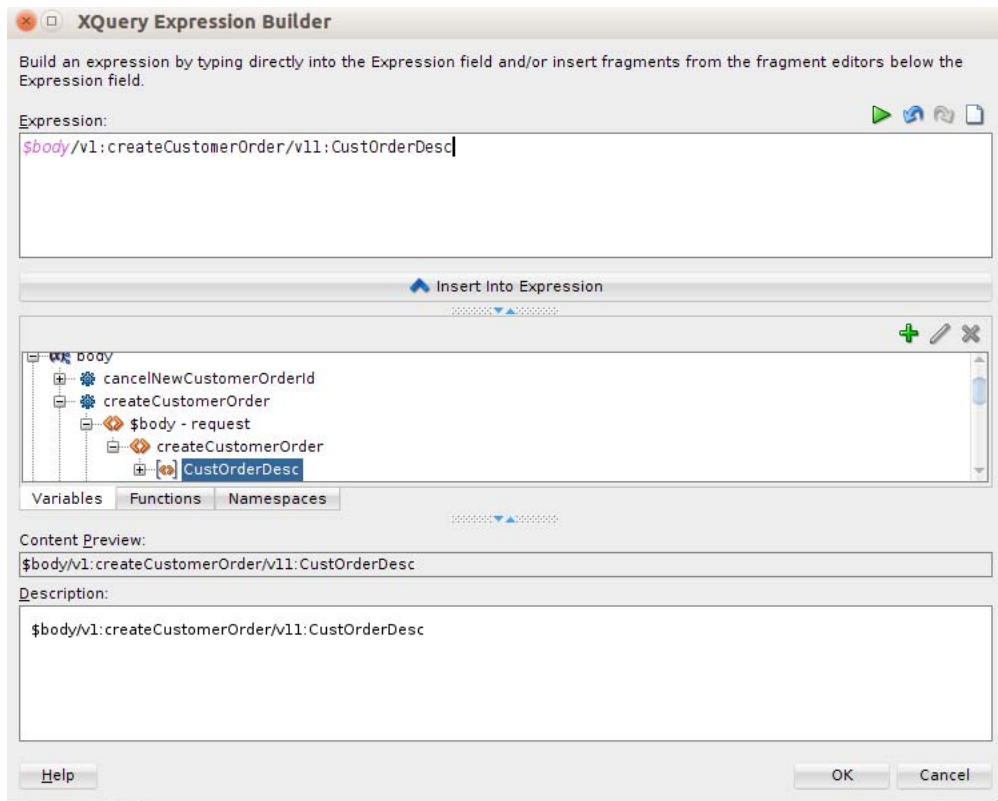
19. Add a Replace action for the first condition by dragging Replace component from components window.



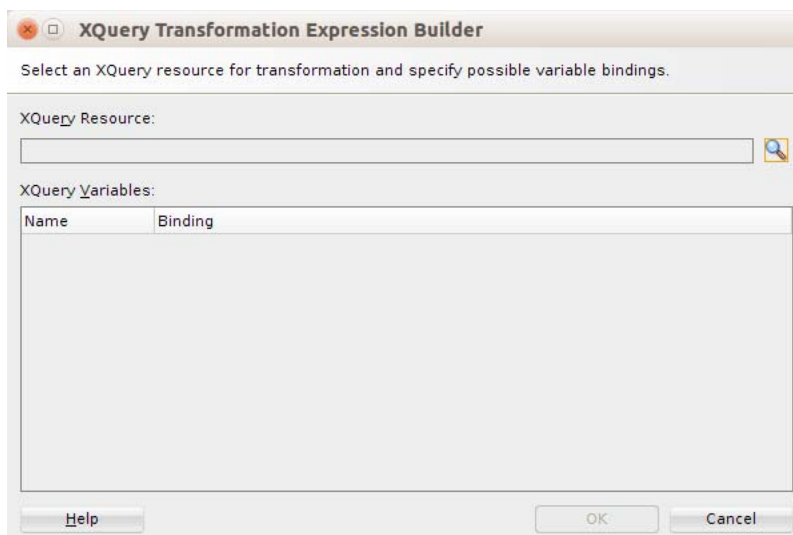
20. Click the XPath link under Location field of Replace Action. Here we need to provide xpath of the variable which we need to transform.



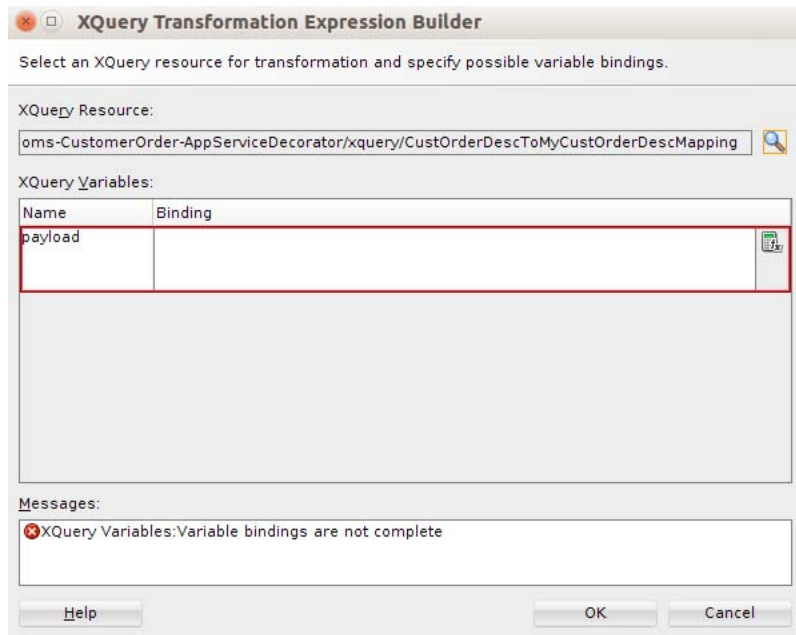
21. Select CustOrderDesc and click **Insert into Expression** to add expression to to Expression filed.



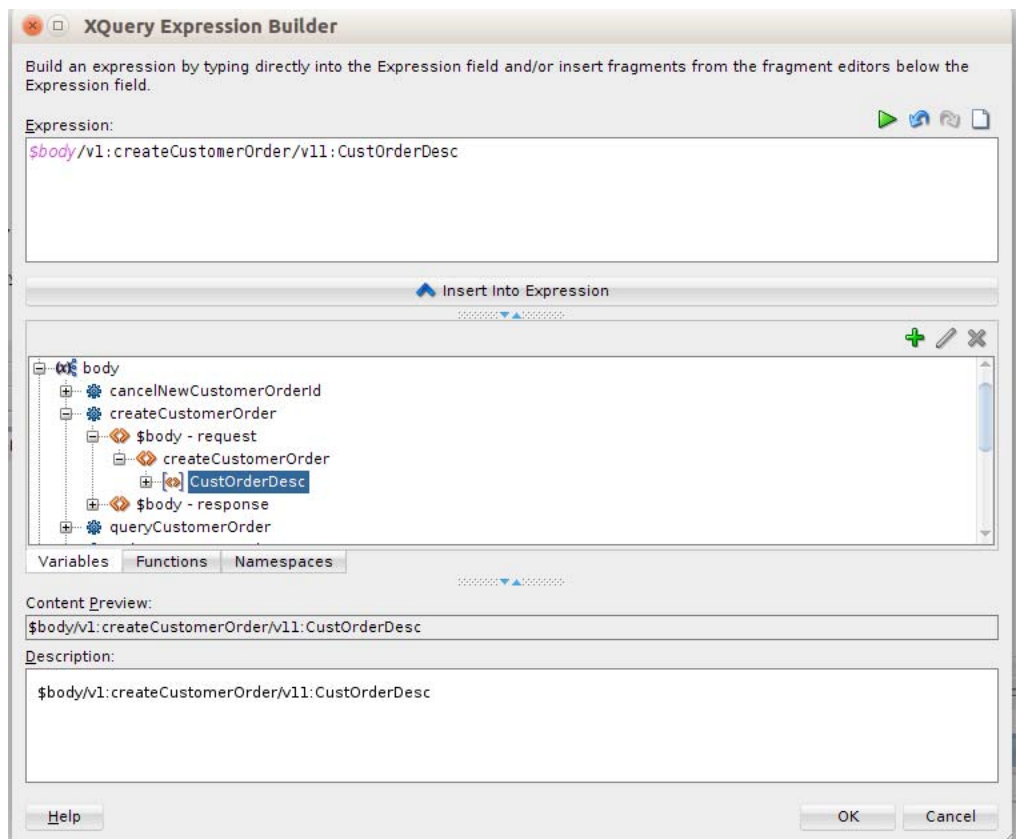
22. Click **OK** to return to the main window. Then click browse under the Xquery Resources field. Here we need to provide xquery which will return the transformed payload.



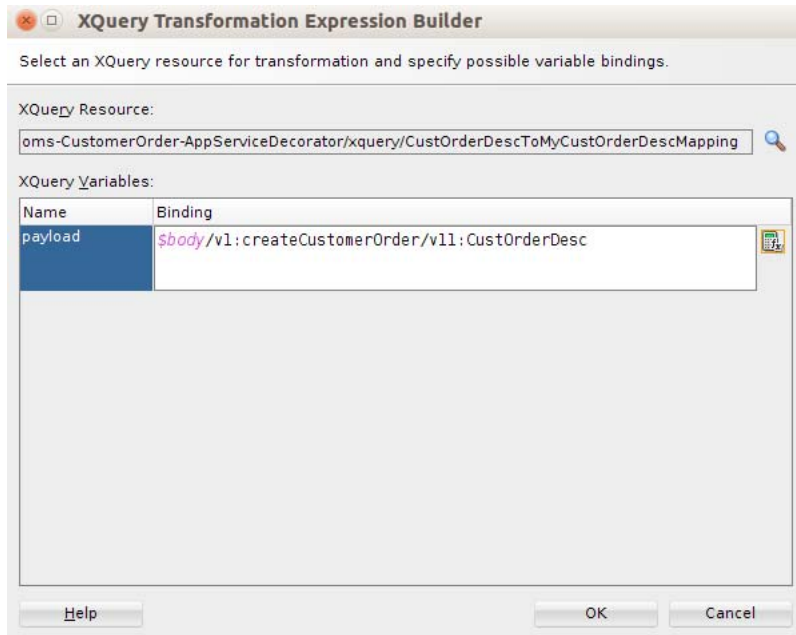
23. Click browse and select the xquery file.
24. In the Binding field, we need to provide path of the input payload which needs to be transformed. Click Xquery Expression Builder link and select CustOrderDesc.



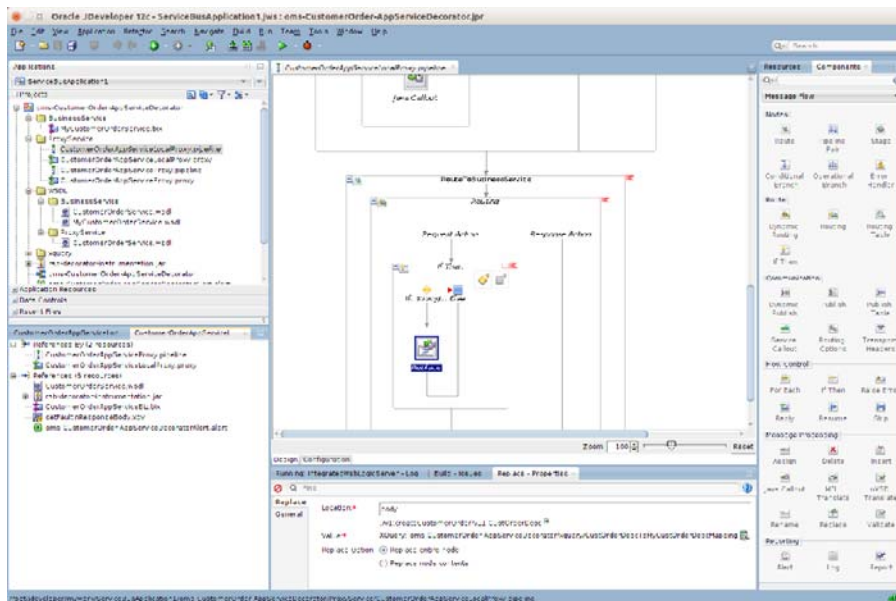
25. On the XQuery Expression Builder window, click **OK**.



26. On the XQuery Transformation Expression Builder window, click **OK**.

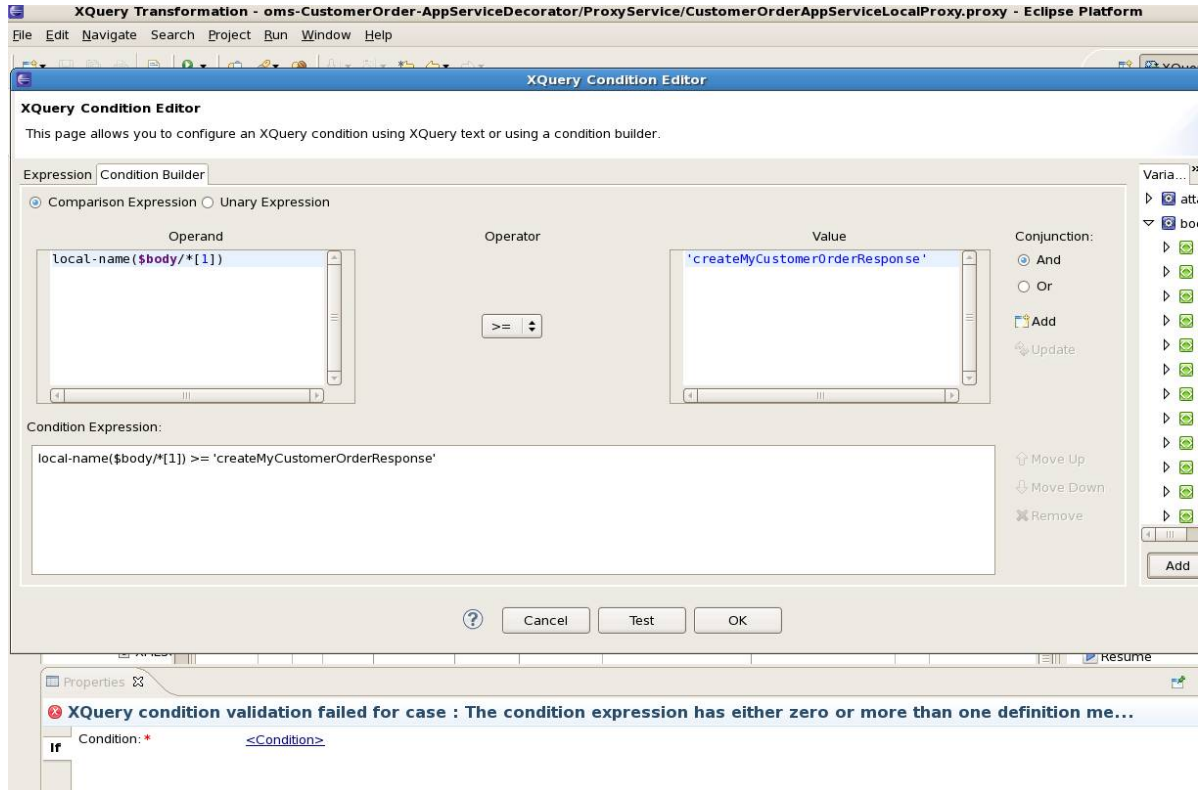


You can add more Replace actions if conditions to transform payload for each operation type are added. You will need to write xquery files for each input payload to output payload transformation.

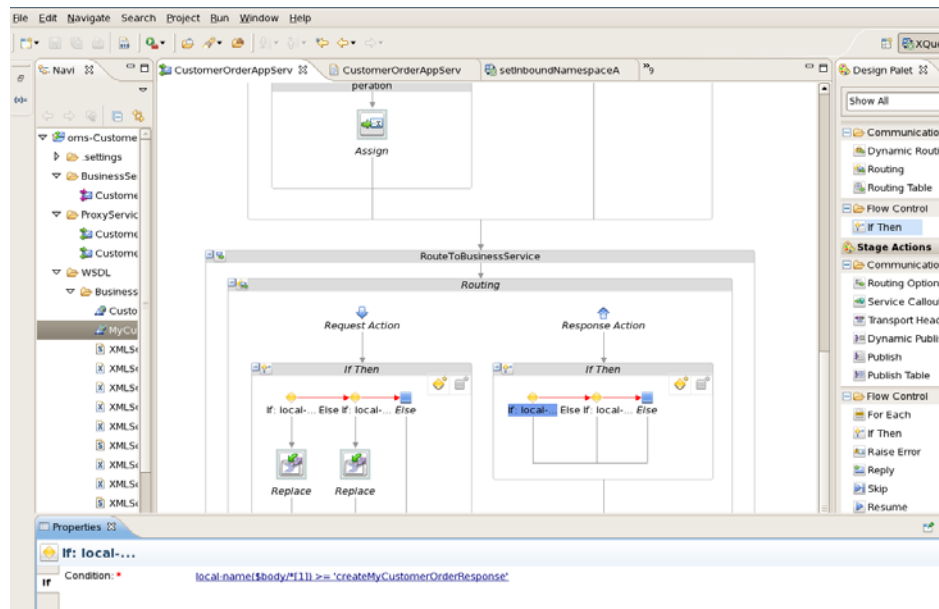


27. Similar transformations need to be performed, in reverse order, in the Response pipeline as well. This will transform MyCustOrderDesc to CustOrderDesc. To do this, write an xquery file which transforms MyCustOrderDesc to CustOrderDesc.

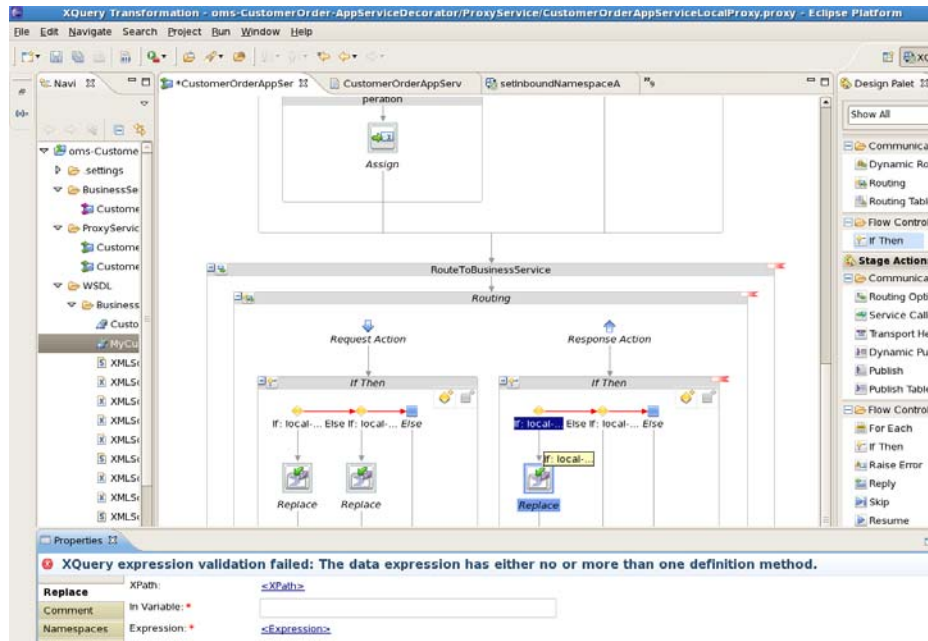
Note: The steps for creating xquery file will be exactly similar to the steps for creating CustOrderDescToMyCustOrderDescMapping file. The only difference will be that the source and target types will be reversed in this case. Here, the source type will be MyCustOrderDesc.



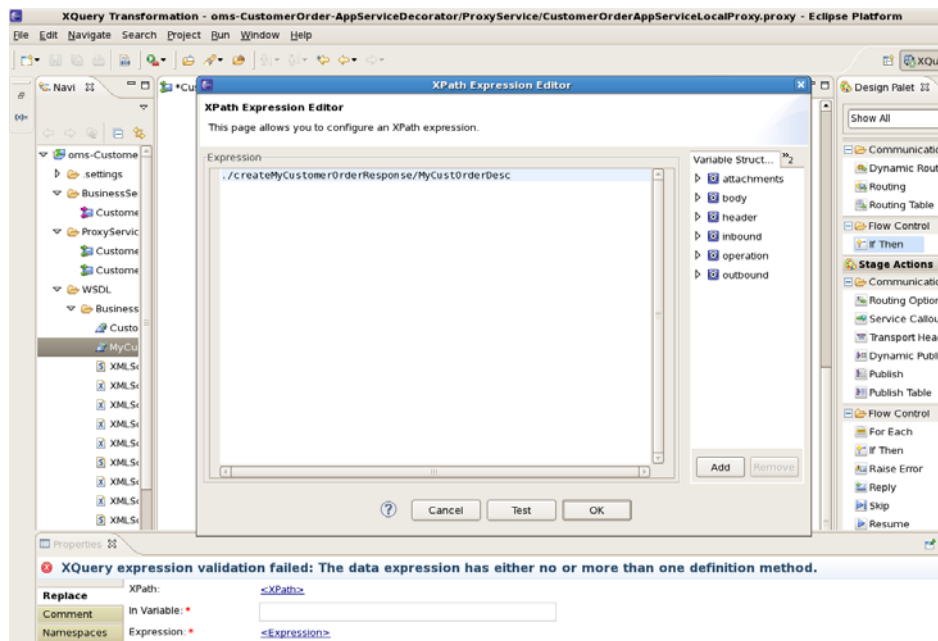
28. Click OK to go back to properties window.



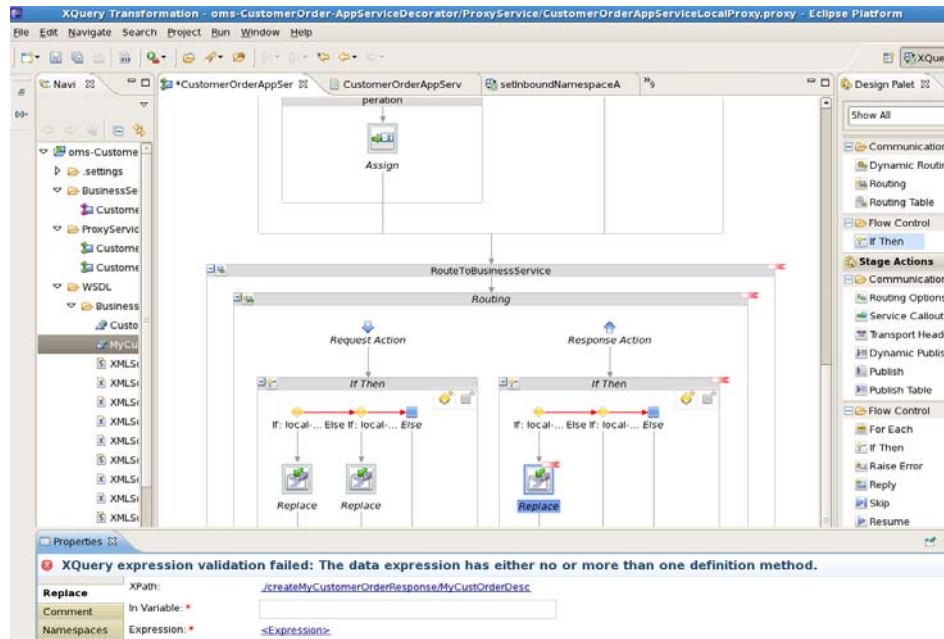
29. Add a Replace action in first if condition flow:



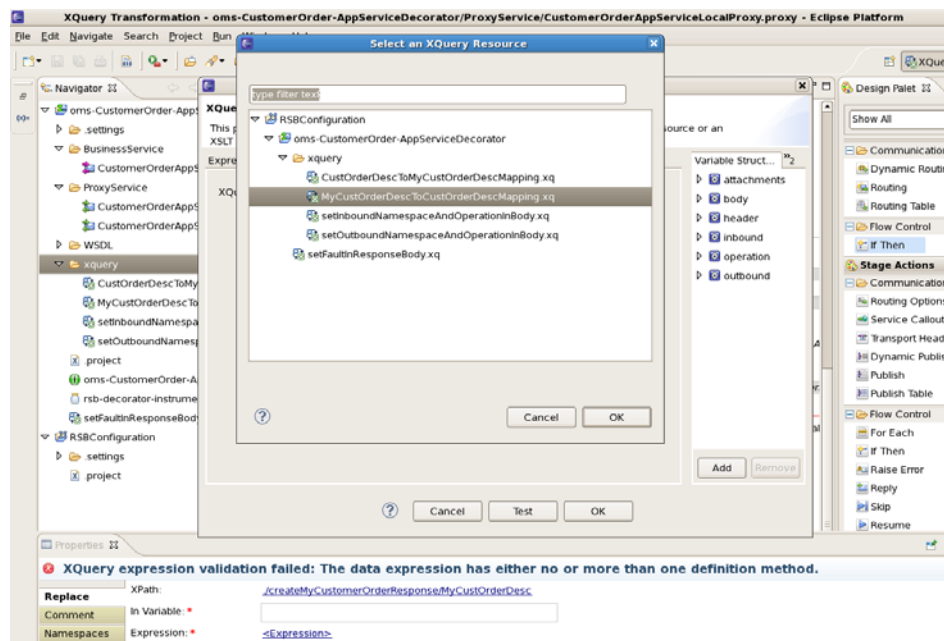
30. Click the `<xpath>` link. Here we need to enter the xpath of the element which needs to be transformed. Enter as shown below:



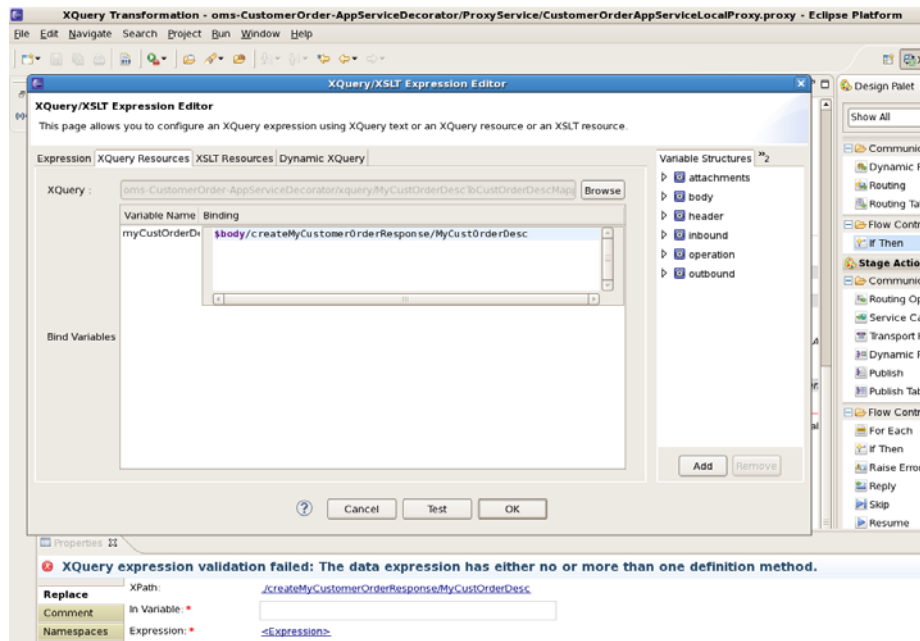
31. Click OK.



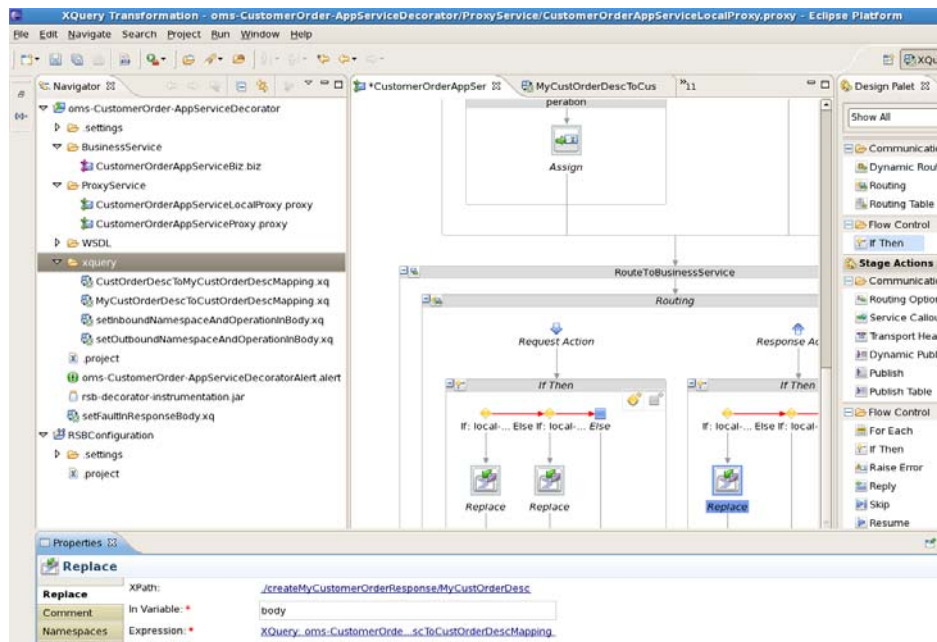
32. Click the <Expression> and go to XQuery Explorer window to select the xquery file. Select MyCustOrderDescToCustOrderDescMapping xquery file.



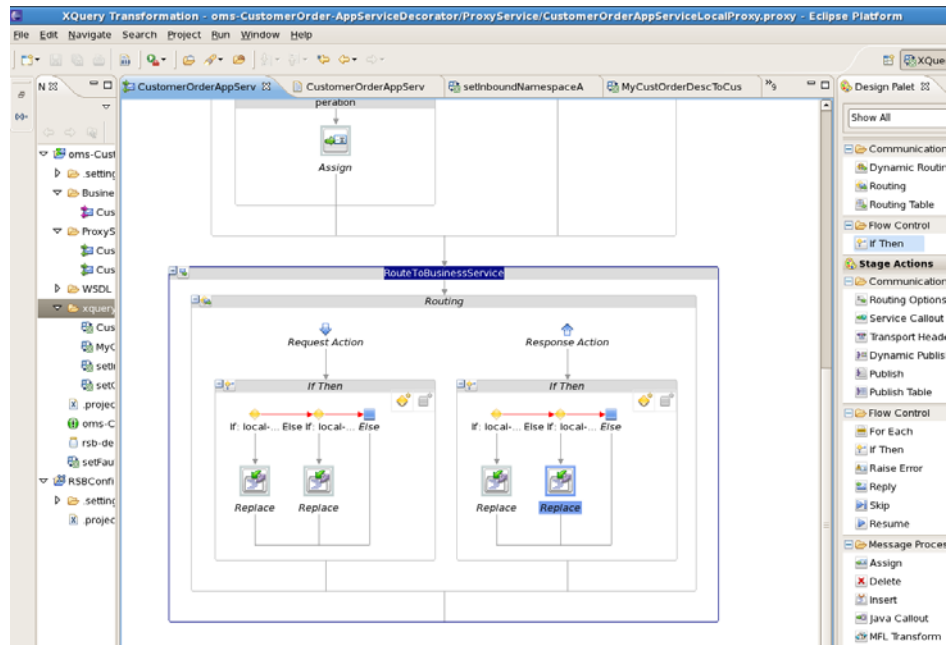
33. Click OK. In the binding field enter path to MyCustOrderDesc as that is the source payload for transformation:



34. Click OK. In Variable field enter body.



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



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

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 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 JDeveloper or from OSB console. It is recommended to add pipeline alerts in the OEPE and then export the decorator jar.

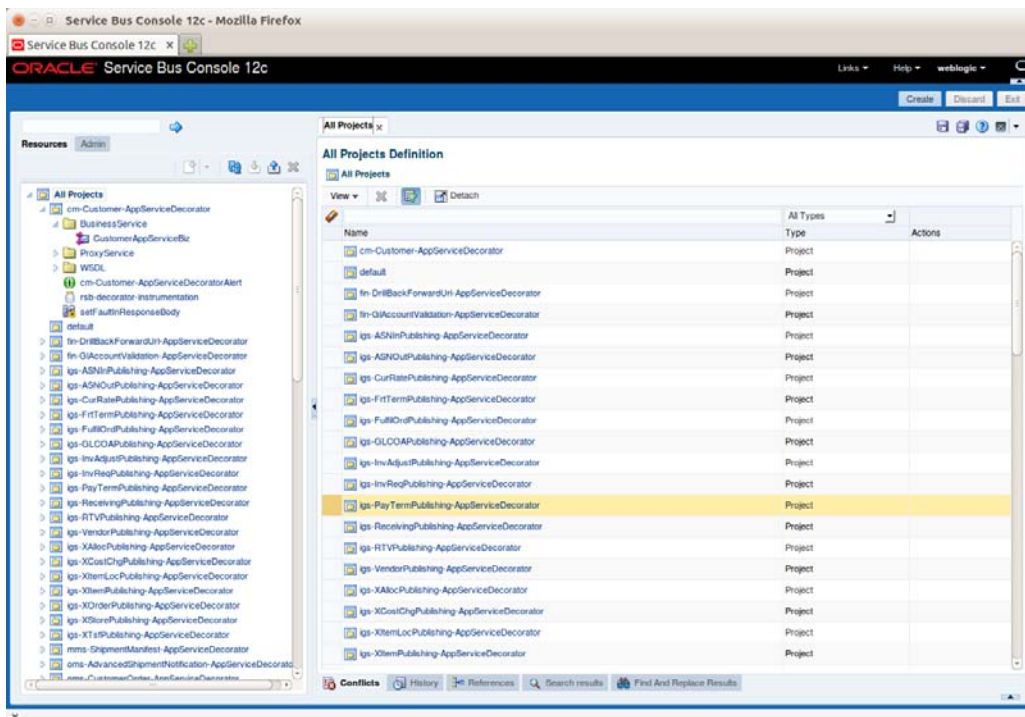
Default Alerts in RSB Decorator Projects

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

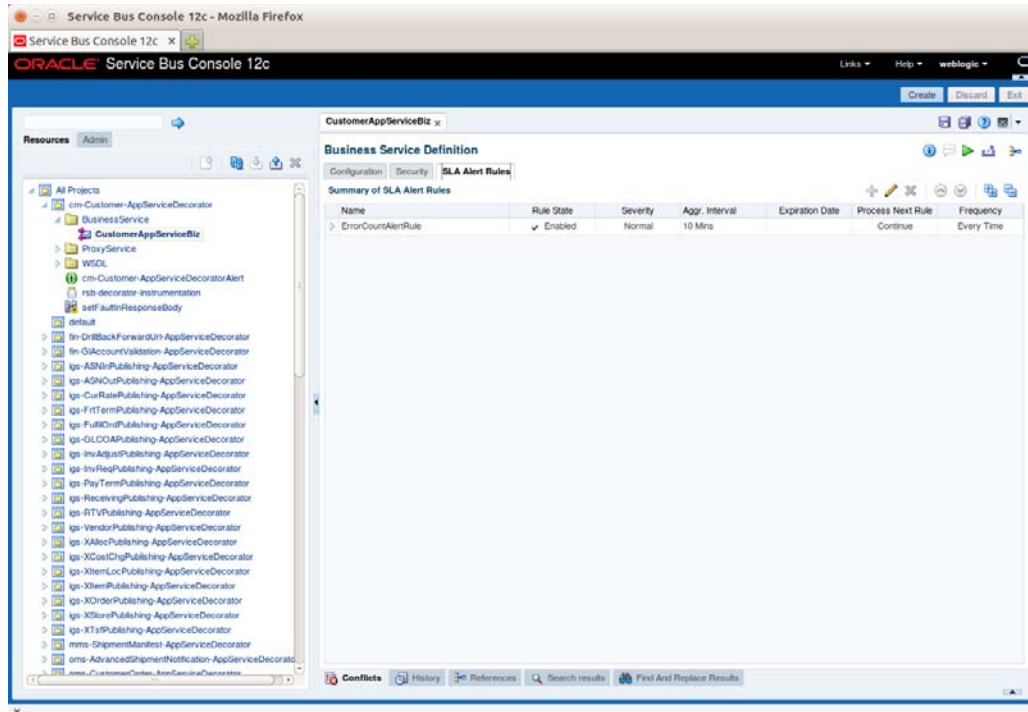
SLA alerts are operational settings and they can be added or modified only from OSB console. Follow the steps to delete the default alert rule and add a new rule:

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

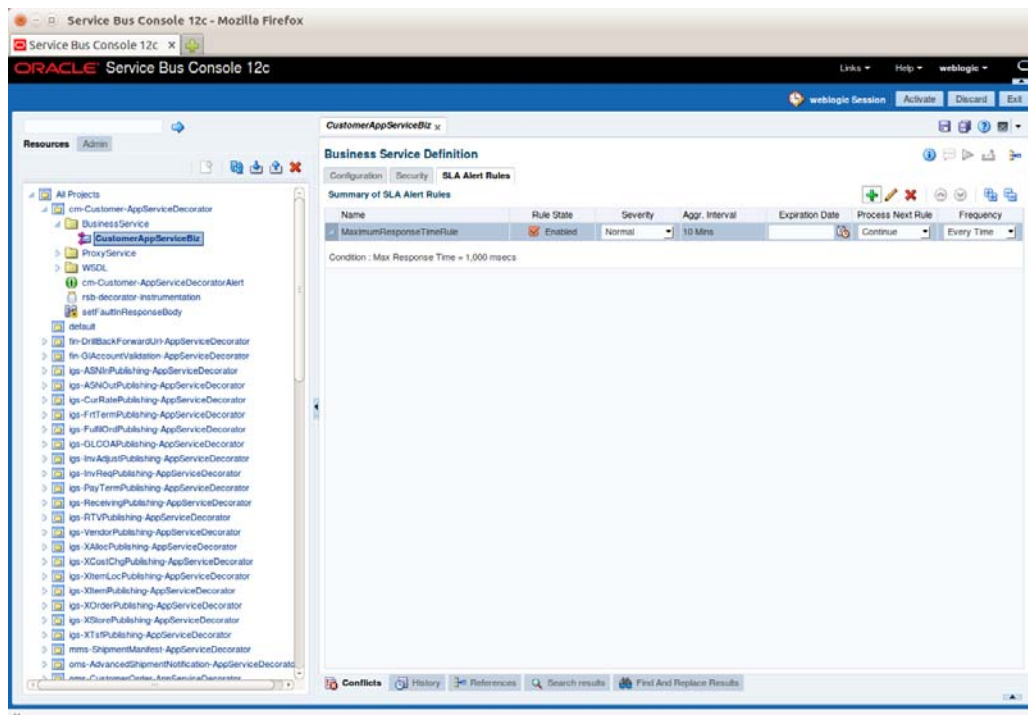


2. Click the project for which you want to modify SLA alert.

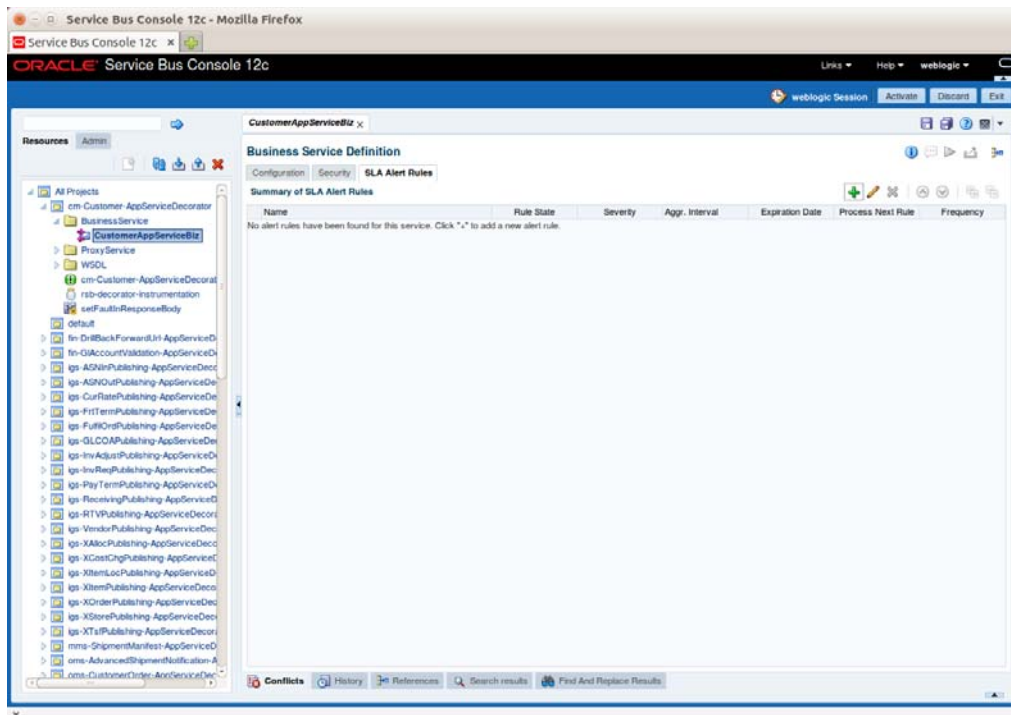
The steps to add or modify an SLA alert are same for both business service and proxy service. In this example, we will show steps for a business service. Browse to the business service and go to SLA Alert Rules tab of that service.



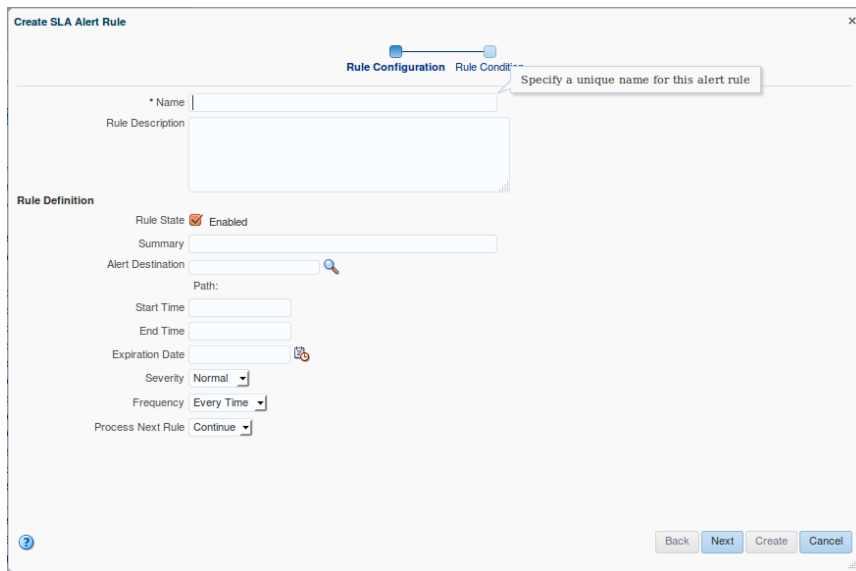
3. Click **Create** in the Change Center to create a new section.



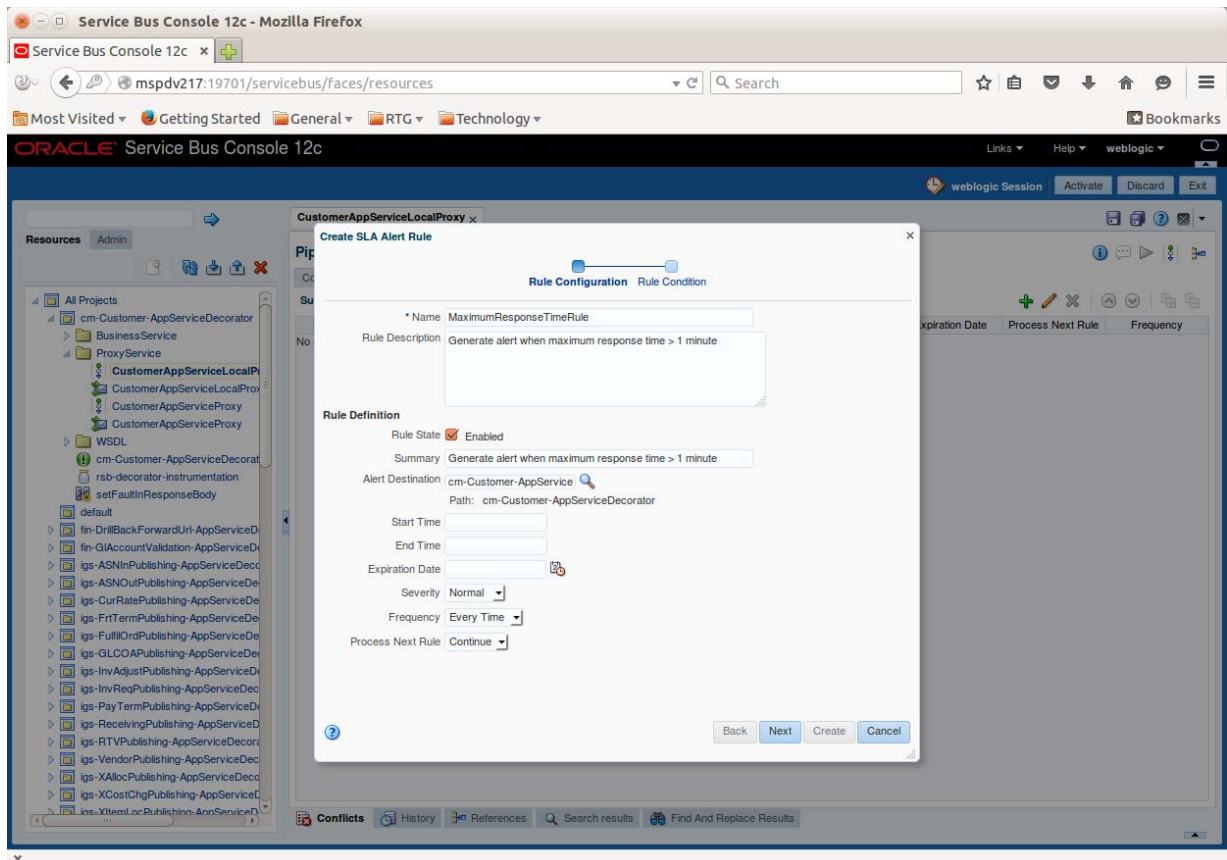
4. Select the rule and click **Delete**. The rule is deleted.



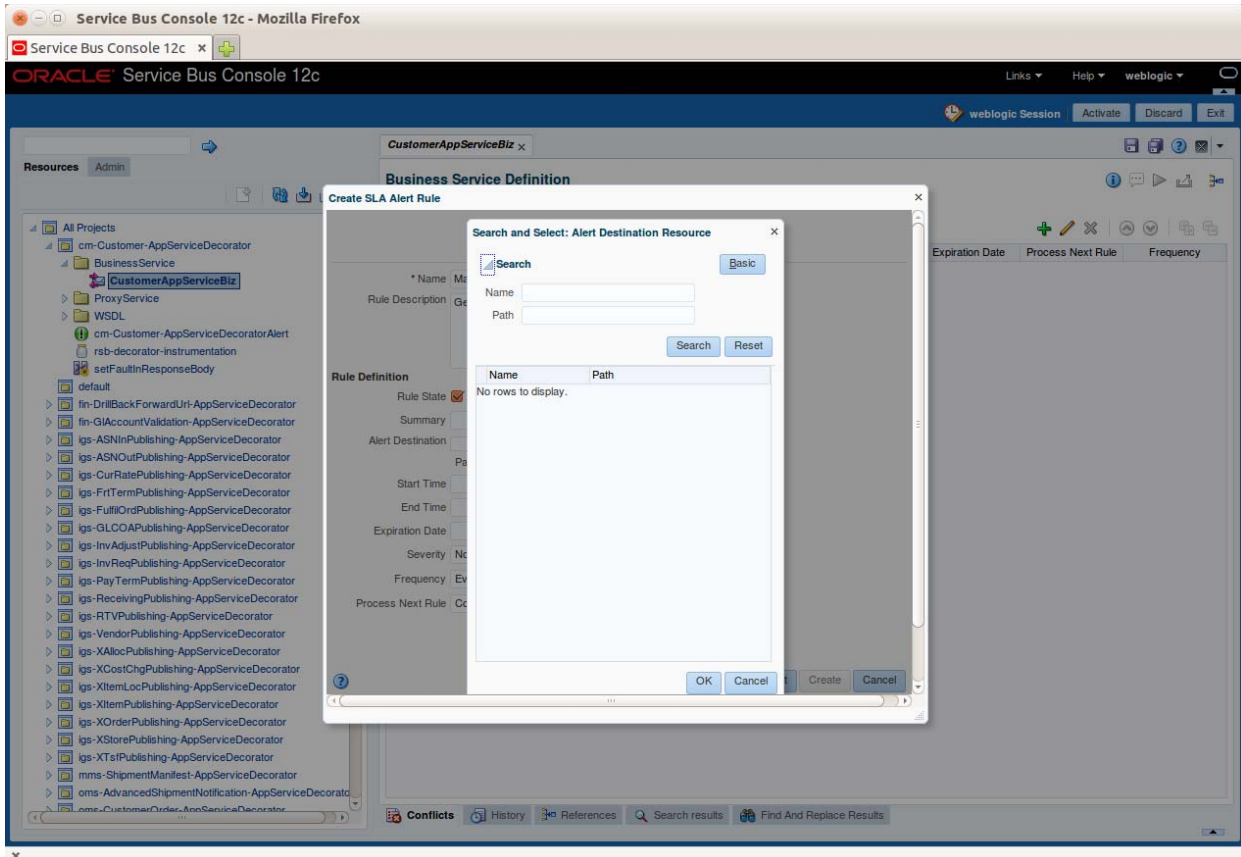
5. Click add to add a new rule.
6. Enter appropriate values for Rule Name and Alert Summary fields. It is recommended to have a good summary of why this alert rule should be generated. Having proper description of all fields will be useful when looking at rules in RIC console and it will help better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



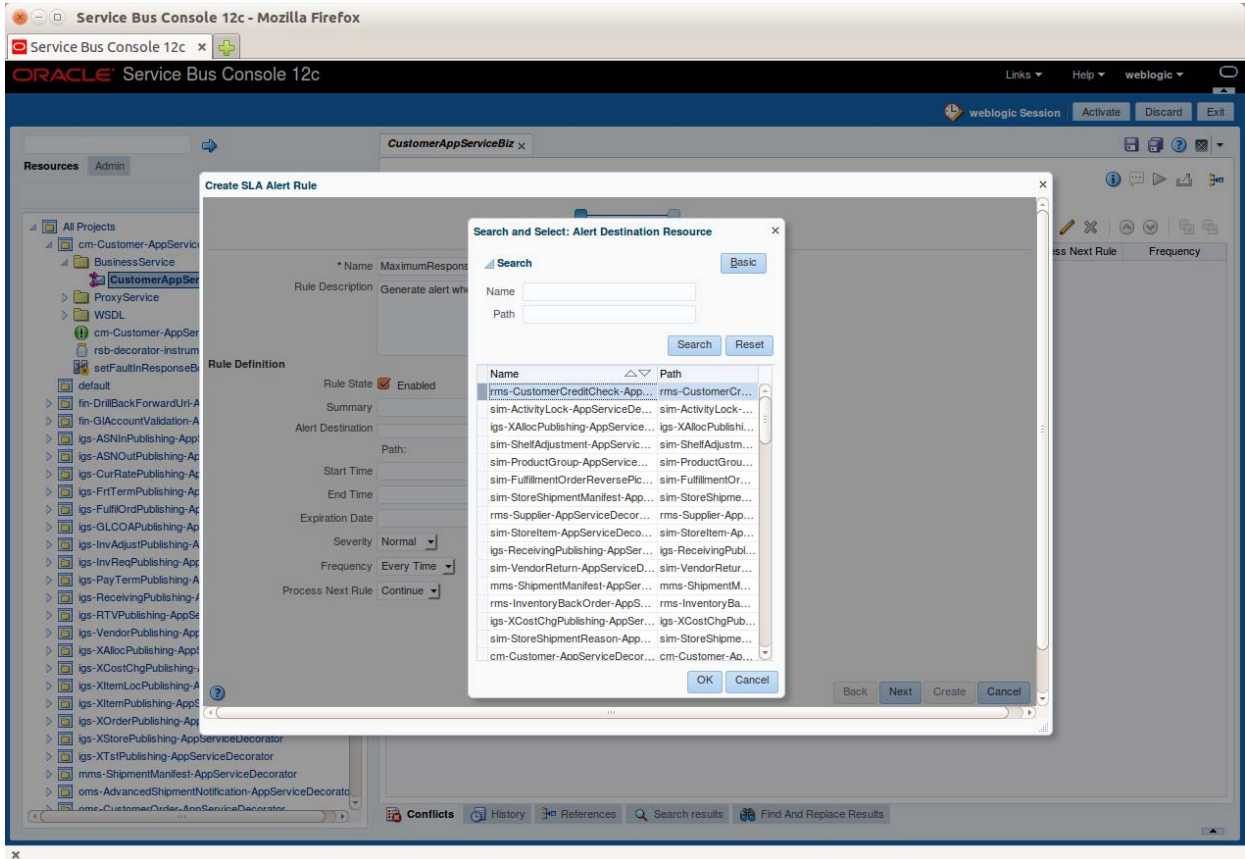
7. For Alert Destination, click browse.



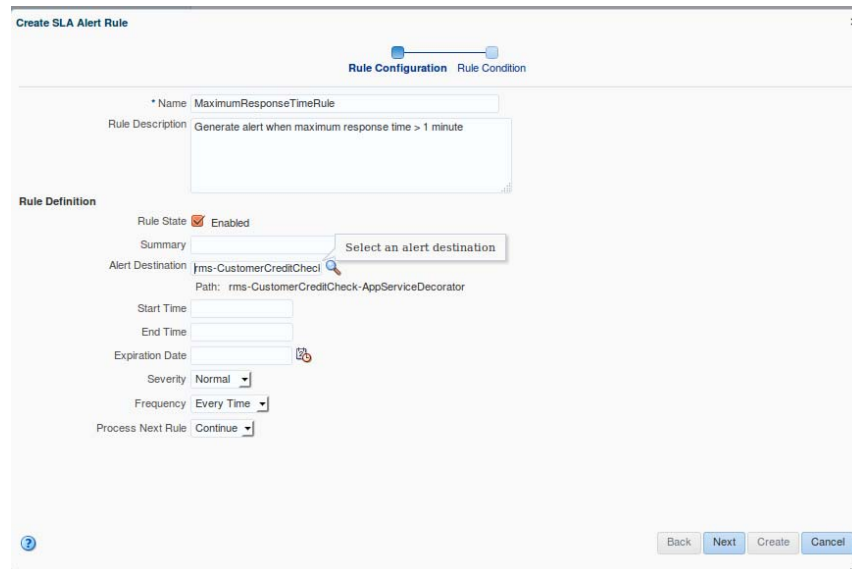
8. Click **Search** to display all alert destinations.



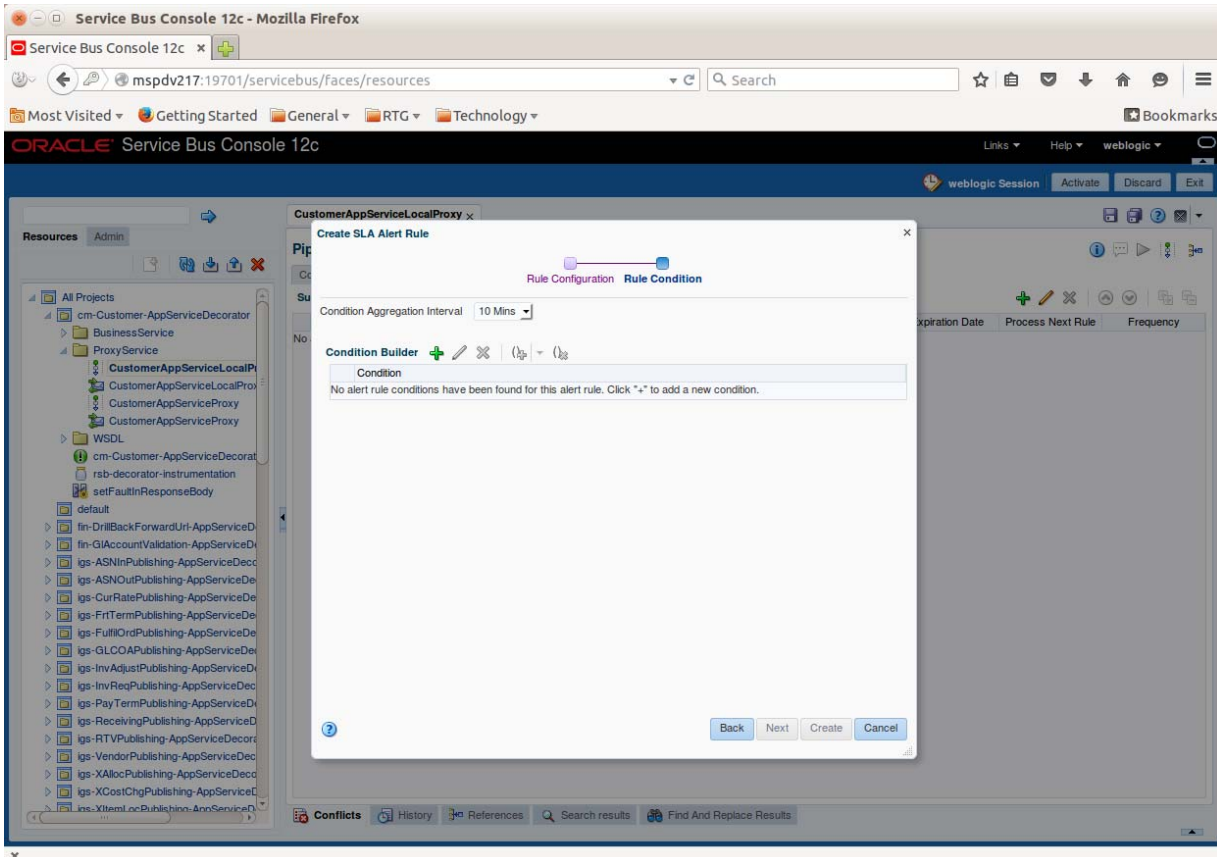
9. Select Alert Destination and click OK.



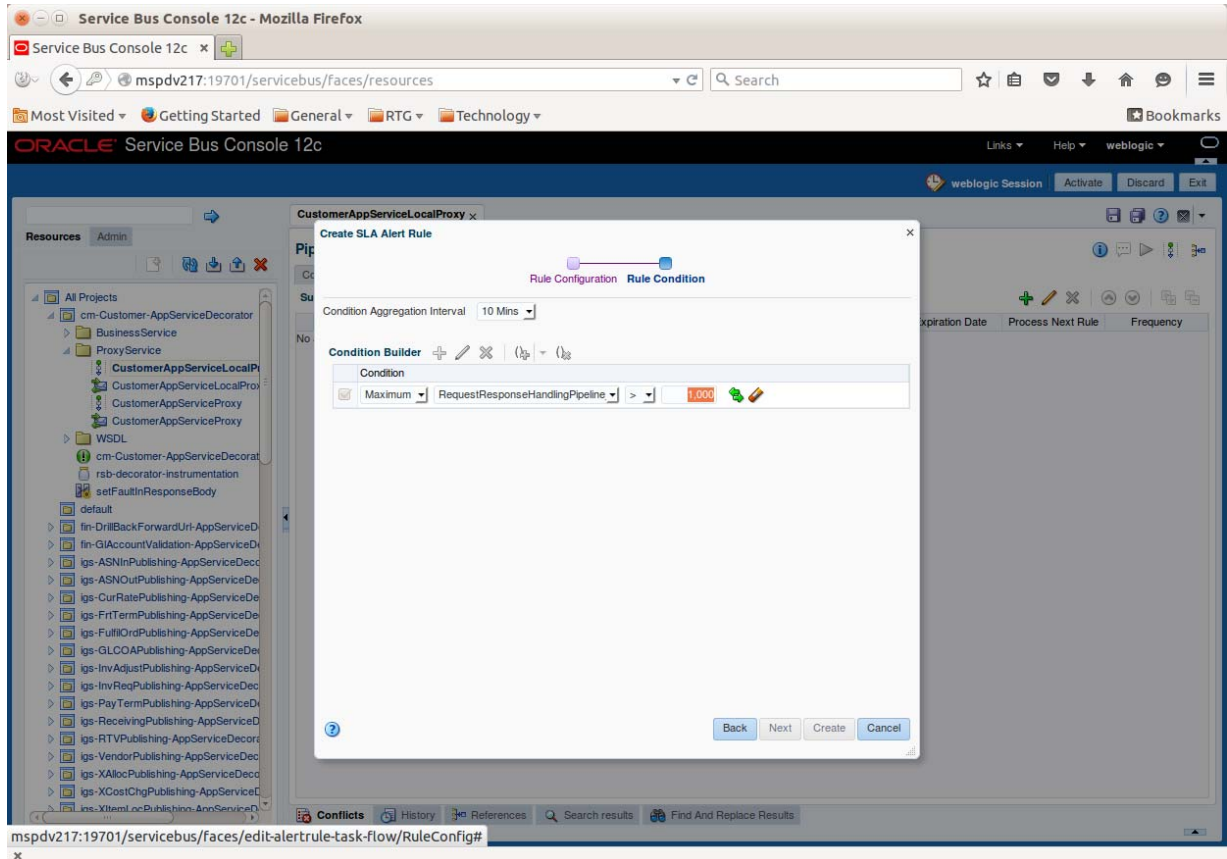
10. Click **Next**. You can build the expression which defines the criteria for alert rule on the next page.



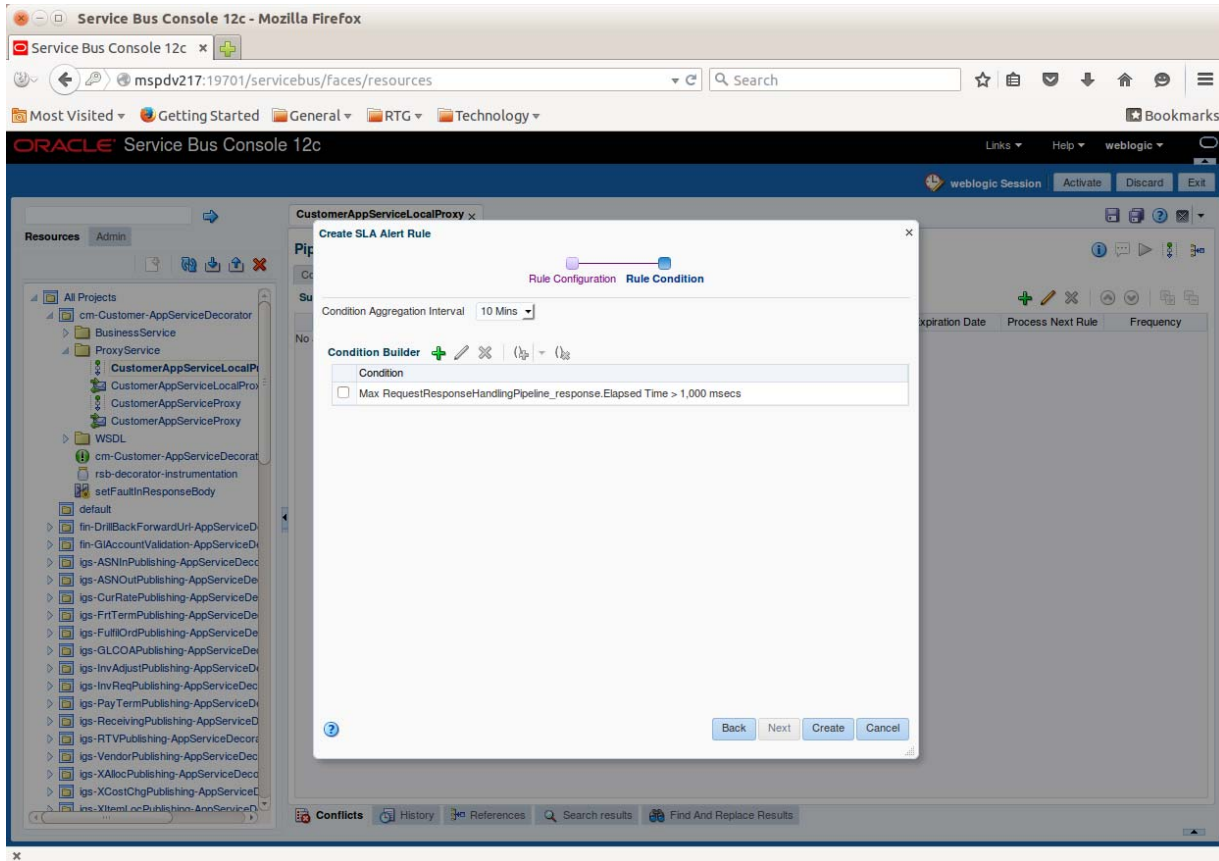
11. Now in the Condition Builder area, you can build the expression for which you want to generate SLA alert. In this example, we will build an expression to generate an SLA alert whenever the response time of business service is more than one second. Click **add**.

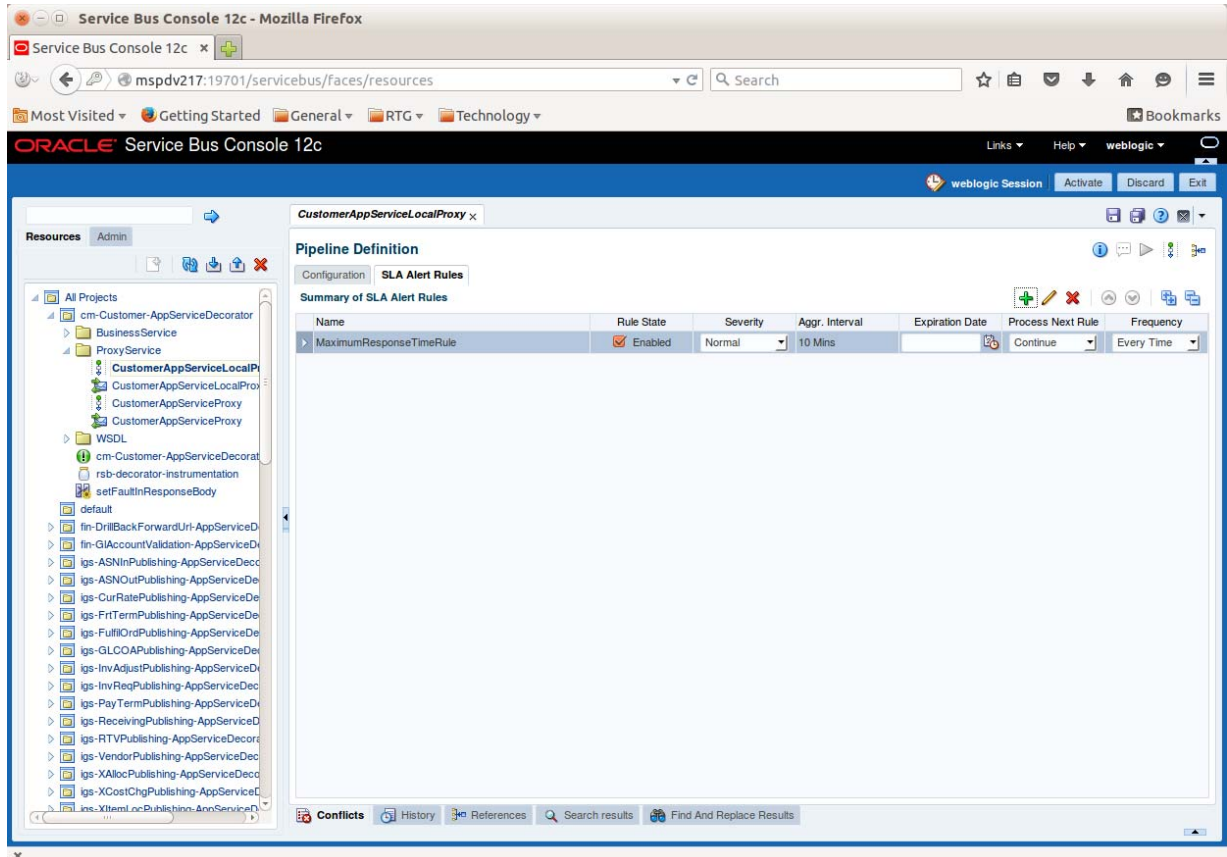


12. Select the appropriate values in the condition and click update.



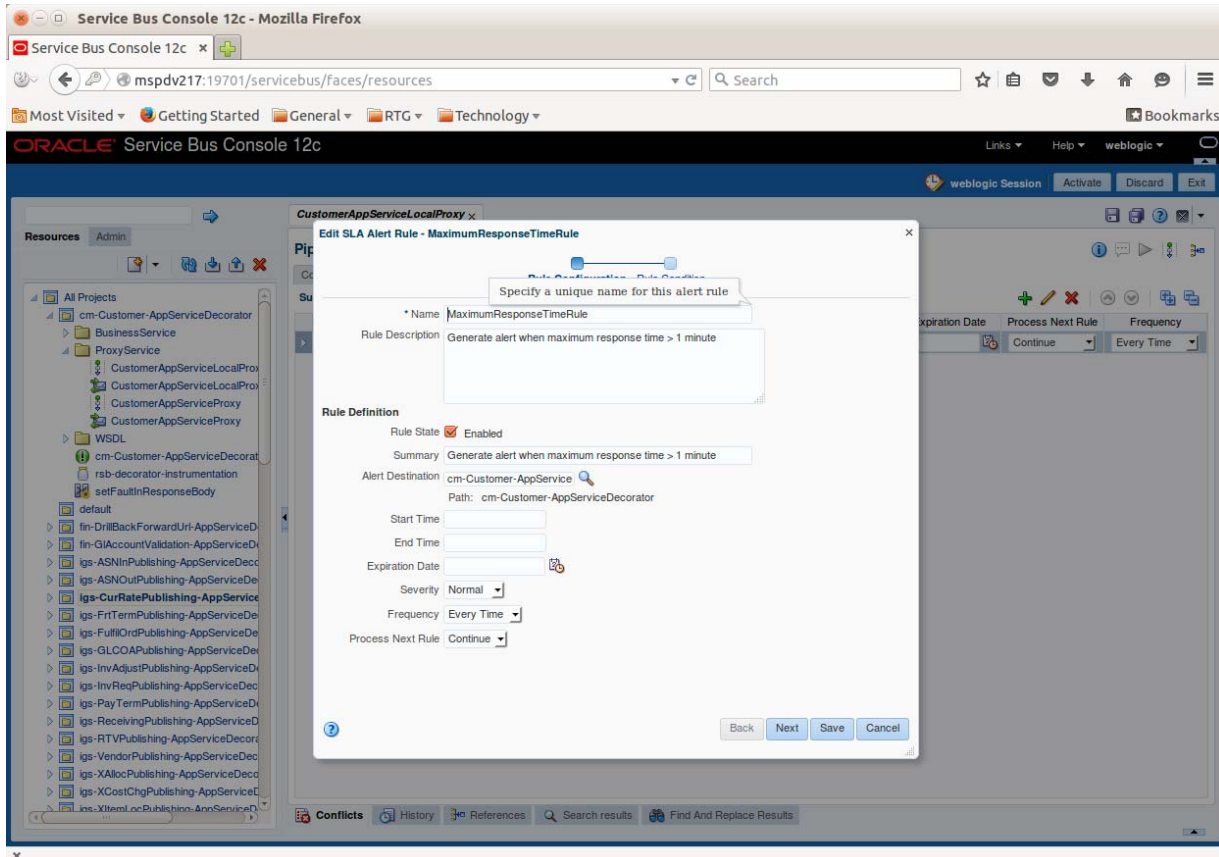
13. You can build more complex expression using join and split condition buttons to build the rule. After the expression is build, select the condition and click **Create**.





In the above page, you will see all the details about the new rule. We recommend that the condition expression is copied into the description field as well, so that when the SLA alert is displayed in 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. To edit the rule and add the description, click edit.

14. Paste the condition expression into the Rule Description field and click **Save**.



15. 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.

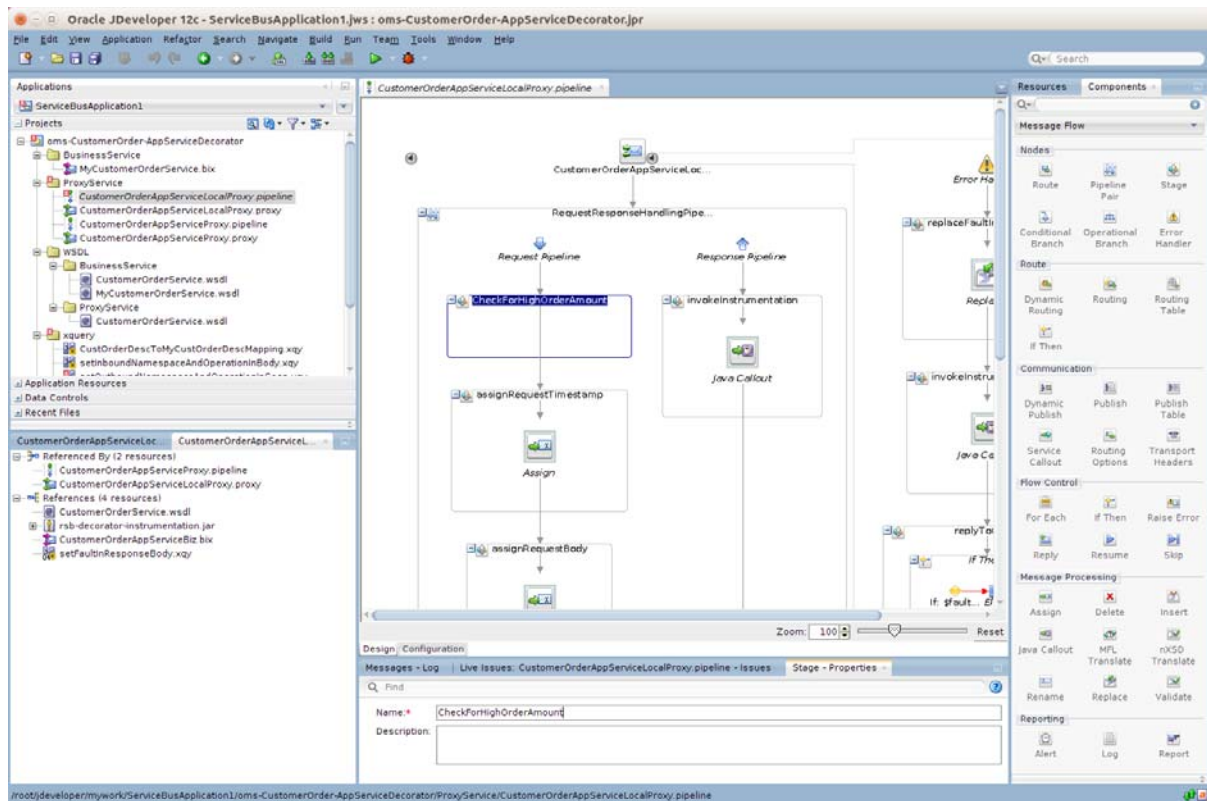
How to Add New Pipeline/Business Alert

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.

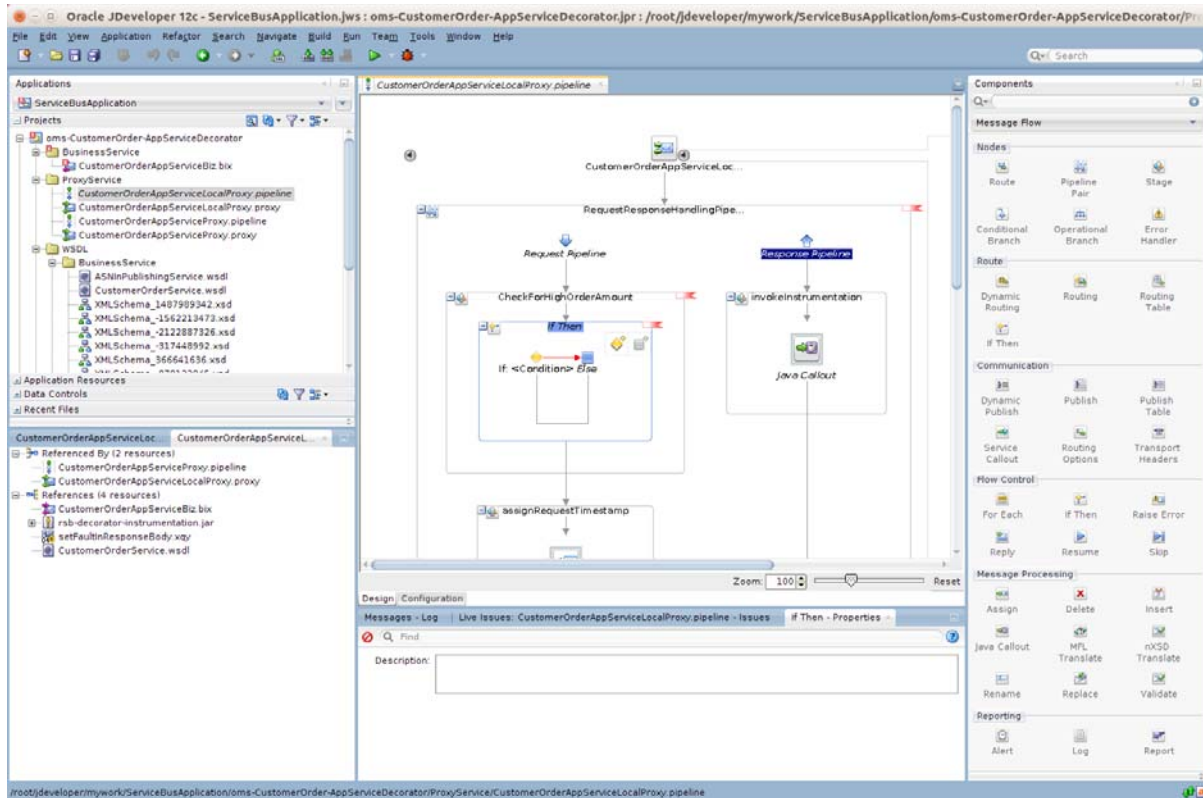
Pipeline alerts can be added in proxy services only. In RSB decorators, the message flow is defined in local proxy services. Therefore any new pipeline alerts should be added in local proxy service.

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:

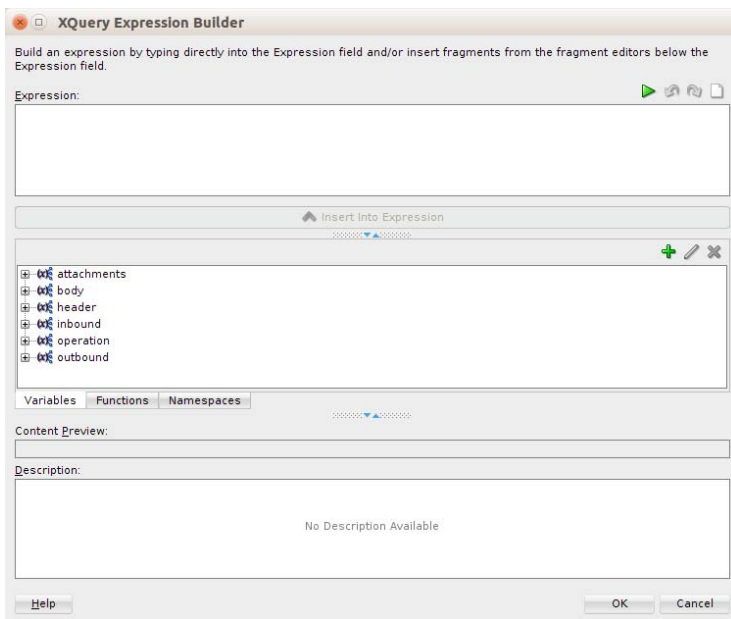
1. Drag the Stage component from Components windows.



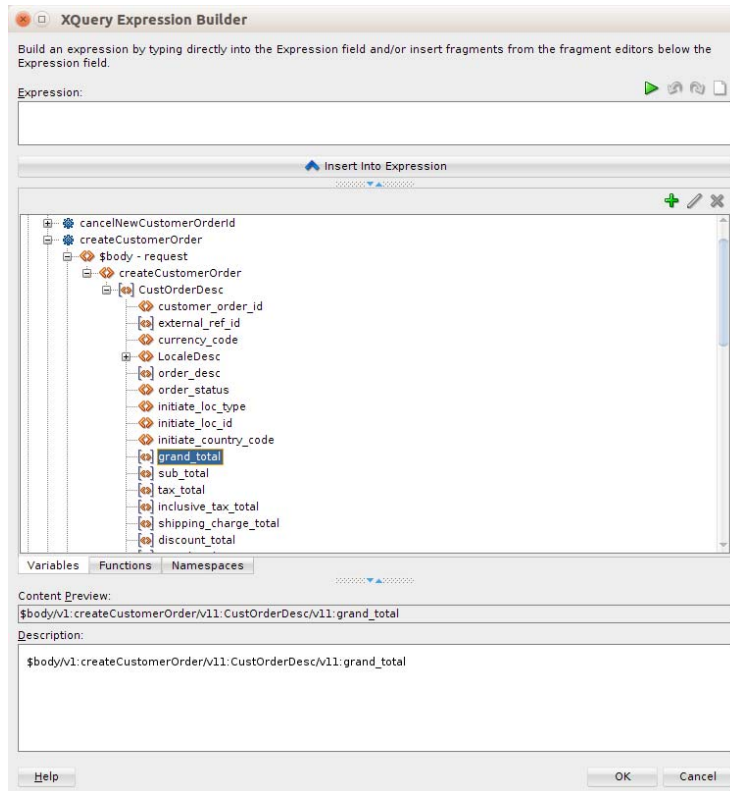
2. Enter appropriate name for the stage.
3. In the stage, add a new If Then flow, drag the If Then component from the Components window to the newly added stage.
4. For the first If condition, select the If condition and click the <Condition> link in If properties window to access Xquery access expression builder.



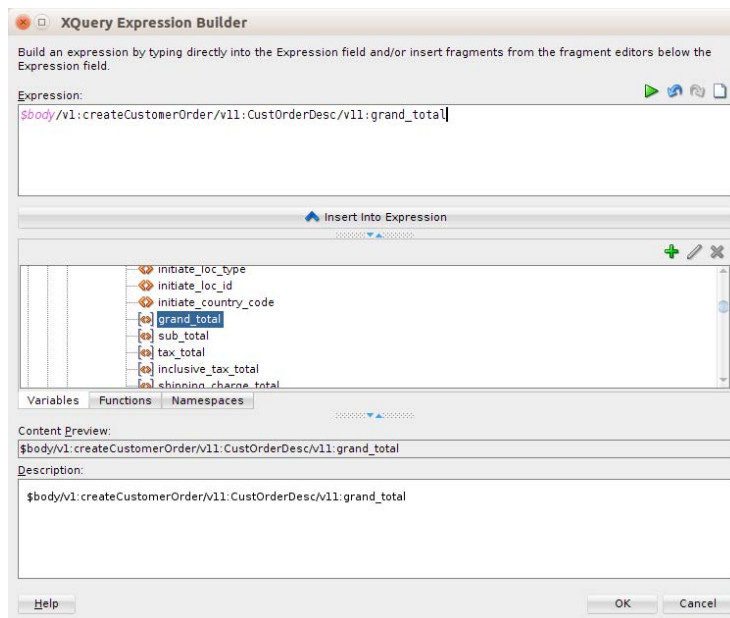
- In this screen, you can build the expression for the alert rule. Select the request schema element body > createCustomerOrder > CustOrderDesc > grand_total for the operation:



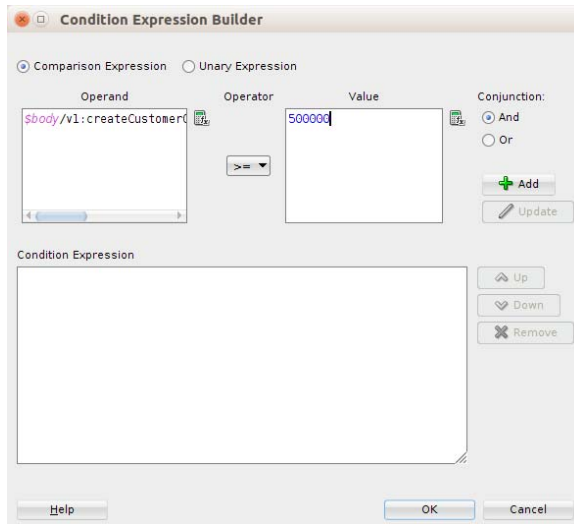
- Click **Insert into Expression** to add to Expression field.



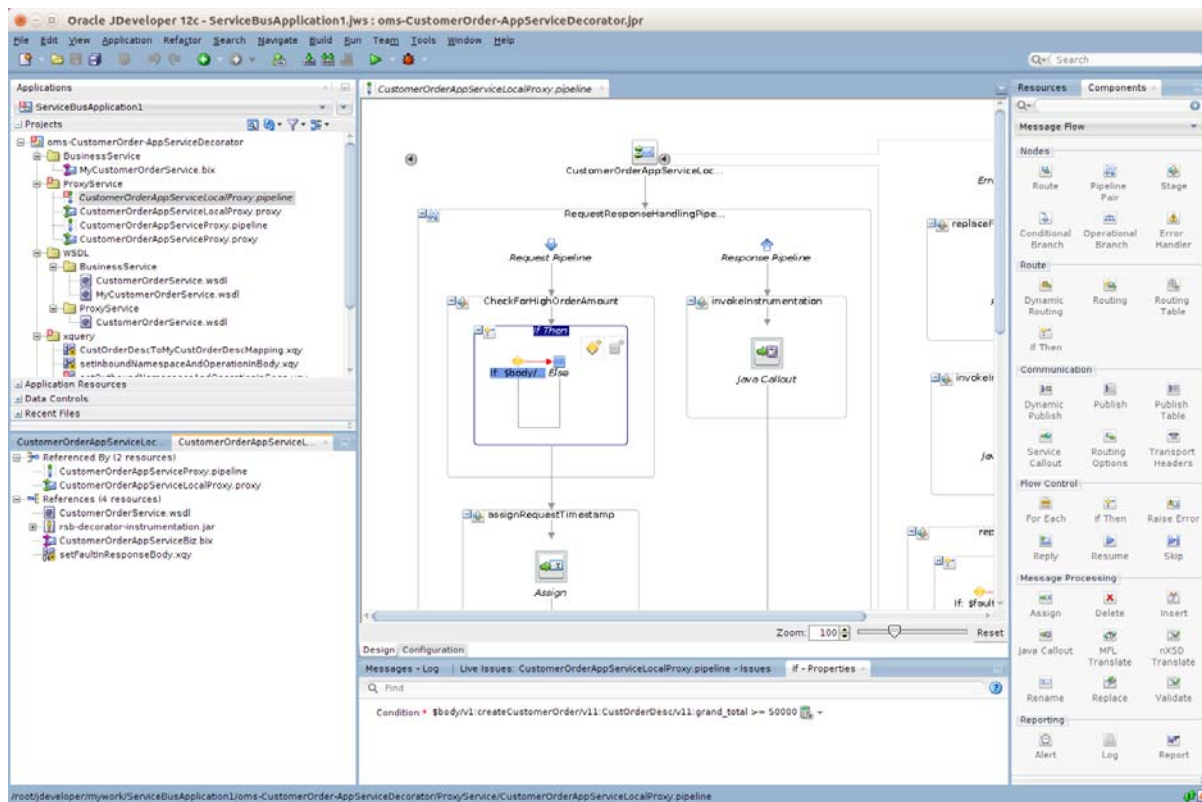
7. Click OK. Select Condition link from If variable window.



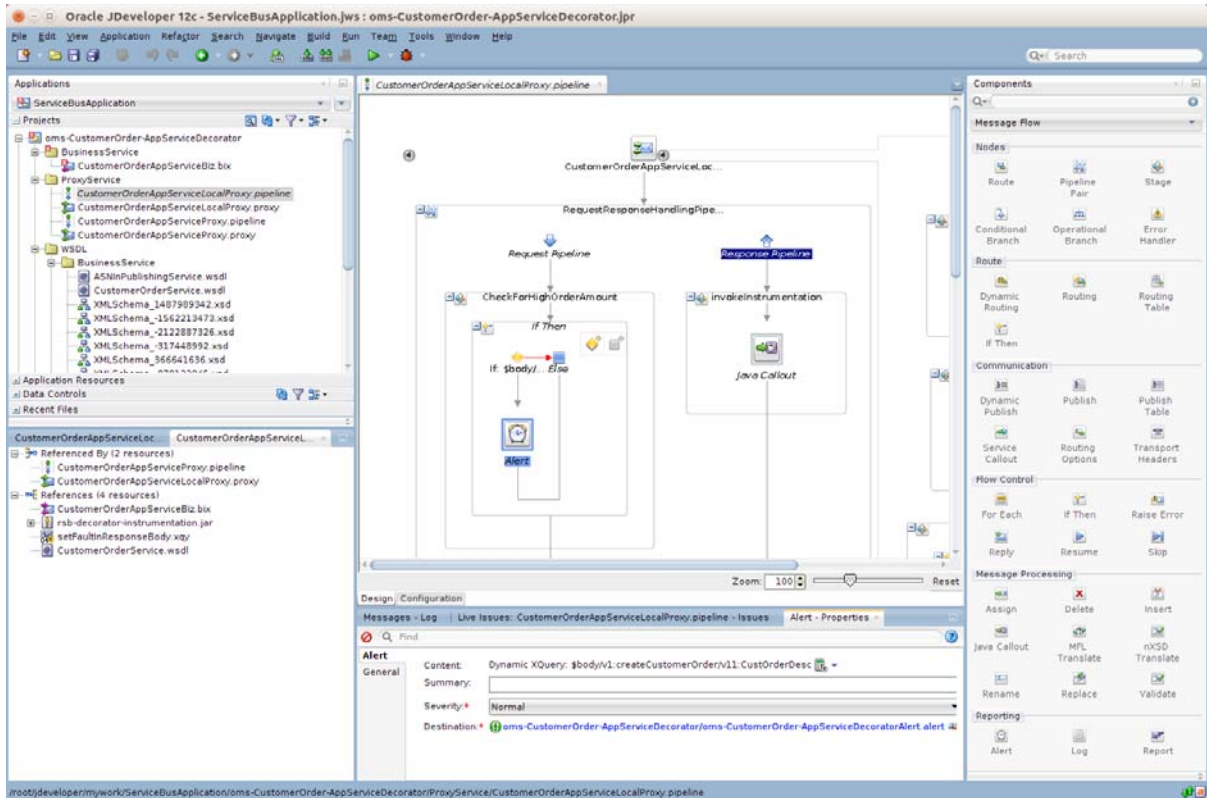
8. Click Launch Xquery expression builder link next to Operand field. Select body > createCustomerOrder > CustOrderDesc > grand_total and click **Insert into Expression**. Enter appropriate value in the Value field.



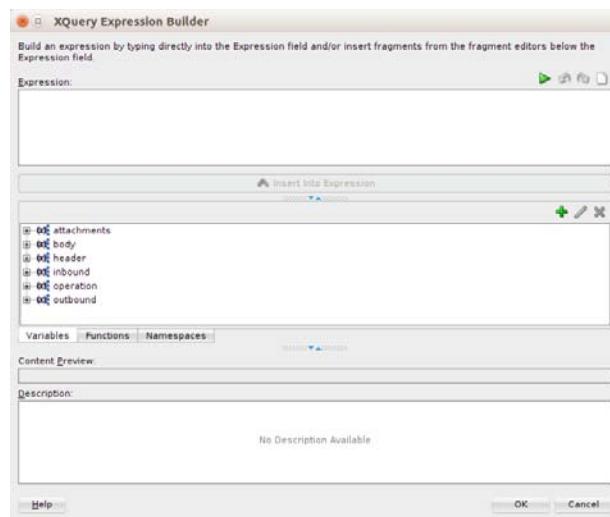
9. Click **OK** and then click **Add** to add the condition.
10. You can build more complex expression using the **And** and **Or** options. After building the condition expression, click **OK**.



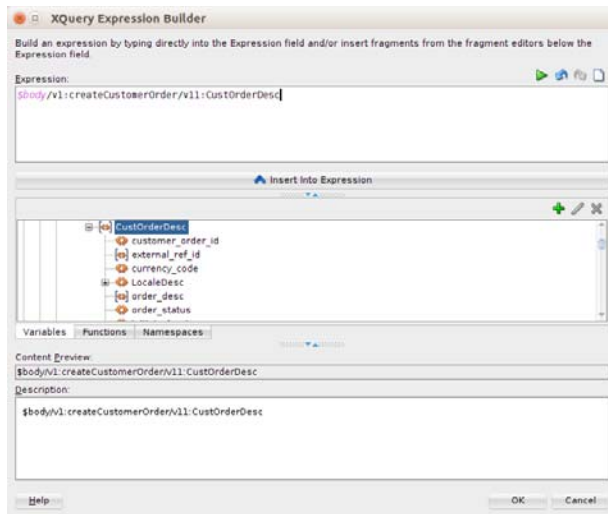
11. Now we need to add an Alert action for this If condition.
12. The alert action gets added, right click the If condition added in the above steps, select **Insert Into > Alert**.



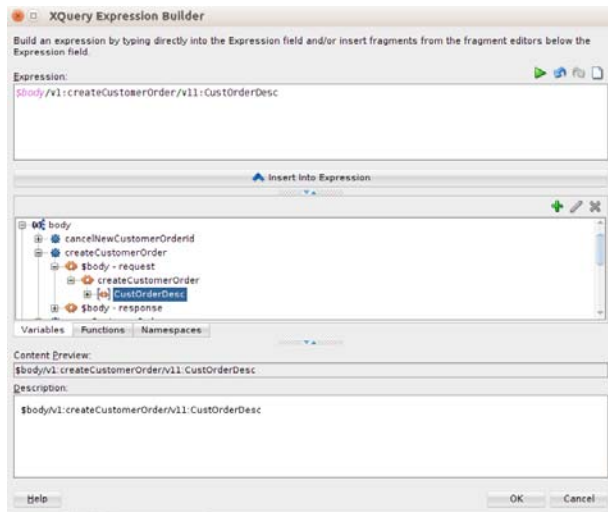
- In the **Expression** field, you can enter the xml that you want to see in alert description when alert is generated. For example, you may want to see the whole SOAP body which caused the alert to be generated or a subset of the SOAP body. Click the **<Expression>** link in the Alert-Properties window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



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

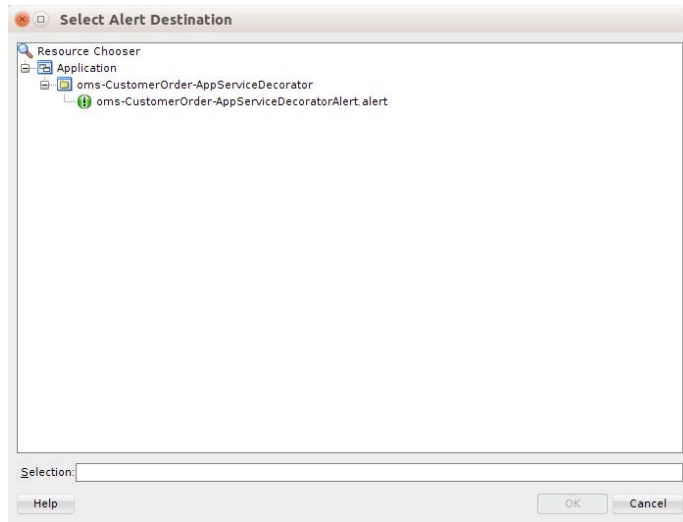


15. Drag CustOrderDesc to the Expression window.

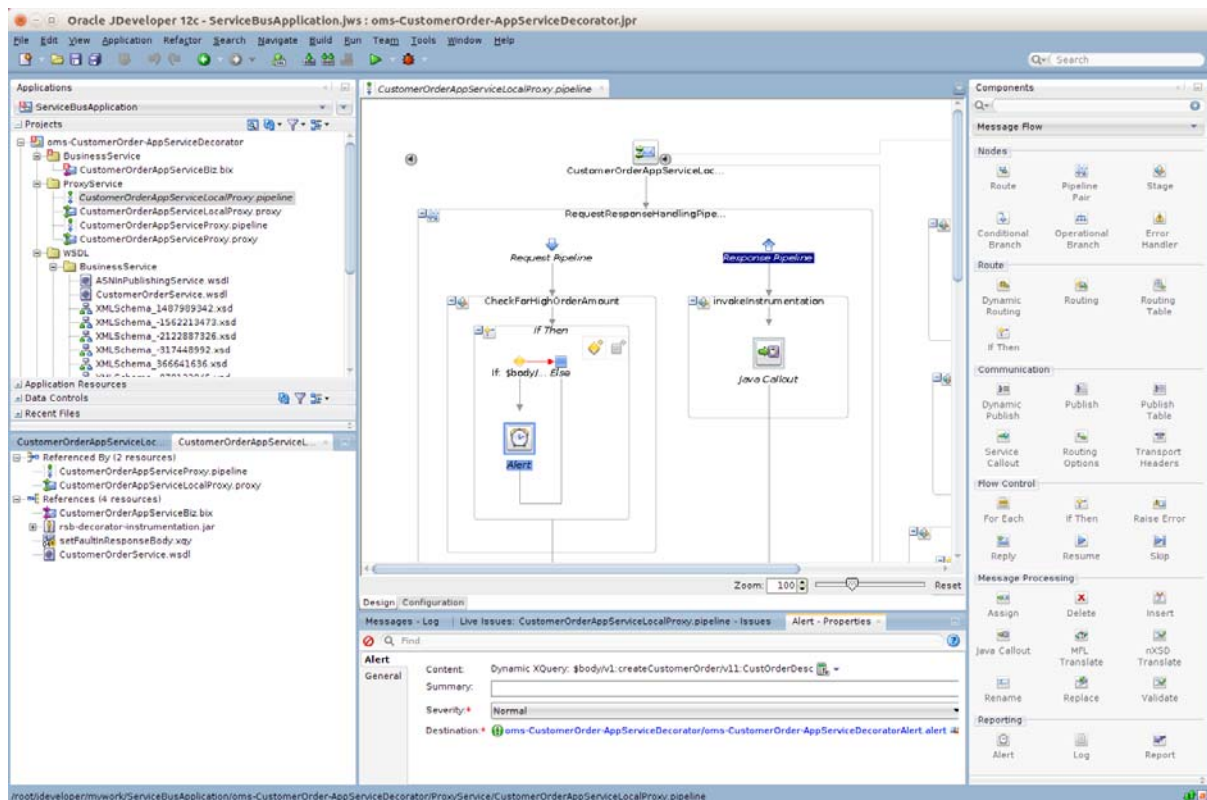


16. Click OK. In the Summary field, enter an appropriate name for the alert.

17. Select Destination link from the Alert - Properties window for the Alert. You should select the destination that was created by default in this project.



18. Click OK to close the dialog box.



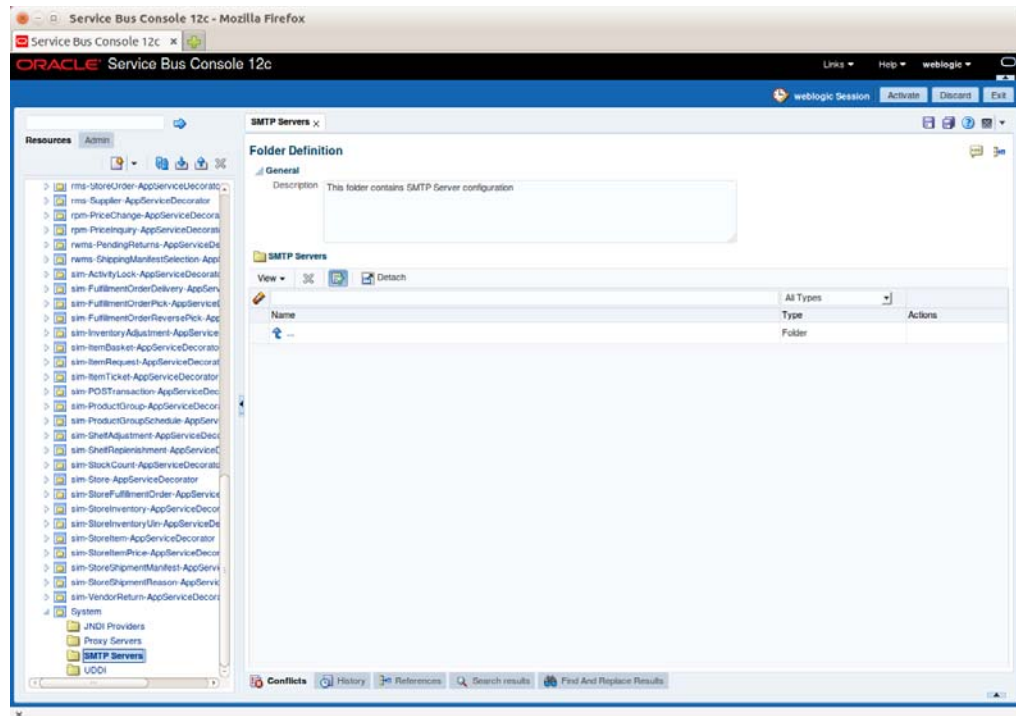
This finishes up the steps for adding new business alert to a message flow in a decorator.

How to add E-mail Notification for Alerts

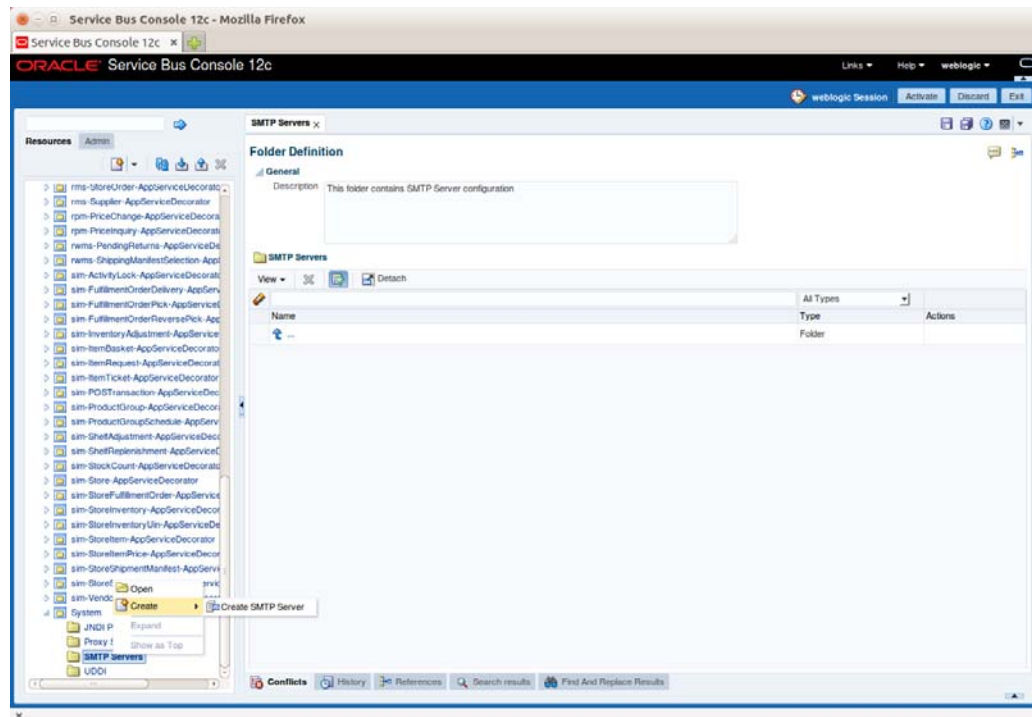
The default alert destination created by RSB only sends alerts on OSB reporting provider JMS. The alert destination can also be configured to send email notifications; this will be useful to get immediate notifications for SLA alerts. For generating email notifications first step is to create SMTP server 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 > SMTP Servers** page.



3. Right click SMTP Servers and select **Create > Create SMTP Server**.



4. Create SMTP Server window is displayed.

Create SMTP Server

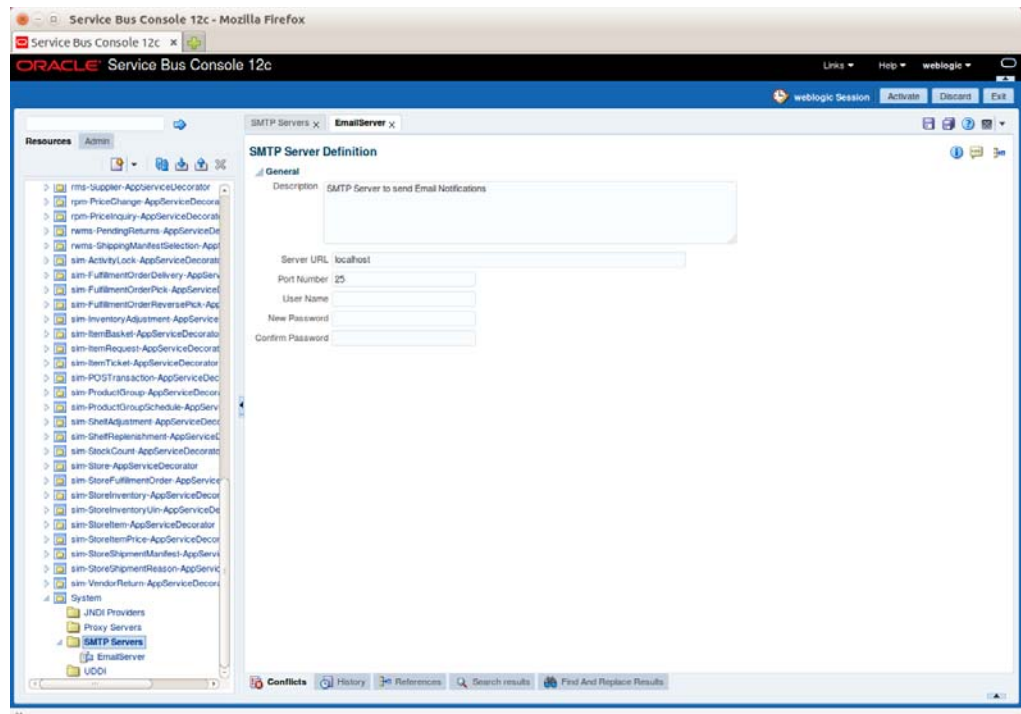
* Resource Name

Description

* Server URL

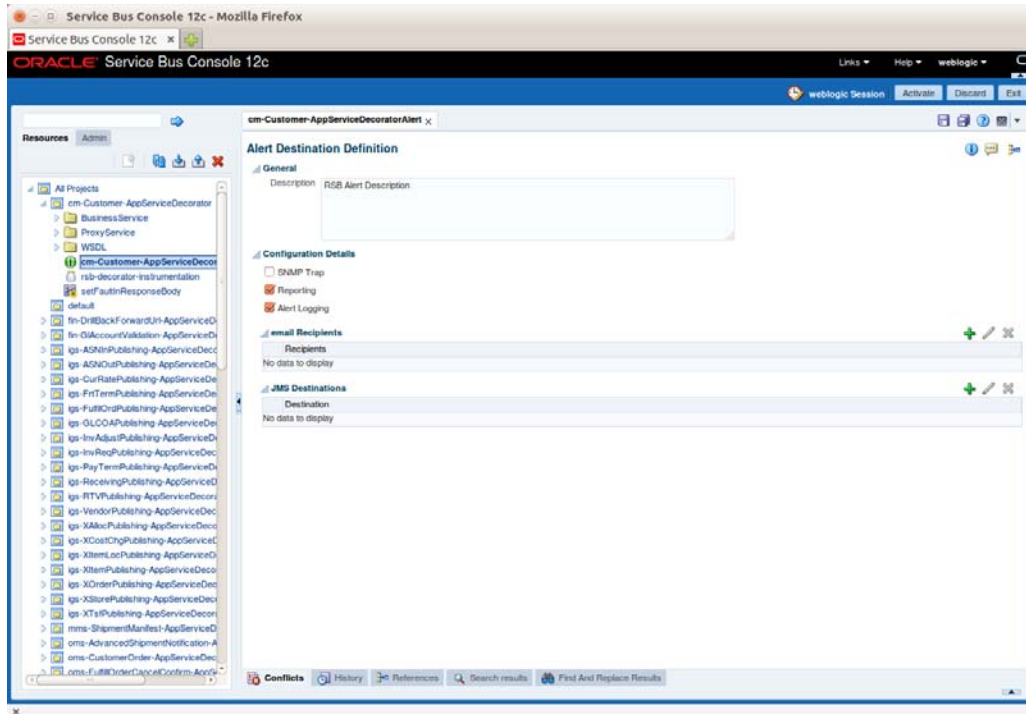
* Port Number 25

5. Provide a name and 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:

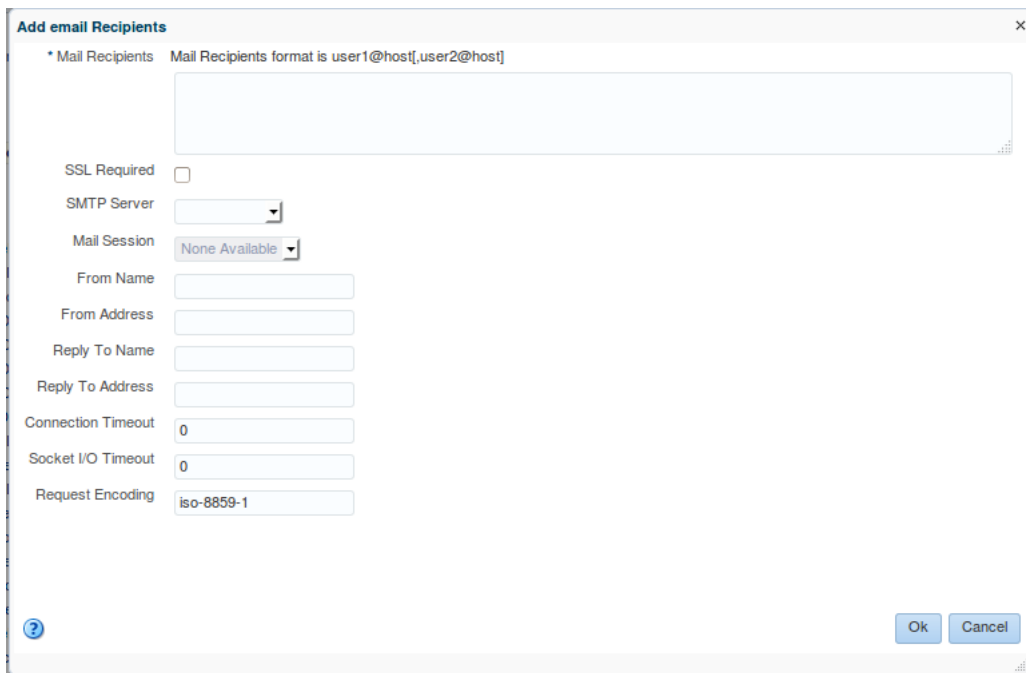


After creating SMTP server configuration, we need to update alert destination to use the SMTP server for sending notifications.

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. In the e-mail Recipients section, click add. 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.

Add email Recipients

• Mail Recipients Mail Recipients format is user1@host[,user2@host]

receiver@host.com

SSL Required

SMTP Server EmailServer

Mail Session None Available

From Name EmailServer

From Address sender@host.com

Reply To Name EmailServer

Reply To Address sender@host.com

Connection Timeout 0

Socket I/O Timeout 0

Request Encoding iso-8859-1

The email address to reply to

Ok Cancel

5. Click **OK** 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.

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.

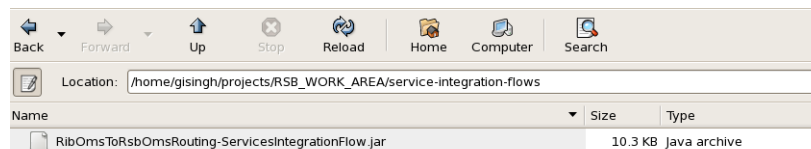
Injector Service Implementation in RSB

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 RsbServiceIntegrationFlowPak15.0.0ForRibOmsToRsbOmsRouting_eng_ga.zip PAK. The OSB jar packaged inside the PAK is RibOmsToRsbOmsRouting-ServicesIntegrationFlow.jar.

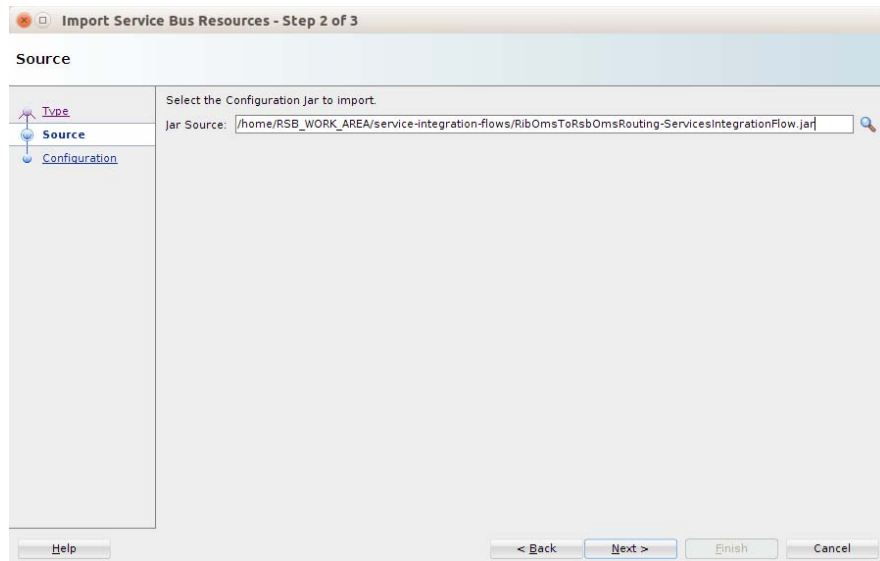
This OSB jar contains a Proxy Service which is based on Injector Service WSDL. The name of the proxy service is RibOmsToRsbOmsRoutingService. The WSDL contains an operation named as injectMessage(). This operation requires four parameters: message family, message type, business object ID and payload. 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 JDeveloper

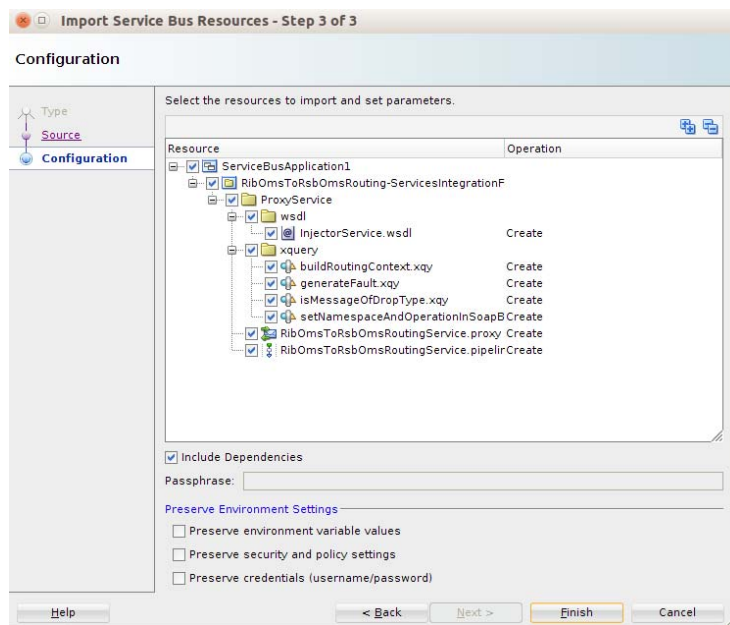
1. Copy the RibOmsToRsbOmsRouting-ServicesIntegrationFlow.jar to RSB_WORK_AREA/service-integration-flows folder. The directory structure looks like this:



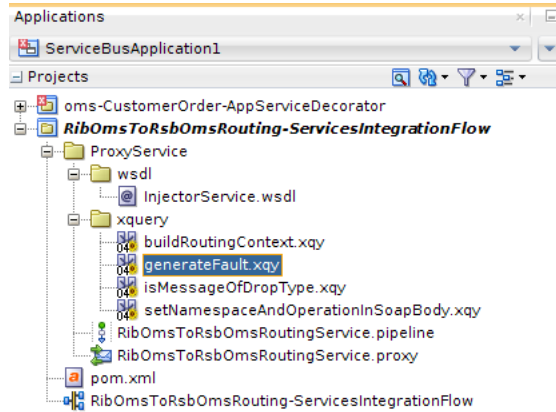
2. Now we need to import this jar in JDeveloper. To import the jar, select **File > Import > Service Bus Resources**.



3. Click **Browse** and select the jar file.
4. Click **Next**. It will show all the files available in the jar.

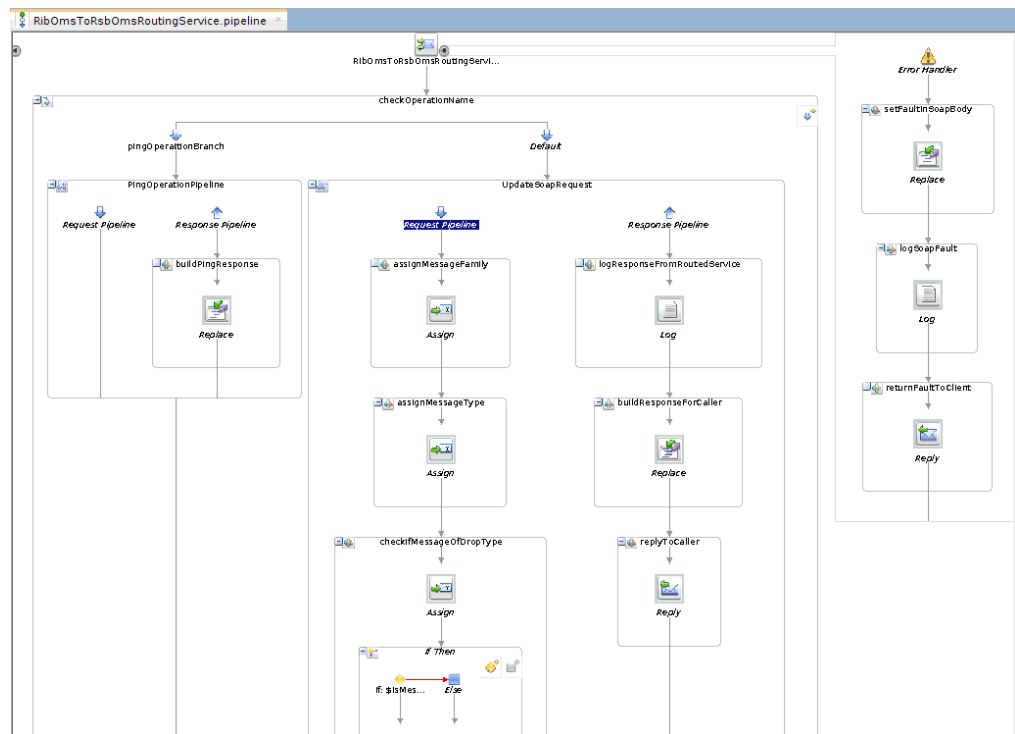


5. Click **Finish**. The jar is imported as an OSB project and the project structure looks like the following:

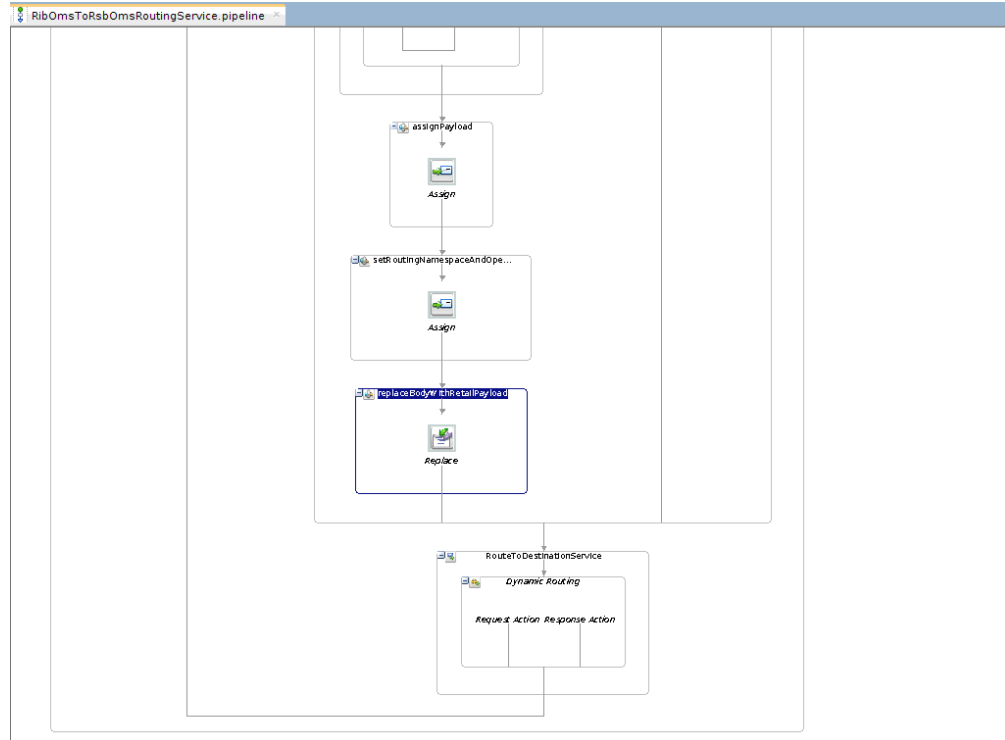


Message Flow in RSB-OMS Routing Service

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 remaining part 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:

Message Family	Message Type
pendreturn	pendretcre
pendreturn	pendretmod
pendreturn	pendretdel
pendreturn	pendretdlcre
pendreturn	pendretdlmod
pendreturn	pendretdlldel
asnout	asnoutmod
receiving	appointcre
receiving	appointdel
receiving	receiptcre
receiving	receiptmod
receiving	appointdlcre
receiving	appointdlldel
receiving	appointdlmod
receiving	appointhdrmod

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

Message Family	Message Type	Decorator Service URI
pendreturn	rtrnrcptnotify	oms-OrderReturn-AppServiceDecorator/ProxyService/OrderReturnAppServiceLocalProxy
pendreturn	rtrncomplete	oms-OrderReturn-AppServiceDecorator/ProxyService/OrderReturnAppServiceLocalProxy
sostatus	sostatuscre	oms-StockOrderStatus-AppServiceDecorator/ProxyService/StockOrderStatusAppServiceLocalProxy
asnout	asnoutcre	oms-AdvancedShipmentNotification-AppServiceDecorator/ProxyService/AdvancedShipmentNotificationAppServiceLocalProxy

Message Family	Message Type	Decorator Service URI
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)
```

