

Oracle® Fusion Middleware
Developer's Guide for Oracle Service Bus
11g Release 1 (11.1.1.3)
E15866-01

April 2010

Oracle Fusion Middleware Developer's Guide for Oracle Service Bus, 11g Release 1 (11.1.1.3)

E15866-01

Copyright © 2008, 2010, Oracle and/or its affiliates. All rights reserved.

Primary Authors: Floyd Jones, Legacy authors

Contributing Author:

Contributor:

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

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software 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 which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on 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. 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.

Contents

Preface	xliii
Documentation Accessibility	xliii
Conventions	xliii
Part I IDE Help for Oracle Service Bus	
1 Introduction to Oracle Service Bus	
2 Tasks	
Working with Projects, Folders, Resources, and Configurations	2-1
Editing Resources.....	2-2
Cloning Oracle Service Bus Projects and Folders.....	2-2
Creating Oracle Service Bus Configuration Projects.....	2-2
Creating Oracle Service Bus Projects.....	2-2
Creating Custom Resources.....	2-3
Creating JNDI Provider Resources.....	2-3
Creating Proxy Server Resources.....	2-3
Creating Message Format Files	2-3
Editing JNDI Provider Resources	2-3
Editing Proxy Server Resources.....	2-3
Exporting Resources	2-4
Using the Export Wizard	2-4
Using the Command Line or a Script to Export an Oracle Service Bus Configuration....	2-4
Before You Begin.....	2-4
Exporting a Configuration Using Ant	2-4
Exporting a Configuration Using WLST	2-5
Exporting a Configuration Using the Command Line.....	2-5
Generating an Effective WSDL	2-6
Modifying JAR Dependencies.....	2-6
Importing Resources.....	2-6
Using the Import Wizard.....	2-7
Using the Command Line or a Script to Import an Oracle Service Bus Configuration....	2-7
Creating Servers	2-7
Creating Service Account Resources.....	2-7
Creating Service Key Provider Resources	2-7

Creating SMTP Server Resources	2-7
Creating XQuery Transformations	2-8
Creating XSL Transformations.....	2-8
Working with Business Services	2-8
Creating Business Services.....	2-8
Generating a Business Service from an Existing Service.....	2-8
Generating a JCA Business Service from an Outbound JCA File	2-9
Generating a Business Service from Oracle Enterprise Repository	2-9
Re-generating an Existing Business Service from Oracle Enterprise Repository	2-9
Editing Business Services.....	2-9
Working with Proxy Services	2-10
Creating Proxy Services	2-10
Generating a Proxy Service from an Existing Service	2-10
Generating a JCA Proxy Service from an Inbound JCA File	2-11
Consuming Oracle Service Bus Proxy Services in Oracle JDeveloper with WSIL	2-11
Editing Proxy Services.....	2-12
Working with Proxy Service Message Flows	2-13
Constructing Proxy Service Message Flows.....	2-14
Adding and Configuring Alert Actions in Message Flows	2-14
Adding and Configuring Assign Actions in Message Flows	2-15
Adding and Configuring Conditional Branch Nodes in Message Flows	2-15
Adding and Configuring Delete Actions in Message Flows	2-15
Adding and Configuring Dynamic Publish Actions in Message Flows	2-16
Adding and Configuring Dynamic Routing Actions in Message Flows	2-16
Adding and Configuring Error Handlers in Message Flows	2-16
Adding and Configuring For-Each Actions in Message Flows.....	2-17
Adding and Configuring If-Then Actions in Message Flows	2-17
Adding and Configuring Insert Actions in Message Flows	2-18
Adding and Configuring Java Callout Actions in Message Flows.....	2-18
Adding and Configuring Log Actions in Message Flows.....	2-18
Adding and Configuring MFL Transform Actions in Message Flows	2-19
Adding and Configuring Operational Branch Nodes in Message Flows.....	2-19
Adding and Configuring Pipeline Pair Nodes in Message Flows.....	2-20
Adding and Configuring Publish Actions in Message Flows	2-20
Adding and Configuring Publish Table Actions in Message Flows	2-20
Adding and Configuring Raise Error Actions in Message Flows.....	2-21
Adding and Configuring Rename Actions in Message Flows	2-22
Adding and Configuring Replace Actions in Message Flows.....	2-22
Adding and Configuring Reply Actions in Message Flows	2-22
Adding and Configuring Report Actions in Message Flows	2-23
Adding and Configuring Resume Actions in Message Flows	2-23
Adding and Configuring Route Nodes in Message Flows	2-23
Adding and Configuring Routing Actions in Message Flows	2-24
Adding and Configuring Routing Options Actions in Message Flows	2-24
Adding and Configuring Routing Table Actions in Message Flows	2-24
Adding and Configuring Service Callout Actions in Message Flows.....	2-25
Adding and Configuring Skip Actions in Message Flows.....	2-25

Adding and Configuring Stages in Message Flows.....	2-26
Adding and Configuring Transport Headers Actions in Message Flows.....	2-26
Adding and Configuring Validate Actions in Message Flows.....	2-26
Working with Alert Destinations.....	2-27
Creating Alert Destinations.....	2-27
Editing Alert Destinations.....	2-27
Adding E-mail Recipients to Alert Destinations.....	2-27
Adding JMS Destinations to Alert Destinations.....	2-27
Working with MQ Connections.....	2-28
Adding MQ Connections.....	2-28
Editing MQ Connections.....	2-28
Working with Oracle Enterprise Repository and Harvester.....	2-28
Generating Business Services from Oracle Enterprise Repository.....	2-29
Using Harvester.....	2-30
Using Harvester from Eclipse.....	2-31
Using Harvester from the Command Line or a Script.....	2-32
Performing Queries in Oracle Enterprise Repository from Eclipse.....	2-33
Working with UDDI Registries.....	2-33
Adding UDDI Registries.....	2-34
Configuring UDDI Registries.....	2-34
Importing Business Services From a UDDI Registry.....	2-34
Working with Split-Join.....	2-34
Introduction to Split-Join.....	2-35
Using Split-Join with Content in SOAP Headers.....	2-35
Transaction Support.....	2-36
Security with Split-Joins.....	2-36
Designing a Split-Join.....	2-36
Initial Setup.....	2-37
Creating/Importing a WSDL Containing the Base Operation.....	2-37
Creating/Importing a Business Service to Use the Split-Join.....	2-37
Designing a Static Split-Join.....	2-37
1. Creating a New Split-Join.....	2-37
2. Adding an Assign.....	2-38
3. Adding a Parallel Node.....	2-38
4. Adding an Assign for Each Branch.....	2-39
5. Adding an Invoke Service.....	2-39
6. Adding an Assign for Each Branch.....	2-39
7. Exporting and Testing the Split-Join.....	2-39
Designing a Dynamic Split-Join.....	2-40
1. Creating a New Split-Join.....	2-41
2. Adding an Assign.....	2-42
3. Adding a For Each.....	2-42
4. Adding an Assign.....	2-42
5. Adding an Invoke Service.....	2-42
6. Adding an Assign.....	2-43
7. Adding an Error Handler.....	2-43
8. Exporting and Testing the Split-Join.....	2-43

Creating a New Split-Join	2-44
Configuring the Start Node	2-45
Adding General Information.....	2-46
Defining Global Variables.....	2-46
Viewing External Services	2-47
Configuring a Receive	2-47
Viewing the Operation.....	2-47
Defining the Receive Variable.....	2-47
Adding General Information.....	2-48
Creating an Assign.....	2-48
Adding and Configuring Assign Operations	2-48
Adding an Operation to an Assign	2-49
Adding a Copy Operation.....	2-49
Adding General Information.....	2-49
Invoking a Service.....	2-50
Selecting an Operation	2-50
Defining Input and Output Variables.....	2-50
Adding General Information.....	2-51
Creating a Parallel.....	2-51
Adding Nodes	2-51
Adding General Information.....	2-52
Creating a For Each.....	2-52
Defining the For Each Logic	2-52
Adding General Information.....	2-52
Creating an If Activity	2-53
Configuring the If	2-53
Writing the logic of the condition.....	2-53
Adding resulting nodes	2-54
Adding General Information	2-54
Adding and Configuring Else If	2-54
Writing the Logic of the Condition	2-54
Adding Resulting Nodes	2-54
Adding General Information	2-54
Configuring the Else.....	2-55
Adding Resulting Nodes	2-55
Adding General Information.....	2-55
Creating an Error Handler.....	2-55
Creating a Raise Error.....	2-56
Configuring a Reply.....	2-56
Viewing the Operation.....	2-56
Defining the Reply Variable	2-56
Adding General Information.....	2-58
About Scope	2-58
Scope and Variables.....	2-58
Exporting and Testing a Split-Join.....	2-59
Creating a Transport Typed Business Service	2-59
Exporting the Split-Join Files.....	2-59

Exporting from the Business Service Menu	2-59
Auto-export	2-60
Manual export	2-60
Testing the Split-Join in the Test Console.....	2-61
Exporting from the Business Service Menu	2-61
Using the Oracle Service Bus Debugger.....	2-61
Enabling Debugging.....	2-62
Using Standard Debugging	2-62
Debug Views.....	2-62
Step Actions and Breakpoints	2-64
Using the Oracle Service Bus Debugger Launch Configuration.....	2-64
Remote Debugging	2-65
Debugging Oracle Service Bus Running Stand-Alone on a Managed Server.....	2-65
Server Sharing	2-65

3 Transport Configuration

Protocol-Specific Transport Configuration Pages.....	3-1
BPEL-10g Transport Configuration Page (Business Services).....	3-2
DSP Transport Configuration Page (Business Services).....	3-3
EJB Transport Configuration Page (Business Services).....	3-4
E-Mail Transport Configuration Page (Business Services)	3-5
E-Mail Transport Configuration Page (Proxy Services).....	3-6
File Transport Configuration Page (Business Services).....	3-7
File Transport Configuration Page (Proxy Services).....	3-7
FTP Transport Configuration Page (Business Services).....	3-8
FTP Transport Configuration Page (Proxy Services).....	3-9
HTTP Transport Configuration Page (Business Services)	3-10
HTTP Transport Configuration Page (Proxy Services)	3-13
JCA Transport Configuration Page (Proxy and Business Services).....	3-14
JEJB Transport Configuration Page (Business Services).....	3-16
JEJB Transport Configuration Page (Proxy Services)	3-17
JMS Transport Configuration Page (Business Services).....	3-18
JMS Transport Configuration Page (Proxy Services).....	3-21
MQ Transport Configuration Page (Business Services)	3-24
MQ Transport Configuration Page (Proxy Services).....	3-26
SB Transport Configuration Page (Business Services).....	3-28
SB Transport Configuration Page (Proxy Services)	3-29
SFTP Transport Configuration Page (Business Services).....	3-29
SFTP Transport Configuration Page (Proxy Services).....	3-30
SOA-DIRECT Transport Configuration Page (Business Services).....	3-32
Tuxedo Transport Configuration Page (Business Services).....	3-33
Tuxedo Transport Configuration Page (Proxy Services).....	3-35
WS Transport Configuration Page (Business Services)	3-36
WS Transport Configuration Page (Proxy Services).....	3-37

4 User Interface Reference

Alerts	4-1
Alert Destination Editor	4-2
Edit E-mail Recipient Page	4-2
Edit JMS Destination Page	4-3
Business Service Configuration	4-3
Business Service General Configuration Page	4-3
Business Service Editor Options	4-4
Business Service Message Type Configuration Page	4-4
Business Service SOAP Binding Configuration Page	4-5
Business Service Transport Configuration Page	4-6
Business Service Message Handling Configuration Page	4-9
Business Service - Service Policy Configuration Page	4-12
Business Service Security Configuration Page	4-13
Proxy Service Configuration	4-13
Proxy Service General Configuration Page	4-14
Proxy Service Editor Options	4-14
Proxy Service Message Type Configuration Page	4-15
Proxy Service Operation Selection Configuration Page	4-16
Proxy Service Message Handling Configuration Page	4-19
Proxy Service SOAP Binding Configuration Page	4-22
Proxy Service Transport Configuration Page	4-23
Proxy Service - Service Policy Configuration Page	4-26
Proxy Service Security Configuration Page	4-26
Oracle Service Bus Configurations and Projects	4-28
Oracle Service Bus Configurations View	4-28
New Oracle Service Bus Configuration Project Wizard	4-29
Oracle Service Bus Configuration Page	4-29
New Oracle Service Bus Project	4-30
Custom Resources	4-30
New Custom Resource Wizard	4-31
New Custom Resource Editor	4-31
New Custom Resource - Resource Type Page	4-31
Custom MQ Resource Configuration Page	4-31
Export Wizard	4-32
Export Wizard - Oracle Service Bus Configuration JAR Export Page.....	4-32
Export Wizard - Export to Server - Select Resources Page	4-33
Export Wizard - Export to Server - Review Resources Page	4-34
Import Wizard	4-34
Import Wizard - Config JAR Import - Load Resources Page	4-34
Import Wizard - Config JAR Import - Review Resources Page	4-35
Import Wizard - Config ZIP Import - Load Resources Page	4-35
Import Wizard - Config ZIP Import - Review Resources Page	4-35
Import Wizard - URL Import - Load Resources Page	4-36
Import Wizard - URL Import - Review Resources Page	4-36
JNDI Providers	4-36
JNDI Provider Editor	4-36

New JNDI Provider Resource Wizard	4-37
Proxy Servers	4-37
Message Flow Design Palette	4-38
Message Flow Nodes	4-38
Message Flow Route Actions - Communication Actions	4-39
Message Flow Route Actions - Flow Control Actions	4-39
Message Flow Stage Actions - Communication Actions	4-40
Message Flow Stage Actions - Flow Control Actions	4-40
Message Flow Stage Actions - Message Processing Actions	4-41
Message Flow Stage Actions - Reporting Actions	4-42
Message Flow Editor	4-42
Alert Action Properties	4-44
Assign Action Properties	4-44
Conditional Branch Node Properties	4-45
Delete Action Properties	4-46
Dynamic Publish Action Properties	4-47
Dynamic Routing Action Properties	4-48
Error Handler Node Properties	4-48
For-Each Action Properties	4-48
If-Then Action Properties	4-49
If Action and Else-If Action Properties	4-50
Else Action Properties	4-50
Insert Action Properties	4-50
Java Callout Action Properties	4-51
Log Action Properties	4-52
Message Flow Properties - Comment	4-53
Message Flow Properties - Namespaces	4-53
Message Flow Properties - Variables	4-54
MFL Transform Action Properties	4-54
Operational Branch Node Properties	4-55
Pipeline Pair Node Properties	4-55
Publish Action Properties	4-56
Publish Table Action Properties	4-57
Publish Table Properties	4-57
Case Action Properties	4-58
Publish Action Properties	4-58
Raise Error Action Properties	4-58
Transactions	4-59
Rename Action Properties	4-59
Replace Action Properties	4-59
Reply Action Properties	4-60
Report Action Properties	4-61
Resume Action Properties	4-62
Route Node Properties	4-63
Routing Action Properties	4-63
Routing Options Action Properties	4-64
Routing Table Action Properties	4-65

Routing Table Properties	4-66
Case Action Properties	4-66
Routing Action Properties	4-67
Service Callout Action Properties	4-67
Skip Action Properties	4-70
Stage Node Properties	4-70
Transport Headers Action Properties	4-70
Validate Action Properties	4-72
Modify JAR Dependencies Dialog	4-73
SMTP Servers	4-73
Edit SMTP Server Page	4-74
New SMTP Server Resource Wizard	4-74
UDDI Registry Configuration Page	4-74
Outline view - Oracle Service Bus	4-75
Resource Management	4-75
Preferences dialog - Oracle Service Bus - Type Associations Page	4-75
References View	4-76
Select Clone Target Dialog	4-76
Select a Resource Dialog	4-76
New Service Key Provider Resource	4-76
New WS-Policy	4-77
Service Accounts	4-77
New Service Account Resource	4-78
Service Account Editor - General Configuration Page	4-78
Service Account Editor - Static User Configuration Page	4-78
Service Account Editor - User Mappings Configuration Page	4-79
Expression Editors	4-79
XQuery/XSLT Expression Editor	4-79
XPath Expression Editor	4-80
Condition Editor	4-80
Condition Builder Page	4-81
Expression Page	4-81
XQuery Resource Page	4-81
XSLT Resource Page	4-82
Dynamic XQuery Page	4-82
Variable Structures Page	4-83
Namespace Definitions Page	4-83
XQuery Functions Page	4-84
Add Variable Structure Dialog	4-84
New XSL Transformation	4-85
Split-Join User Interface Reference	4-85
Split-Join Design Palette	4-86
Operations	4-86
Global / Start Node Properties	4-88
Variable Properties	4-88
Error Handler Properties	4-89
Invoke Service Properties	4-89

Invoking Another Split-Join	4-90
Reply Properties	4-91
For Each Properties	4-91
If Properties	4-92
If and Else If Properties	4-92
Parallel Properties	4-93
Raise Error Properties.....	4-93
Repeat Until Properties	4-93
Re-Raise Error Properties.....	4-93
Scope Properties	4-94
While Properties.....	4-94
Wait Properties	4-94
Assign Properties	4-95
Assign Operation Properties	4-95
Copy Properties.....	4-96
Delete Properties	4-96
Insert Properties	4-97
Java Callout Properties.....	4-98
Log Properties.....	4-99
Replace Properties.....	4-99
Receive Properties.....	4-100
Counter Variable Dialog	4-100
Create/Edit Variable Dialog.....	4-100
Scope and Variables.....	4-101
Create Message Variable Dialog	4-101
Service Browser	4-101
SOAP Fault Variable Dialog	4-101
WSDL Browser	4-102
Split-Join Wizard - New Split-Join.....	4-102
Split-Join Wizard - Specify Operations	4-102

Part II XQuery Mapper

5 Introduction

Overview of XQuery Mapper	5-1
Support for XQuery 2002 and 2004.....	5-2
Restrictions Applicable to the XQuery Test View	5-2

6 Transforming Data Using XQuery Mapper

Launching XQuery Mapper.....	6-1
Importing the XQuery Mapper Sample Project	6-1
Creating an XQuery Mapper Project	6-2
Importing and Creating Schema Files.....	6-2
Importing XML Schemas and MFL Files.....	6-3
Creating XML Schemas	6-4
Creating XML Files from XML Schemas	6-4

Creating WSDL Files.....	6-5
Creating MFL Files.....	6-5
Selecting Source and Target Data Types	6-5
Creating Data Transformations	6-7
Creating Basic Element Transformations	6-7
Prerequisite	6-7
Creating Element-to-Element Links.....	6-7
Creating Basic Attribute Transformations.....	6-8
Prerequisite	6-8
Creating an Attribute-to-Element Link.....	6-8
Creating Complex Transformations.....	6-9
Prerequisite	6-9
Creating a Complex Transformation	6-9
Editing Data Transformations.....	6-10
Viewing and Editing XQuery Files.....	6-10
Creating Joins and Unions.....	6-11
Creating If-Then-Else Expressions.....	6-11
Creating For-Let-Where-Order By-Return (FLWOR) Expressions.....	6-13
Creating Typeswitch Expressions.....	6-14
Inserting XQuery Functions	6-15
Inserting Expression Variables.....	6-16
Viewing Schema Properties.....	6-17
Restricting Output of Optional Elements	6-17
Testing Data Transformations.....	6-18
Features of the Test View	6-18
Related Topics	6-21
Graphical Features in Design View.....	6-21
Right-Click Menu Options.....	6-21
Link Patterns	6-22
Link Colors.....	6-23
XML Global Elements, Global Types, Local Elements, and Attributes.....	6-24

7 Examples: Data Transformation Using XQuery Mapper

Combining Data from Different Schemas	7-1
Mapping Repeating Elements and Creating Joins	7-4
Step 1. Create an XQuery File.....	7-4
Step 2. Add a Constraint	7-6
Step 3. Add Data to Return Element	7-7
Step 4. Add Function to Calculate Value of Quote	7-7
Step 5. Add a Constraint with Multiple Conditions	7-9
Test the XQuery.....	7-10
Creating Unions.....	7-10
Creating Repeating-Source to Nonrepeating-Target Transformations	7-12
Creating Nonrepeating-Source to Repeating-Target Transformation.....	7-15
Creating Nested If-Then-Else Expressions.....	7-18
Step 1. Create the XQuery Transformation	7-18
Step 2. Create the First "If" Condition	7-19

Step 3. Create the First Nested If-Then-Else Condition.....	7-20
Step 4. Create the Second Nested If-Then-Else Condition.....	7-20
Creating FLWOR Expressions.....	7-21
Using Recursive Schemas.....	7-23
Grouping Data by Key Fields.....	7-25
8 Upgrading XQuery Code	
Upgrading Inline XQuery Code.....	8-1
Upgrading XQuery Files.....	8-2
Part III Format Builder	
9 Introduction	
Overview.....	9-1
10 Format Builder Main Window	
Using the Menu Bar.....	10-1
Using the Toolbar.....	10-1
Using the Tree Pane.....	10-2
Using the Shortcut Menus.....	10-3
Using Drag and Drop.....	10-3
Valid Names.....	10-4
11 Message Format Detail Window	
12 Field Detail Window	
13 Group Detail Window	
14 Reference Detail Window	
15 Comment Detail Window	
16 Format Builder Options	
17 Importing Metadata	
Importing a Guideline XML File.....	17-1
Importing an XML Schema.....	17-1
Importing a COBOL Copybook.....	17-2
Importing C Structures.....	17-3
Starting the C Structure Importer.....	17-3
Generating MFL Data.....	17-4
Generating C Code.....	17-4
Importing an FML Field Table Class.....	17-5

FML Field Table Class Importer Prerequisites	17-5
Sample FML Field Table Class Files	17-6
Creating XML with the FML Field Table Class Importer	17-6

18 Format Tester

Format Tester Window	18-1
Format Tester Menus	18-1
File Menu	18-1
Edit Menu	18-2
Display Menu.....	18-2
Generate Menu	18-3
Transform Menu	18-3
Shortcut Menu	18-3
Using the Non-XML Window	18-3
See Also.....	18-4
Using the Data Offset Feature.....	18-4
Using the Text Feature	18-4
Using the XML Window	18-4
Using the Debug Window	18-4
Using the Resize Bars	18-5
Debugging Format Definitions	18-5
Searching for Values	18-5
Searching for Offsets	18-6
Using the Debug Log.....	18-6

19 Format Builder Menus

File Menu	19-1
Edit Menu	19-1
Insert Menu	19-2
View Menu	19-3
Tools Menu	19-3
Help Menu.....	19-3
Shortcut Menu	19-3

20 How Do I?

Create a Message Format	20-1
Create a Group	20-1
Create a Field.....	20-2
Create a Comment	20-2
Create a Reference	20-3
Save a Document	20-3
Use Format Tester	20-4
Debug Format Definitions	20-4
Search for Values	20-4
Search for Offsets	20-5
Use the Debug Log.....	20-5

Character Delimiters.....	20-5
Specify a Delimiter	20-6
Specify by Reference.....	20-6
Specify by Value	20-7
Delimiter Match Rule.....	20-7
Data Delimiter	20-7
Escape Character	20-7
None.....	20-8

21 Using the Palette

Displaying the Palette Window.....	21-1
Adding Items to the Palette.....	21-1
Adding Palette Items to a Message Format	21-2
Using the File Menu	21-2
Using the Shortcut Menu.....	21-2

22 Format Builder Supported Data Types

MFL Data Types.....	22-1
COBOL Copybook Importer Data Types	22-5
Unsupported C Language Features.....	22-6

Part IV General Development Topics

23 Creating and Using Custom XPath Functions

Registering Custom Functions with Oracle Service Bus.....	23-1
Creating and Packaging the Custom Function Java Classes.....	23-2
Creating the Class and Method.....	23-3
Using Single-Dimensional Arrays.....	23-3
Packaging the Custom Function Class.....	23-4
Using Custom Functions.....	23-4
Using Custom Functions in Inline XQuery Expressions and XQuery Resources	23-4
Using Custom Functions in XSLT Resources.....	23-4
Testing Custom XPath Functions in Eclipse.....	23-5
Deploying Custom Functions in a Cluster.....	23-5

Part V Transports

24 Oracle SOA Suite Transport (SOA-DIRECT)

About the SOA-DIRECT Transport.....	24-1
WS-Addressing.....	24-2
Security	24-2
Environment Values	24-3
Error Handling	24-3
Connection Errors.....	24-3
Application Errors	24-3

Generic Errors.....	24-3
Using SOA Suite Services with Oracle Service Bus.....	24-3
Simple Use Cases – Synchronous	24-4
Synchronous Invocation of a SOA Composite	24-4
Creating and Configuring the Services	24-4
Synchronous Invocation from a SOA Composite	24-5
Creating and Configuring the Services	24-5
Associating Messages with the Correct Conversation	24-6
Advanced Use Cases – Asynchronous.....	24-6
Asynchronous Invocation of a SOA Composite.....	24-6
Creating and Configuring the Services	24-6
Asynchronous Invocation from a SOA Composite.....	24-8
Creating and Configuring the Services	24-9
Transport Configuration Reference	24-10
SOA-DIRECT Endpoint URI	24-10
Cluster URI	24-11
URI Examples	24-11
SOA-DIRECT Transport Configuration for Business Services.....	24-12
WS-Addressing Reference.....	24-13
ReplyTo Header	24-13
Calling a SOA Composite Asynchronously with a SOA-DIRECT Business Service ...	24-14
Calling Back to a SOA Composite Asynchronously with a SOA-DIRECT Business Service	24-14
MessageID / RelatesTo Headers	24-14
XML Examples	24-14
Conversation ID Examples	24-14
Port and Message Definitions	24-15
WS-Addressing that Sets the Conversation ID.....	24-16
Message Payload Data that Sets the Conversation ID.....	24-16
Transformation Examples.....	24-18
Asynchronous Composite to Composite Native Communication Through Oracle Service Bus Example	24-19
Port and Message Definitions	24-20
BP1 to P1 – Initiate operation	24-20
P1/B1 to BP2.....	24-20
BP2 to P2 – onResult operation	24-21
P2/B2 to BP1 – onResult operation	24-22

25 JCA Transport

About the JCA Transport	25-1
Messaging.....	25-2
Transactions.....	25-2
Transport Headers	25-2
\$inbound and \$outbound Request Headers	25-2
\$inbound-Only Request Headers	25-2
\$outbound-Only Request Headers.....	25-3
Endpoint Properties.....	25-4

Proxy Service Endpoint Properties.....	25-4
Business Service Endpoint Properties.....	25-4
Security	25-5
Proxy Services.....	25-5
Business Services.....	25-6
Logging	25-6
URI Rewriting.....	25-6
Environment Variables.....	25-6
Encoding.....	25-7
Working with Adapters	25-7
Adapter Support.....	25-7
Oracle JCA Adapter Limitations.....	25-8
JCA Adapter Framework.....	25-8
Configuring Adapters – General	25-8
Configuring Adapters that Poll a Database	25-9
Configuring the Oracle JCA Adapter for Database	25-9
Configuring the Oracle JCA Adapter for AQ	25-10
Rejected Messages.....	25-10
Invoking an EIS Service Through Oracle Service Bus	25-10
Creating, Configuring, and Invoking the Services.....	25-10
Invoking an External Service from an EIS	25-11
Creating, Configuring, and Invoking the Services.....	25-11
Transport Configuration Reference	25-12
Endpoint URI.....	25-12
Endpoint Redeployment.....	25-12
JCA Transport Configuration for Proxy and Business Services.....	25-12
Proxy Service Operation Configuration	25-14
Proxy Service Message Handling	25-14
Policies	25-14
Business Service Retry Application Errors.....	25-14

26 HTTP and Poller Transports

HTTP Transport	26-1
Configuring Proxy Services using the HTTP Transport	26-1
Configuring Business Services using the HTTP Transport.....	26-3
REST Support.....	26-6
REST in Proxy Services	26-6
XQuery Examples	26-7
Headers	26-7
REST in Business Services.....	26-7
Response Codes for HTTP Business Services.....	26-8
E-mail Transport	26-9
Configuring Proxy Services Using the E-mail Transport	26-9
Configuring Business Services Using the E-mail Transport.....	26-10
File Transport	26-11
Configuring Proxy Services using the File Transport.....	26-11
Configuring Business Services using the File Transport	26-12

FTP Transport	26-13
Configuring Proxy Services using the FTP Transport	26-13
Configuring Business Services using the FTP Transport	26-14
SFTP Transport	26-15
Environment Values	26-16
General Principles of SFTP Authentication	26-16
Run-Time Behavior	26-17
Using the SFTP Transport	26-18
Enabling SFTP Authentication	26-18
Creating the Known Hosts File	26-18
Enabling Username-Password Authentication	26-19
Enabling Host-Based Authentication	26-19
Enabling Public Key Authentication	26-20
Configuring Proxy Services	26-20
Configuring Transport Headers and Metadata	26-23
Configuring Business Services	26-24
Handling Communication Errors	26-25
Troubleshooting	26-26
Importing Resources	26-26
Importing and Publishing Services: UDDI Registries	26-26

27 SB Transport

Environmental Values	27-2
Configuring Proxy Services to Use the SB Transport	27-2
Configuring Business Services to Use the SB Transport	27-3
JNDI Provider	27-4
Handling Errors	27-5
UDDI	27-5
Publishing a Service	27-5
Importing a Service	27-6

28 EJB Transport

Introduction	28-1
Invoking EJBs from Oracle Service Bus	28-2
Register a JNDI Provider Resource	28-2
Adding a JNDI Provider	28-2
Register an EJB Client JAR Resource	28-3
Adding a Client or Converter JAR	28-3
Create a Service Account (Optional)	28-3
Locate an EJB in the JNDI Tree	28-3
Transport Configuration Reference	28-3
EJB Endpoint URI	28-4
EJB Transport Configuration for Business Services	28-4
Invoking EJB Business Services	28-5
Exposing EJBs as Web Services	28-5
Advanced Topics	28-6
Transaction Processing, Retries, and Error Handling	28-6

Transactions.....	28-6
Retries and Failover.....	28-7
Error Handling.....	28-8
Supported Types and Converter Class.....	28-8
Converter Classes.....	28-8
Troubleshooting.....	28-9
Enabling Debug Mode.....	28-9
Temp Directories.....	28-9
Deployed Application.....	28-9
Errors.....	28-9
29 JEJB Transport	
About the JEJB Transport.....	29-1
Difference Between the JEJB Transport and the EJB Transport.....	29-1
Environment Values.....	29-2
WSDL Generation.....	29-2
Error Handling.....	29-2
Exception Propagation in the Response.....	29-2
Java Callout and Service Callout Exceptions.....	29-3
Application and Connection Errors.....	29-3
Connection Errors.....	29-3
Application Errors.....	29-4
Creating and Configuring JEJB Services.....	29-4
Creating and Packaging Your Client EJB JAR.....	29-4
Register a JNDI Provider Resource (Business Services).....	29-5
Adding a JNDI Provider.....	29-5
Configuring a JEJB Proxy or Business Service.....	29-5
Use Cases.....	29-6
EJB Invokes an External Service.....	29-6
Non-EJB Client Invokes an EJB.....	29-7
EJB Invokes EJB.....	29-7
Transport Configuration Reference.....	29-8
JEJB Endpoint URI.....	29-8
Proxy Service JEJB Endpoint URI.....	29-8
Business Service JEJB Endpoint URI.....	29-9
JEJB Transport Configuration for Proxy Services.....	29-9
JEJB Transport Configuration for Business Services.....	29-10
Testing JEJB Services.....	29-12
UDDI Integration.....	29-12
UDDI Publish.....	29-12
UDDI Import.....	29-12
30 JMS Transport	
Overview of JMS Interoperability.....	30-1
Asynchronous Request-Response Messaging.....	30-2
Using SOAP-JMS Transport.....	30-2

Interoperating with Oracle WebLogic Server	30-3
Configuring the Response Queues for Cross-domain JMS Calls.....	30-3
Naming Guidelines for Domains, WebLogic, and JMS Servers	30-4
Specifying the JMS Type for Services	30-4
WSDL-Defined SOAP Fault Messages	30-4
Interoperability with WebSphere MQ	30-6
Message ID and Correlation ID Patterns for JMS Request/Response.....	30-6
Overview of JMS Request-Response and Design Patterns	30-6
Patterns for Messaging.....	30-7
JMS Message ID Pattern.....	30-8
JMS Correlation ID Pattern.....	30-9
Comparison of Message ID and Correlation ID Patterns.....	30-9
Interoperating with JAX-RPC Over JMS	30-10
Invoking a JAX-RPC Web Service Using the JMS Message ID Pattern	30-10
Invoking a JMS Request-Response Proxy Service from a JAX-RPC Client.....	30-11
JMS Message ID Pattern Examples.....	30-12
Example 1: An MQ Service Uses a JMS Message ID to Correlate the Request-Response Message	30-12
Example 2: A JAX-RPC Client with Oracle Service Bus Proxy Service.....	30-13
Example 3: Oracle Service Bus as a Client of an Oracle WebLogic Server JAX-RPC Request/Response Service	30-13
Using the JMS Transport.....	30-14
Configuring Proxy Services using JMS Transport Protocol.....	30-14
Transport Headers	30-14
Configuring Transport Headers	30-16
Configuring Business Services using JMS Transport Protocol.....	30-17
Error Handling	30-18

31 Local Transport

Introduction.....	31-1
Features and Characteristics of Local Transport Proxy Services	31-1
Message Handling for Local Transport Proxy Services	31-2
Usage of Local Transport Proxy Services	31-2
Limitations.....	31-4

32 WS Transport

Supported Functionality	32-2
Messaging Patterns	32-2
Policies	32-2
WS-Policy Configurations	32-3
Streaming Content for Large Messages	32-3
Web Services Interoperability	32-3
Authentication and Authorization of Services	32-3
Proxy Service Authentication	32-3
Proxy Service Authorization	32-4
Business Service Authentication	32-4
Using the WS Transport	32-4

Adding Resources to an Oracle Service Bus Domain	32-5
Configuring WS Policies	32-5
Attaching WS Policies to a Service	32-5
Configuring an Error Queue	32-6
Configuring Proxy Services to Use the WS Transport.....	32-6
Assigning Transport Access Control to Proxy Services	32-8
Adding Policy Conditions	32-10
Routing the WS Transport Through an HTTP Proxy Server	32-15
Configuring Business Services to Use the WS Transport	32-15
Error Handling	32-17
Importing and Exporting Resources	32-17
Importing and Publishing Services Using UDDI Registries	32-17

33 MQ Transport

Key Features	33-1
Advantages of Using the MQ Transport	33-1
Supported Service Types	33-2
Messaging Patterns	33-2
Environment Values	33-3
Quality of Service	33-3
MQ Clusters and the MQ Transport	33-4
Using the MQ Transport	33-4
Adding MQ Client Libraries to Your Environment.....	33-4
MQ Connection Resources.....	33-5
Creating an MQ Connection Resource	33-5
Configuring Proxy Services to Use the MQ Transport.....	33-6
Configuring Business Services to Use the MQ Transport	33-8
Transport Headers	33-11
Configuring Transport Headers	33-17
About RFH2 Headers	33-18
Error Handling	33-18
Limitations of the MQ Transport.....	33-18
Using the WebSphere JMS MQ Interface	33-19
Using the WebSphere MQ JMS Interface.....	33-19
Messaging Types.....	33-20
Non-Persistent Messaging	33-20
Non-XA Persistent Messaging	33-20
XA Messaging.....	33-20
Tuning WebSphere MQ	33-20

34 Oracle BPEL Process Manager Transport

Before You Begin	34-1
Overview	34-1
SOAP Support.....	34-2
Transaction Propagation	34-2
SSL Support.....	34-2

Environment Variables.....	34-2
Simple Use Cases (Synchronous)	34-3
Synchronous: Invoking Processes in Oracle BPEL Process Manager.....	34-3
Creating and Configuring the Services.....	34-3
Synchronous: Calling External Services from Oracle BPEL Process Manager	34-3
Creating and Configuring the Services.....	34-4
Associating Messages with the Correct Conversation	34-4
Advanced Use Cases (Asynchronous)	34-5
Asynchronous: Invoking Processes in Oracle BPEL Process Manager	34-5
Creating and Configuring the Services.....	34-5
Asynchronous: Calling Service Providers from Oracle BPEL Process Manager.....	34-6
Creating and Configuring the Services.....	34-7
Transport Configuration Reference	34-8
Endpoint URI.....	34-8
bpel-10g Transport Configuration.....	34-8
Security	34-10
Using SSL from Oracle Service Bus to Oracle Servers	34-10
Error Handling	34-10
Application Errors.....	34-11
Connection Errors	34-11
Other Errors	34-11
WS-Addressing Reference	34-11
ReplyTo.....	34-11
Calling a BPEL Process Asynchronously Through Oracle Service Bus	34-11
BPEL Processes Calling External Services Through Oracle Service Bus	34-12
MessageID / RelatesTo	34-12
XML Examples	34-12
Conversation ID Examples	34-12
Port and Message Definitions	34-13
WS-Addressing that Sets the Conversation ID.....	34-13
Message Payload Data that Sets the Conversation ID.....	34-14
Transformation Examples.....	34-16
Asynchronous BPEL to BPEL Through Oracle Service Bus Example	34-17
Port and Message Definitions	34-18
BP1 to P1 – Initiate operation	34-18
P1/B1 to BP2.....	34-18
BP2 to P2 – onResult operation	34-19
P2/B2 to BP1 – onResult operation	34-19

35 Tuxedo Transport

Overview	35-1
Capabilities of the Tuxedo transport.....	35-2
Configuring the Oracle Tuxedo Connector	35-3
Before You Begin.....	35-3
Configuring Oracle Tuxedo Connector	35-4
Using Tuxedo Services from Oracle Service Bus	35-4
Configuring a New Business Service	35-5

Add a New Project.....	35-5
Add a Business Service	35-5
Load Balancing and Failover	35-9
Handling Errors.....	35-9
Testing Your Configuration.....	35-9
Using Oracle Service Bus from Tuxedo.....	35-10
Adding and Configuring a Proxy Service	35-10
Add a New Project.....	35-10
Add a Proxy Service	35-10
Configure the Proxy Service.....	35-12
Testing Your Configuration.....	35-13
Tuxedo Transport Buffer Transformation	35-13
Any XML Service Type	35-14
Messaging Service Type.....	35-14
Tuxedo Transport Transaction Processing	35-15
Inbound Services	35-15
Outbound Services.....	35-16

36 DSP and Oracle Data Service Integrator Transport

Enabling Data Services for Oracle Service Bus.....	36-1
Using the DSP Transport	36-2
Actions Needed Within Oracle Data Service Integrator	36-2
Step 1. Start Your Server	36-2
Step 2. Generate a WSDL from the Data Service.....	36-2
Step 3: Obtaining the Web Service Address.....	36-3
Actions Needed Within Oracle Service Bus	36-3
Step 4: Import the Data Service WSDL into Oracle Service Bus	36-3
Step 5: Create the Business Service.....	36-3
Step 6: Create the Proxy Service	36-4
Step 7: Test Your Setup	36-4

Part VI Transport SDK

37 Introduction

Purpose of this Guide	37-1
Audience for this Guide	37-1
Overview of this Guide.....	37-1

38 Design Considerations

What is a Transport Provider?.....	38-1
What is the Transport SDK?	38-2
Purpose of the SDK.....	38-2
Transport SDK Features	38-3
Handling Inbound and Outbound Messages	38-3
Deploying Transport-Related Artifacts	38-3
Processing Messages Asynchronously	38-4

Transport Provider Modes.....	38-4
Related Features	38-4
Load Balancing.....	38-4
Monitoring and Metrics	38-4
Do You Need to Develop a Custom Transport Provider?	38-4
When to Use the Transport SDK.....	38-5
When Alternative Approaches are Recommended.....	38-5
Transport Provider Components	38-6
Overview	38-6
Design-Time Component	38-7
Runtime Component	38-8
The Transaction Model	38-9
Overview of Transport Endpoint Properties	38-9
Transactional vs. Non-Transactional Endpoints	38-10
Supported Message Patterns	38-10
Support for Synchronous Transactions	38-10
Use Case 1 (Response Pipeline Processing)	38-10
Use Case 2 (Service Callout Processing)	38-11
Use Case 3 (Suspending Transactions)	38-12
Use Case 4 (Multiple URIs)	38-12
The Security Model	38-12
Inbound Request Authentication	38-12
Outbound Request Authentication	38-13
Outbound Username/Password Authentication	38-13
Outbound SSL Client Authentication (Two-Way SSL)	38-14
Outbound JAAS Subject Authentication	38-14
Link-Level or Connection-Level Credentials	38-14
Uniform Access Control to Proxy Services	38-15
Identity Propagation and Credential Mapping	38-15
The Threading Model	38-15
Overview	38-15
Inbound Request Message Thread	38-16
Outbound Response Message Thread	38-17
Support for Asynchrony	38-17
Publish and Service Callout Threading	38-17
Designing for Message Content	38-18
Overview	38-18
Sources and Transformers	38-18
Sources and the MessageContext Object	38-19
Built-In Transformations.....	38-20

39 Developing a Transport Provider

Development Road Map	39-1
Planning.....	39-1
Developing.....	39-2
Packaging and Deploying.....	39-2
Before You Begin	39-2

Basic Development Steps	39-2
1. Review the Transport Framework Components	39-3
2. Create a Directory Structure for Your Transport Project	39-4
3. Create an XML Schema File for Transport-Specific Artifacts	39-4
4. Define Transport-Specific Artifacts	39-4
EndPointConfiguration	39-5
RequestMetaDataXML	39-5
RequestHeadersXML	39-6
ResponseMetaDataXML	39-7
ResponseHeadersXML	39-7
5. Define the XMLBean TransportProviderConfiguration	39-7
6. Implement the Transport Provider User Interface	39-8
7. Implement the Runtime Interfaces	39-9
8. Deploy the Transport Provider	39-10
Important Development Topics	39-10
Handling Messages	39-10
Overview	39-11
Sending and Receiving Message Data	39-11
Request and Response Metadata Handling	39-12
Character Set Encoding	39-12
Co-Located Calls	39-13
Returning Outbound Responses to Oracle Service Bus Runtime	39-13
Transforming Messages	39-13
Working with TransportOptions	39-14
Inbound Processing	39-15
Outbound Processing	39-15
Request Mode	39-15
Handling Errors	39-16
Case 1	39-16
Case 2	39-17
Case 3	39-17
Catching Application Errors	39-18
Identifying Application Errors	39-18
Configuring Application Error Retries	39-18
Catching Connection Errors	39-19
Identifying Connection Errors	39-19
Defining Custom Environment Value Types	39-20
Publishing Proxy Services to a UDDI Registry	39-21
When to Implement TransportWLSArtifactDeployer	39-22
Creating Help for Custom Transports	39-23
Custom Transport Help Overview	39-23
Eclipse Help	39-23
Context-Sensitive Help (F1)	39-23
Eclipse Help Table of Contents	39-24
Oracle Service Bus Console Help	39-25
Providing Custom Transport Help in Eclipse	39-26
Providing Context-Sensitive Help in Eclipse	39-26

Providing Help in the Eclipse Help System.....	39-27
Help Implementation Reference.....	39-27
plugin.xml.....	39-28
toc.xml.....	39-29
context.xml.....	39-29
Help Content and Resources.....	39-30
Packaging Help for the Transport Plug-in.....	39-30
Related Topics.....	39-31
Providing Custom Transport Help in the Oracle Service Bus Console.....	39-31
Implementing the CustomHelpProvider Interface.....	39-32
Creating an HTML File to Launch.....	39-33
Creating a Simple Web Application to Display Expanded Help (Optional).....	39-34
META-INF/application.xml.....	39-34
WEB-INF/web.xml.....	39-35
Help Content and Resources.....	39-35
Packaging Transport Help for the Oracle Service Bus Console.....	39-35

40 Developing Oracle Service Bus Transports for Eclipse

Introduction	40-1
Services Runtime and Services Configuration	40-1
Offline Methods.....	40-2
Restrictions when Working Offline.....	40-4
Working Offline with a Remote Server.....	40-4
Bootstrapping Transports in Offline Mode.....	40-5
Packaging Transports in Offline Mode.....	40-5
Packaging Transports as Eclipse Plug-Ins	40-6
Transport Plug-in Resources.....	40-6
Transport Plug-in Packaging.....	40-7
Reference	40-7
Working in Different Modes.....	40-8
TransportProviderFactory.....	40-9
Extension Point Schema.....	40-10
plugin.xml.....	40-10
MANIFEST.MF.....	40-11
Build.xml.....	40-11
TransportManagerHelper Methods.....	40-11

41 Transport SDK Interfaces and Classes

Introduction	41-1
Schema-Generated Interfaces	41-1
General Classes and Interfaces	41-2
Summary of General Classes.....	41-2
Summary of General Interfaces.....	41-3
Source and Transformer Classes and Interfaces	41-4
Summary of Source and Transformer Interfaces.....	41-4
Summary of Source and Transformer Classes.....	41-4
Metadata and Header Representation for Request and Response Messages	41-6

Runtime Representation of Message Contents	41-6
Interfaces	41-6
User Interface Configuration	41-7
Overview	41-7
Summary of UI Interfaces	41-7
Summary of UI Classes	41-7
42 Sample Socket Transport Provider	
Sample Socket Transport Provider Design	42-1
Concepts Illustrated by the Sample	42-1
Basic Architecture of the Sample	42-2
Configuration Properties	42-2
Sample Location and Directory Structure	42-3
Building and Deploying the Sample	42-4
Setting Up the Environment	42-4
Building the Transport	42-4
Deploying the Sample Transport Provider	42-5
Start and Test the Socket Server	42-5
Start the Socket Server	42-5
Test the Socket Transport	42-6
Configuring the Socket Transport Sample	42-6
Create a New Project	42-6
Create a Business Service	42-7
Create a Proxy Service	42-7
Edit the Pipeline	42-7
Testing the Socket Transport Provider	42-9
43 Deploying a Transport Provider	
Packaging the Transport Provider	43-1
Deploying the Transport Provider	43-1
Transport Registration	43-2
Undeploying a Transport Provider	43-2
Deploying to a Cluster	43-3
Part VII Security	
44 Introduction	
Document Audience	44-1
Related Information	44-2
45 Understanding Oracle Service Bus Security	
Inbound Security	45-1
Outbound Security	45-3
Options for Identity Propagation	45-3
Example: Authentication with a User Name Token	45-10

Administrative Security	45-11
Access Control Policies	45-11
Configuring Proxy Service Access Control	45-12
Access Control Policy Management	45-12
Deleting a Proxy Service	45-13
Deleting the Access Control Policy Assigned to a Proxy Service	45-13
Moving or Renaming a Proxy Service	45-13
Renaming a Proxy Service Operation	45-13
Preserving Security Configuration During Import	45-14
Preserve Security and Policy Configuration Check Box	45-14
Preserve Credentials Check Box	45-15
Preserve Access Control Check Box	45-15
Configuring the Oracle WebLogic Security Framework: Main Steps	45-15
Context Properties Are Passed to Security Providers	45-19
Context Properties for HTTP Transport-Level Authentication.....	45-19
ContextHandler Properties for Access Control and Message-Level Custom Authentication	45-20
Additional Transport-Specific Context Properties.....	45-20
Administrator-Supplied Context Properties for Message-Level Authentication.....	45-21
Security Provider Must Have Knowledge of the Property Name	45-21
Oracle WebLogic Server Administrative Channel is Supported	45-22
Using Security Providers	45-23
Configuring Authentication Providers	45-23
Using a Custom Authorization Provider to Protect Oracle Service Bus Resources	45-24
WebLogic Authorization Provider Usage Information.....	45-24
Oracle Service BusProxyServiceResource Object	45-25
ALSBProxyServiceResource Examples	45-26
ProjectResourceV2 Object	45-26
ConsoleResource Object.....	45-27

46 Oracle Service Bus Security FAQ

How are Oracle Service Bus and Oracle WebLogic Server Security related?	46-1
What is Transport-Level Security?	46-2
What is Web Services Security?	46-2
What is Web Service Policy?	46-2
What are Web Service Policy assertions?	46-2
Are Access Control Policy and Web Service Policy the same?	46-2
What is Web Services Security Pass-Through?.....	46-3
What is a Web Services Security Active Intermediary?	46-3
What is outbound Web Services Security?	46-3
What is SAML?	46-3
Is it possible to customize the format of the subject identity in a SAML assertion?	46-3
What is the Certificate Lookup And Validation Framework?	46-3
Does Oracle Service Bus support identity propagation in a proxy service?	46-4
If both transport-level authentication and message-level authentication exist on inbound messages to the proxy service, which identity is propagated?	46-4
Is single sign-on supported in Oracle Service Bus?	46-4

Are security errors monitored?	46-4
Can I configure security for MBeans?	46-5
47 Configuring Administrative Security	
Administrative Security Roles and Privileges.....	47-1
Role-Based Access in the Oracle Service Bus Console.....	47-2
Administrative Security Groups	47-7
Configuring Administrative Security: Main Steps.....	47-8
48 Securing Oracle Service Bus in a Production Environment	
Undeploying the Service Bus (SB) Resource	48-1
Protection of Temporary Files With Streaming body Content	48-2
Protecting Against Denial of Service Attacks on the Oracle Service Bus Console	48-2
49 Configuring Transport-Level Security	
Configuring Transport-Level Security for HTTPS	49-2
HTTPS Authentication Levels	49-2
Configuring Inbound HTTPS Security: Main Steps.....	49-3
Configuring Outbound HTTPS Security: Main Steps.....	49-4
Configuring Transport-Level Security for HTTP	49-4
Configuring Inbound HTTP Security: Main Steps.....	49-5
Configuring Outbound HTTP Security: Main Steps.....	49-5
Configuring Transport-Level Security for JMS	49-6
Configuring Inbound JMS Transport-Level Security: Main Steps.....	49-6
Configuring Outbound JMS Transport-Level Security: Main Steps.....	49-7
Configuring Transport-Level Security for SFTP Transport.....	49-8
How Two-Way Authentication is Performed	49-8
Use of the known_hosts File.....	49-8
SFTP Transport Authentication Process	49-9
Configuring Inbound SFTP Transport-Level Security: Main Steps.....	49-10
Configuring Outbound SFTP Transport-Level Security: Main Steps.....	49-12
SFTP Security Attributes Preserved During Import	49-14
SFTP Credential Life Cycle	49-14
Email, FTP, and File Transport-Level Security	49-14
Email and FTP Transport-Level Security	49-14
File Transport Security	49-14
Configuring Transport-Level Security for SB Transport	49-15
Configuring SAML Authentication With Service Bus (SB) Transport	49-15
Configuring Transport-Level Security for WS Transport.....	49-16
Reliable Web Services Messaging Defined.....	49-16
WS Transport Resources Visible in WLS Console.....	49-16
Use of WS-Policy Files for Web Service Reliable Messaging Configuration	49-16
Preconfigured WS-RM Policy Files	49-17
RM WS-Policy Required Prior to Activation.....	49-17
Async Responses	49-17
Proxy Service Authentication	49-18

Preserving Security Configuration on Import	49-19
Configuring Inbound and Outbound WS Transport-Level Security	49-19
Configuring Transport-Level Security for WebSphere Message Queue Transport	49-19
Configuring Inbound MQ Transport-Level Security: Main Steps	49-20
Configuring Outbound MQ Transport-Level Security: Main Steps	49-20
Transport-Level Security Elements in the Message Context	49-21

50 Securing Oracle Service Bus with Oracle Web Services Manager

About Oracle Web Services Manager Integration with Oracle Service Bus	50-1
Security Providers	50-2
JPS Providers	50-2
CSS Providers	50-2
Setting Up and Using Oracle Web Services Manager with Oracle Service Bus	50-3
Adding Oracle Web Services Manager and Oracle Enterprise Manager to an Oracle Service Bus Domain 50-3	
Attaching Oracle Web Services Manager Policies to Oracle Service Bus Services	50-4
Policy Overrides	50-5
Configuring SAML	50-5
Deployment Considerations	50-5
Auditing	50-6
Monitoring Statistics	50-6
Unsupported Assertions and Seed Policies	50-6
Use Cases: Oracle Service Bus and WLS 9.2 Policies with Oracle Web Services Manager	50-7
Message Protection	50-8
Message Protection with Client Agent	50-8
Message Protection with Server Agent	50-8
Message Protection with Client and Server Agents	50-9
Message Protection with Gateway	50-9
Authentication	50-10
Perimeter Security	50-10
Identity Propagation	50-11

51 Using WS-Policy in Oracle Service Bus Proxy and Business Services

About Web Services Policy	51-1
Relationship Between WS-Security and WS-Policy	51-2
WS-Policies Can be Bound Directly to Service	51-2
Abstract and Concrete WS-Policy Statements	51-3
Oracle Service Bus WS-Policy Files	51-3
Predefined Oracle Proprietary Policy Files	51-4
Predefined Reliable Messaging Policy Files	51-5
When to use the Predefined Policy Files	51-5
Creating and Using Custom WS-Policy Statements	51-5
Attaching WS-Policy Statements to WSDL Documents	51-6
Determining the URI of a WS-Policy Statement	51-6
Specifying the URI of a WS-Policy Statement in a WSDL Document	51-7
Best Practices: Attaching WS-Policy Statements	51-8
Example: Requiring X.509 Credentials for Identity and Confidentiality	51-8

Example: Attaching Custom Inline WS-Policy Statements to a WSDL Document.....	51-9
Oracle-Proprietary Security Policy Best Practices.....	51-10
Policy Subjects and Effective Policy.....	51-11

52 Configuring Message-Level Security for Web Services

About Message-Level Security	52-2
Sample Sequence of Actions in Message-Level Security.....	52-2
Message-Level Access Control Policies for Proxy Services	52-3
Configuring Proxy Service Message-Level Security	52-3
Creating an Active Intermediary Proxy Service: Main Steps	52-3
Creating a Pass-Through Proxy Service: Main Steps.....	52-5
Configuring Business Service Message-Level Security: Main Steps.....	52-6
Examples of Custom WS-Policy Statements	52-7
Example: Encrypting Part of the SOAP Body and Header	52-7
Example: Encryption Policy for a Business Service	52-9
Example: Encrypting a Custom SOAP Header	52-11
Example: Signing the Message Body and Headers.....	52-11
Example: Signing a SOAP Body with SAML Holder-of-Key	52-12
Example: Authenticating, Signing, and Encrypting a SOAP Body and Headers with SAML Sender Vouches	52-14
Disabling Business Service Message-Level Security	52-16

53 Using SAML for Authentication

Configuring SAML Credential Mapping: Main Steps	53-1
Configuring SAML Pass-Through Identity Propagation	53-2
Authenticating SAML Tokens in Proxy Service Requests	53-3
Configuring SAML Authentication with Service Bus (SB) Transport.....	53-4
Troubleshooting SAML Web Services Security	53-4

54 Configuring Custom Authentication

What Are Custom Authentication Tokens?.....	54-2
Custom Authentication Token Use and Deployment	54-2
Understanding Transport-Level Custom Authentication.....	54-3
Importing and Exporting and Transport-Level Custom Token Authentication	54-3
Understanding Message-Level Custom Authentication	54-4
Format of XPath Expressions.....	54-4
Configuring Identity Assertion Providers for Custom Tokens.....	54-5
Object Type of Custom Tokens	54-6
Configuring a Custom Token Type in an Identity Assertion Provider	54-6
Steps for Configuring a Custom Token Type in an Identity Assertion Provider.....	54-7
Setting the Supported and Active Types in the MBean	54-7
Configuring Custom Authentication Transport-Level Security	54-8
Steps for Configuring Custom Authentication Transport-Level Security.....	54-8
Configuring Custom Authentication Message-Level Security	54-9
Steps for Configuring Custom Authentication Message-Level Security	54-9
Propagating the Identity Obtained From Custom Authentication Tokens	54-10

Combining WS-Security with Custom Username/Password and Tokens	54-10
--	-------

55 Message-Level Security with .Net 2.0

Message-Level Security Between .NET 2.0 and Oracle Service Bus	55-1
What is .NET?	55-1
Message-Level Security Configuration in .NET For Oracle Service Bus Interoperability	55-1
Oracle Service Bus Configuration for Message-Level Security with .NET	55-3
Sample WSDL File	55-5

Part VIII Appendix

A Transport SDK UML Sequence Diagrams

Oracle Service Bus Runtime Inbound Messages	A-1
Oracle Service Bus Runtime Outbound Messages	A-2
Design Time Service Registration.....	A-3

List of Examples

6-1	XML Data with Optional Element.....	6-17
6-2	XML Schema with Global and Local Elements	6-24
7-1	calculateTotalPrice Function	7-7
7-2	Where Clause.....	7-10
7-3	If-Then Expression.....	7-19
7-4	Nested If-Then-Else Expression.....	7-20
7-5	Nested If-Then-Else Expression.....	7-21
7-6	Code for FLWOR Expression	7-23
7-7	Example of Recursive Schema	7-23
7-8	Example Input XML Document.....	7-25
7-9	Example Output XML Document.....	7-25
7-10	XQuery Code for Group-By Expression.....	7-26
23-1	Syntax for Invoking a Custom Function with the Xalan Engine	23-4
30-1	Sample WSDL Definitions	30-4
30-2	Sample SOAP	30-5
39-1	Sample SocketEndPointConfiguration Definition	39-5
39-2	Sample SocketRequestMetaDataXML Definition	39-6
39-3	Sample SocketRequestHeadersXML Definition.....	39-6
39-4	Sample SocketResponseMetaDataXML Definition.....	39-7
39-5	Sample SocketResponseHeadersXML Definition	39-7
39-6	Example tModel	39-21
39-7	Sample transport plugin.xml.....	39-28
39-8	Sample Socket Transport toc.xml for a Top-Level Entry in the Eclipse Help System .	39-29
39-9	Sample context.xml (contexts_socketTransport.xml) for the sample socket transport	39-29
39-10	Implementing CustomHelpProvider to provide help for your transport in the Oracle Service Bus console	39-33
39-11	JavaScript function that provides a redirect	39-34
39-12	application.xml for the sample socket transport Web application.....	39-34
39-13	web.xml for the sample socket transport Web application	39-35
40-1	Connection to the Remote Server	40-8
40-2	The TransportProviderFactory Class.....	40-9
40-3	Example of the Socket Transport Implementing the Interface.....	40-9
40-4	Part of the Extension Point Schema.....	40-10
40-5	Plugin.xml File.....	40-10
40-6	Sample MANIFEST.MF File	40-11
40-7	TransportManagerHelper Methods	40-11
43-1	Application Deployment Entry.....	43-2
45-1	Getting the HttpServletRequestProperty	45-21
49-1	LongRunningReliability.xml File	49-17
49-2	Wssp1.2-Https.xml File (Partial).....	49-18
51-1	WSDL with Policy References to Oracle Service Bus WS-Policies	51-9
51-2	WSDL with Policy References to a Custom Inline Policy	51-9
52-1	Encrypting Part of the SOAP Body and Header	52-8
52-2	Encrypting the Body with a Concrete Policy, Embedding the Policy in the WSDL Document	52-9
52-3	Encrypting a Custom SOAP Header.....	52-11
52-4	Requiring a Signature for SOAP Headers and Body.....	52-12
52-5	Signing a SOAP Body with SAML Holder-of-Key Method	52-13
52-6	Signing a SOAP Body and Headers with SAML Sender-Vouches Method.....	52-15
54-1	SampleIdentityAsserter MDF: SupportedTypes Attribute	54-7
54-2	SampleIdentityAsserter MDF: ActiveTypes Attribute with Default	54-8
55-1	Configuring WS-Policy for Message-Level Security	55-5

List of Figures

2-1	New Split-Join	2-38
2-2	Parallel Node	2-39
2-3	Completed Split-Join Ready for Testing.....	2-40
2-4	New Split-Join With Edited Labels	2-41
2-5	For Each Node Labeled "Iterate Through Orders".....	2-42
2-6	Error Handler	2-43
2-7	Completed Split-Join Ready for Testing.....	2-44
2-8	Add Scope Button	2-51
2-9	Add Else If Button.....	2-53
2-10	Add Else If Button.....	2-54
2-11	Debugging a Proxy Service	2-63
4-1	Branch Node With "Submit" Operation From WSDL and "Default" Branch	4-55
4-2	Publish Table Action	4-57
4-3	Routing Table Action.....	4-66
5-1	Data Transformation from Multiple Sources to One Target	5-2
6-1	New XQuery Transformation	6-3
6-2	Selecting Source Types.....	6-6
6-3	Element-to-Element Links	6-8
6-4	Attribute-to-Element Link	6-9
6-5	Repeating-Group-to-Repeating-Group Link.....	6-9
6-6	Example of If-Then-Else Expression	6-12
6-7	If-Then-Else Expression in Target Expression View.....	6-12
6-8	FLWOR Expression.....	6-13
6-9	Typeswitch Expression	6-15
6-10	Remove Empty Node	6-18
6-11	Retain Empty Node	6-18
6-12	Test View	6-19
7-1	Combining Data From Different Schemas	7-2
7-2	Design View of XQuery Transformation.....	7-3
7-3	Data Transformation in Design View	7-4
7-4	Data Transformation in Design View	7-5
7-5	Adding a Constraint	7-6
7-6	Adding Data in the QuoteResponse Element.....	7-7
7-7	totalCost Calculation in Design View	7-9
7-8	Creating a Union	7-11
7-9	Creating Implied Links	7-12
7-10	Repeating-Source-to-Nonrepeating Target Transformation.....	7-13
7-11	Repeating-Source-to-Nonrepeating-Target Data Transformation	7-13
7-12	Repeating-Source-to-Nonrepeating-Target Data Transformation	7-14
7-13	Nonrepeating-Source-to-Repeating-Target Transformation.....	7-15
7-14	Creating a Union for Structural Links	7-16
7-15	XQuery Transformation for If-Then-Else Example.....	7-19
7-16	XQuery Example - FLWOR Expression.....	7-22
7-17	Mapping Recursive Elements	7-24
8-1	Source Upgrade Screen of the Upgrade Wizard	8-1
8-2	Upgrade XQ2002 to XQ2004.....	8-2
20-1	Select Reference Fields	20-6
20-2	Reference Field	20-6
20-3	Data Delimiter	20-7
20-4	Escape Character	20-8
24-1	Client Invoking a SOA Binding Service Synchronously	24-4
24-2	SOA Binding Service Invoking an External Service Synchronously.....	24-5
24-3	Client Invoking a SOA Binding Service Asynchronously	24-6
24-4	SOA Binding Service Invoking an External Service Asynchronously	24-9

24-5	Operations in a Synchronous Exchange Through Oracle Service Bus.....	24-15
24-6	SOA Composite Invoking an SOA Composite Through Oracle Service Bus.....	24-20
25-1	Oracle Service Bus Services Interacting With an EIS	25-1
25-2	A Client Invoking an EIS Service Through Oracle Service Bus	25-10
25-3	An EIS Invoking an External Service Through Oracle Service Bus	25-11
29-1	An EJB Invokes an External Service	29-6
29-2	A Non-EJB Client Invokes an EJB.....	29-7
29-3	An EJB Invokes an EJB	29-8
30-1	MQ Service Uses a JMS Message ID to Correlate the Request/Response Message	30-12
30-2	JAX-RPC Client with Oracle Service Bus Proxy Service	30-13
30-3	Oracle Service Bus as a Client of an Oracle WebLogic Server JAX-RPC Request/Response Service	30-13
31-1	Using Local Transport to Implement Convergence	31-3
31-2	Using Local Transport to Access Multiple Business Services	31-4
32-1	Conflicts – When no RM policy assertions are specified for the WSDL	32-5
32-2	Security tab for a Proxy Service	32-8
32-3	Transport-Level Policy	32-9
32-4	List of Predicates	32-10
33-1	Oracle Service Bus Front End.....	33-19
33-2	Messages Sent Through Oracle Service Bus.....	33-19
34-1	Invoking Oracle BPEL Processes Synchronously Through Oracle Service Bus	34-3
34-2	Oracle BPEL Processes Invoking a Service Synchronously Through Oracle Service Bus.....	34-4
34-3	Invoking Oracle BPEL Processes Asynchronously Through Oracle Service Bus.....	34-5
34-4	Oracle BPEL Processes Invoking a Service Asynchronously Through Oracle Service Bus.....	34-6
34-5	Operations in a Synchronous Exchange Through Oracle Service Bus.....	34-13
34-6	One BPEL Process Invoking Another BPEL Process Through Oracle Service Bus.....	34-18
35-1	WTC Message Handling	35-2
36-1	From Data Source to Web Service Client.....	36-2
36-2	Request and Response from the Oracle Service Bus Test Console	36-5
38-1	Message Flow Through Oracle Service Bus	38-2
38-2	Design Time UML Diagram	38-7
38-3	EndPointConfiguration Properties.....	38-8
38-4	Runtime UML Diagram	38-9
38-5	Sample Oracle Service Bus Threading Model.....	38-16
38-6	Flow of Attachments	38-20
38-7	Transformation Matrix.....	38-21
39-1	Transport Subsystem Overview	39-3
39-2	Error Case 1.....	39-16
39-3	Error Case 2.....	39-17
39-4	Error Case 3.....	39-18
39-5	Custom Transport Help in the Development and Run-Time Environments	39-23
39-6	Pressing F1 on a Transport Configuration Page to Display Help for the Transport ...	39-24
39-7	Custom Transport Help in Eclipse	39-25
39-8	Custom Transport Help from the Oracle Service Bus Console.....	39-26
39-9	Oracle Service Bus Console Help Framework.....	39-32
40-1	Transport Configuration Page in Eclipse	40-6
40-2	Plug-In Packaging	40-7
42-1	Sample Socket Transport Architecture	42-2
42-2	SocketEndpointConfiguration Properties	42-3
42-3	Request/Response Header and Metadata Configurations	42-3
42-4	Selecting the Message Flow Icon	42-8
42-5	Editing the Message Flow	42-8
42-6	Adding an Action.....	42-8

42-7	Starting the Test Console	42-9
42-8	Test Console.....	42-9
42-9	Successful Test.....	42-10
45-1	Inbound and Outbound Security.....	45-2
45-2	How Service Accounts Are Used.....	45-10
50-1	Message Protection With an Oracle Web Services Manager Client Agent	50-8
50-2	Message Protection With an Oracle Web Services Manager Server Agent.....	50-9
50-3	Message Protection With an Oracle Web Services Manager Client and Server Agents	50-9
50-4	Message Protection With an Oracle Web Services Manager Gateway	50-10
50-5	Authentication with an Oracle Web Services Manager Client Agent.....	50-10
50-6	Perimeter Security with Oracle Web Services Manager Gateway	50-11
50-7	Identity Propagation with Oracle Web Services Manager Gateway.....	50-11
52-1	Binding a Certificate to an Abstract Policy	52-8
54-1	Identity Assertion and Custom Tokens.....	54-6
A-1	Inbound Messages at Runtime.....	A-2
A-2	Outbound Messages at Runtime	A-3
A-3	Service Registration	A-4

List of Tables

2-1	Differences in Harvesting Scenarios	2-30
3-1	BPEL-10g Transport Configuration Options for Business Services.....	3-2
3-2	DSP Transport Configuration Options for Business Services	3-3
3-3	EJB Transport Configuration Options for Business Services.....	3-4
3-4	E-Mail Transport Configuration Options for Business Services	3-6
3-5	E-Mail Transport Configuration Options for Proxy Services.....	3-6
3-6	File Transport Configuration Options for Business Services	3-7
3-7	File Transport Configuration Options for Proxy Services	3-7
3-8	FTP Transport Configuration Options for Business Services.....	3-8
3-9	FTP Transport Configuration Options for Proxy Services.....	3-9
3-10	HTTP Transport Configuration Options for Business Services	3-10
3-11	HTTP Transport Configuration Options for Proxy Services	3-13
3-12	JCA Transport Configuration Options	3-14
3-13	JEJB Transport Configuration for Business Services	3-16
3-14	JEJB Transport Configuration for Proxy Services	3-17
3-15	JMS Transport Configuration Options for Business Services.....	3-18
3-16	JMS Transport Configuration Options for Proxy Services	3-22
3-17	MQ Transport Configuration Options for Business Services.....	3-24
3-18	MQ Transport Configuration Options for Proxy Services.....	3-27
3-19	SB Transport Configuration Options for Business Services	3-29
3-20	SB Transport Configuration Options for Proxy Services	3-29
3-21	SFTP Transport Configuration Options for Business Services.....	3-30
3-22	SFTP Transport Configuration Options for Proxy Services.....	3-30
3-23	SOA-DIRECT Transport Configuration	3-32
3-24	Tuxedo Transport Configuration Options for Business Services	3-33
3-25	Tuxedo Transport Configuration Options for Proxy Services	3-35
3-26	WS Transport Configuration Options for Business Services.....	3-37
3-27	WS Transport Configuration Options for Proxy Services.....	3-37
4-1	Alert Destination Editor Options.....	4-2
4-2	E-mail Recipient Options	4-2
4-3	JMS Destination Options.....	4-3
4-4	Business Service Editor Options	4-4
4-5	Message Type Business Service Options	4-5
4-6	Business Service Transport Configuration Options.....	4-6
4-7	Business Service Message Handling Options	4-10
4-8	Business Service Service Policy Configuration Options	4-13
4-9	Business Service Security Configuration Options.....	4-13
4-10	Proxy Service Editor Options	4-14
4-11	Proxy Service Message Type Options.....	4-16
4-12	Proxy Service Operation Selection Options	4-17
4-13	Proxy Service Message Handling Options.....	4-19
4-14	Proxy Service SOAP Binding Options	4-23
4-15	Proxy Service Transport Configuration Options	4-23
4-16	Proxy Service Service Policy Options	4-26
4-17	Proxy Service Message-Level Security Options	4-27
4-18	Oracle Service Bus Configuration Options	4-29
4-19	New Oracle Service Bus Project Options.....	4-30
4-20	MQ Resource Options	4-31
4-21	Configuration JAR Export Options	4-33
4-22	Resource Export Options	4-33
4-23	Resource Export Options	4-34
4-24	Configuration JAR Import	4-34
4-25	Configuration JAR Import Options.....	4-35

4-26	Configuration ZIP Import.....	4-35
4-27	Configuration ZIP Import Options	4-35
4-28	URL Import Options.....	4-36
4-29	URL Import Options.....	4-36
4-30	New JNDI Provider Resource Options.....	4-37
4-31	Proxy Server Configuration Options	4-37
4-32	Message Flow Nodes.....	4-39
4-33	Message Flow Route Node Communication Actions.....	4-39
4-34	Message Flow Route Node Flow Control Actions.....	4-40
4-35	Message Flow Stage Node Communication Actions.....	4-40
4-36	Message Flow Stage Node Flow Control Actions.....	4-41
4-37	Message Flow Stage Node Message Processing Actions	4-41
4-38	Message Flow Stage Node Reporting Actions.....	4-42
4-39	Alert Action Options	4-44
4-40	Assign Action Options	4-45
4-41	Flow Options	4-45
4-42	Conditional Branch Options.....	4-46
4-43	Branch Options.....	4-46
4-44	Delete Action Options	4-46
4-45	Dynamic Publish Action Options	4-47
4-46	Dynamic Routing Action Properties	4-48
4-47	For-Each Action Properties.....	4-49
4-48	If and Else-If Action Options.....	4-50
4-49	Insert Action Options	4-51
4-50	Java Callout Action Options.....	4-52
4-51	Log Action Options.....	4-53
4-52	MFL Transform Action Options	4-54
4-53	Pipeline Pair Node Options.....	4-56
4-54	Publish Action Options	4-56
4-55	Publish Table Options	4-58
4-56	Case Action Options.....	4-58
4-57	Raise Error Action Options.....	4-58
4-58	Rename Action Options	4-59
4-59	Replace Action Options.....	4-60
4-60	Reply Action Options.....	4-61
4-61	Report Action Options	4-61
4-62	Report Action Execution Results	4-62
4-63	Route Node Options.....	4-63
4-64	Routing Action Options	4-63
4-65	Routing Options Action Properties	4-64
4-66	Routing Table Options	4-66
4-67	Case Action Options.....	4-67
4-68	Service Callout Action Options.....	4-67
4-69	SOAP Body, Payload Parameters, and Payload Document Options	4-68
4-70	SOAP Body, Payload Parameters, and Payload Document Option Descriptions	4-68
4-71	Start Node Options	4-70
4-72	Transport Header Action Options.....	4-71
4-73	Validate Action Options.....	4-73
4-74	Modify JAR Dependencies Properties	4-73
4-75	New SMTP Server Options.....	4-74
4-76	UDDI Registry Options.....	4-74
4-77	Type Association Properties.....	4-75
4-78	Reference Dependency Properties.....	4-76
4-79	Clone Project Options.....	4-76
4-80	New Service Key Provider Options	4-76

4-81	New WS-Policy Options	4-77
4-82	New Service Account Options	4-78
4-83	Service Account General Options	4-78
4-84	Service Account Static User Options	4-79
4-85	Service Account User Mappings Options	4-79
4-86	XQuery Resource Options	4-82
4-87	XSLT Resource Options	4-82
4-88	Dynamic XQuery Options	4-83
4-89	Variable Structure Options	4-84
4-90	New XSL Transformation Options	4-85
4-91	Split-Join Communication Operations	4-86
4-92	Split-Join Flow Control Operations	4-86
4-93	Split-Join Assign Operations	4-87
4-94	Split-Join Start Node Options - Imports Tab	4-88
4-95	Split-Join Start Node Options - General Tab	4-88
4-96	Split-Join Variable Options	4-88
4-97	Split-Join Error Handler Options - Catch Tab	4-89
4-98	Split-Join Error Handler Options - CatchAll Tab	4-89
4-99	Split-Join Invoke Service Options - Operation Tab	4-89
4-100	Split-Join Invoke Service Options - Input Variable Tab	4-90
4-101	Split-Join Invoke Service Options - Output Variable Tab	4-90
4-102	Split-Join Reply Options - Operation Tab	4-91
4-103	Split-Join Reply Options - Variable Tab	4-91
4-104	Split-Join For Each Options - Counter Variables Tab	4-92
4-105	Split-Join For Each Options - Completion Condition	4-92
4-106	Split-Join If and Else If Options	4-93
4-107	Split-Join Raise Error Options	4-93
4-108	Split-Join Repeat Until Options	4-93
4-109	Split-Join While Options	4-94
4-110	Split-Join Wait Options	4-94
4-111	Split-Join Assign Operation Options	4-95
4-112	Split-Join Copy Options	4-96
4-113	Split-Join Delete Options	4-97
4-114	Split-Join Insert Options	4-97
4-115	Split-Join Java Callout Options	4-98
4-116	Split-Join Log Options	4-99
4-117	Split-Join Replace Options	4-99
4-118	Split-Join Receive Options - Operation Tab	4-100
4-119	Split-Join Receive Options - Variable Tab	4-100
4-120	Split-Join Counter Variable Options	4-100
4-121	Split-Join Variable Options	4-101
4-122	Split-Join New Message Variable Options	4-101
4-123	Split-Join SOAP Fault Variable Options	4-102
4-124	New Split-Join Options	4-102
4-125	New Split-Join Options - Selecting an Operation	4-102
5-1	Restrictions Applicable to the XQuery Test View	5-2
6-1	Clauses of FLWOR Expressions	6-13
6-2	Clauses of Typeswitch Expressions	6-15
6-3	Right-Click Menu Options	6-21
6-4	Link Patterns	6-22
6-5	Graphical Representation of XML Components	6-25
8-1	Constructs For Which the Syntax Has Changed in XQuery 2004	8-3
8-2	Functions For Which Parameter Types and Order Have Changed in XQuery 2004	8-3
10-1	List of Toolbar Buttons	10-1
10-2	Tree Icon Descriptions	10-3

11-1	Message Format	11-1
12-1	Fields in Message Format – Field Description.....	12-1
12-2	Fields in Message Format – Field Occurrence	12-1
12-3	Fields in Message Format – Field Data Options.....	12-2
12-4	Fields in Message Format – Field Attributes	12-2
12-5	Fields in Message Format – Termination	12-2
12-6	Fields in Message Format – Literal.....	12-3
12-7	Fields in Message Format – Field Update Buttons.....	12-4
13-1	Groups in Message Format – Group Description	13-1
13-2	Groups in Message Format – Group Occurrence.....	13-1
13-3	Groups in Message Format – Group Attributes	13-1
13-4	Groups in Message Format – Group Delimiter	13-2
13-5	Groups in Message Format – Group Update Buttons	13-2
14-1	Reference Detail Window – Reference Description	14-1
14-2	Reference Detail Window – Field Occurrence.....	14-1
14-3	Reference Detail Window – Field Update Buttons	14-1
15-1	Comments about Message Format	15-1
16-1	Format Builder Options	16-1
16-2	Format Builder Options – Character Encoding Options.....	16-1
16-3	Format Builder Options – XML Formatting Options	16-1
16-4	Format Builder Options – XML Content Model Options	16-1
17-1	EDI Importer Options.....	17-1
17-2	XML Schema Importer Options.....	17-2
17-3	COBOL Copybook Importer Options.....	17-2
17-4	COBOL Copybook Importer Options – Byte Order	17-2
17-5	COBOL Copybook Importer Options – Character Set	17-3
17-6	COBOL Copybook Importer Options – Action Buttons	17-3
17-7	C Structure Importer Options – Input	17-3
17-8	C Structure Importer Options – Output	17-4
17-9	FML Field Table Class Sample Files.....	17-6
17-10	FML Field Table Class Importer Options.....	17-6
17-11	FML Field Table Class Importer Options – FML Field Selector.....	17-7
17-12	FML Field Table Class Importer Options – Action Buttons	17-7
18-1	File Menu Commands.....	18-2
18-2	Edit Menu Commands	18-2
18-3	Display Menu Commands.....	18-2
18-4	Generate Menu Commands	18-3
18-5	Translate Menu Command.....	18-3
18-6	Shortcut Menu Commands	18-3
18-7	Find Options	18-5
18-8	Goto Options.....	18-6
19-1	File Menu Commands.....	19-1
19-2	Edit Menu Commands	19-2
19-3	Insert Menu Commands	19-2
19-4	View Menu Commands	19-3
19-5	Tools Menu Commands.....	19-3
19-6	Help Menu Commands.....	19-3
19-7	Shortcut Menu Commands	19-4
20-1	Character Delimiters.....	20-5
21-1	File Menu Commands.....	21-2
21-2	Shortcut Menu Commands	21-2
22-1	Supported MFL Data Types	22-1
22-2	COBOL Data Types	22-5
23-1	Supported Java Method Types for Custom Functions.....	23-3
24-1	SOA-DIRECT Transport Configuration	24-12

25-1	JCA Transport Configuration Options	25-13
26-1	Parameters for Configuring HTTP Transport for Proxy Service	26-2
26-2	Parameters for Configuring HTTP Transport for Business Service	26-4
26-3	Response codes for HTTP business services.....	26-8
26-4	Parameters for Configuring E-mail Transport for Proxy Services.....	26-9
26-5	Parameters for Configuring E-mail Transport for Business Services.....	26-10
26-6	Parameters for Configuring File Transport for Proxy Services.....	26-11
26-7	Parameters for Configuring File Transport for Business Services.....	26-12
26-8	Parameters for Configuring FTP Transport for Business Services	26-13
26-9	Parameters for Configuring FTP Transport for Business Service	26-15
26-10	Environment Values	26-16
26-11	Configuring SFTP Proxy Service	26-21
26-12	Transport Headers and Metadata.....	26-23
26-13	Configuring SFTP Business Service	26-25
26-14	Properties Imported from UDDI Registry.....	26-27
27-1	Environment Values	27-2
27-2	Fields for Configuring SB Transport for Proxy Services	27-3
27-3	Fields for Configuring SB Transport for Business Services	27-4
28-1	EJB Transport Configuration Options for Business Services.....	28-4
29-1	JEJB Transport Configuration for Proxy Services	29-9
29-2	JEJB Transport Configuration for Business Services	29-11
30-1	Differences Between Message ID and Correlation ID Patterns.....	30-9
30-2	JMS Transport Headers	30-15
32-1	Fields Required to Configure a Proxy Service to Use the WS Transport	32-7
32-2	Fields Required to Configure WS Transport for a Proxy Service	32-7
32-3	Configuring a Business Service to use WS Transport	32-15
32-4	Configuring WS Transport for Business Service.....	32-16
33-1	MQ Proxy Service Configuration	33-7
33-2	MQ Business Service Configuration	33-9
33-3	Transport Headers	33-12
34-1	BPEL transport environment variables.....	34-2
34-2	Specifying an Endpoint URI.....	34-8
34-3	BPEL transport configuration	34-9
35-1	Tuxedo Exceptions.....	35-9
35-2	Buffer Transformation for Any XML Service	35-14
35-3	Buffer Transformation for Messaging Service	35-15
39-1	Context IDs for Transport Configuration Pages	39-30
42-1	Key Sample Transport Provider Directories.....	42-4
45-1	Options for Identity Propagation	45-4
45-2	Combinations of Transport-Level Security Requirements	45-5
45-3	Combinations of Message-Level Security Requirements.....	45-7
45-4	Authentication Providers.....	45-16
45-5	ContextHandler Properties for HTTP Transport Authentication.....	45-19
45-6	ContextHandler Properties for Message-Level Custom Authentication and Access Control.....	45-20
45-7	Additional Message-Level Security ContextHandler Properties for HTTP Proxy Services.....	45-21
47-1	Oracle Service Bus Administrative Security Roles.....	47-1
47-2	Oracle WebLogic Server Security Roles.....	47-2
47-3	Role-Based Operations Access in Oracle Service Bus Console	47-3
47-4	Role-Based Resource Browser Access in Oracle Service Bus Console	47-3
47-5	Role-Based Project Explorer Access in Oracle Service Bus Console.....	47-5
47-6	Role-Based Security Configuration Access in Oracle Service Bus Console.....	47-5
47-7	Role-Based System Administration Access in Oracle Service Bus Console	47-6
47-8	Role-Based Change Center Access in Oracle Service Bus Console.....	47-7

47-9	Oracle Service Bus Groups	47-8
49-1	WS Transport Authentication Matrix	49-18
50-1	Unsupported Oracle Web Services Manager Assertions	50-6
50-2	Unsupported Oracle Web Services Manager Seed Policies.....	50-7
51-1	WSDL Elements That Can Be Protected in Oracle Service Bus.....	51-7

Preface

This preface describes the document accessibility features and conventions of this guide--*Oracle Fusion Middleware Developer's Guide for Oracle Service Bus*.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at <http://www.oracle.com/accessibility/>.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/support/contact.html> or visit <http://www.oracle.com/accessibility/support.html> if you are hearing impaired.

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.

Convention	Meaning
<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.

Part I

IDE Help for Oracle Service Bus

This part contains the Oracle Service Bus IDE help, which includes the following chapters:

- [Chapter 1, "Introduction to Oracle Service Bus"](#)
- [Chapter 2, "Tasks"](#)
- [Chapter 3, "Transport Configuration"](#)
- [Chapter 4, "User Interface Reference"](#)

Introduction to Oracle Service Bus

Oracle Service Bus is a configuration-based, policy-driven Enterprise Service Bus (ESB). It provides highly scalable and reliable service-oriented integration, service management, and traditional message brokering across heterogeneous IT environments. It combines intelligent message brokering with routing and transformation of messages, along with service monitoring and administration in a unified software product. For more information, see the *Oracle Fusion Middleware Concepts and Architecture for Oracle Service Bus* at

<http://www.oracle.com/pls/as1111/lookup?id=OSBCA>.

This documentation tells how to use the Oracle Service Bus plug-ins for Eclipse to configure services and policies. These are design-time activities, so you do not need a running Oracle WebLogic Server instance. The Web-based Oracle Service Bus Console, also provided with Oracle Service Bus, provides the same design tools as the Oracle Service Bus plug-ins, and it also provides run-time activities such as monitoring. You need a running Oracle WebLogic Server instance to use the Oracle Service Bus Console.



This section tells how to perform tasks in the Oracle Service Bus plug-ins for Eclipse:

- [Section 2.1, "Working with Projects, Folders, Resources, and Configurations"](#)
- [Section 2.2, "Working with Business Services"](#)
- [Section 2.3, "Working with Proxy Services"](#)
- [Section 2.4, "Working with Proxy Service Message Flows"](#)
- [Section 2.5, "Working with Alert Destinations"](#)
- [Section 2.6, "Working with MQ Connections"](#)
- [Section 2.7, "Working with Oracle Enterprise Repository and Harvester."](#)
- [Section 2.8, "Working with UDDI Registries"](#)
- [Section 2.9, "Working with Split-Join"](#)
- [Section 2.10, "Using the Oracle Service Bus Debugger"](#)

2.1 Working with Projects, Folders, Resources, and Configurations

This section tells how to perform the following tasks:

- [Section 2.1.1, "Editing Resources"](#)
- [Section 2.1.2, "Cloning Oracle Service Bus Projects and Folders"](#)
- [Section 2.1.3, "Creating Oracle Service Bus Configuration Projects"](#)
- [Section 2.1.4, "Creating Oracle Service Bus Projects"](#)
- [Section 2.1.5, "Creating Custom Resources"](#)
- [Section 2.1.6, "Creating JNDI Provider Resources"](#)
- [Section 2.1.8, "Creating Message Format Files"](#)
- [Section 2.1.9, "Editing JNDI Provider Resources"](#)
- [Section 2.1.9, "Editing JNDI Provider Resources"](#)
- [Section 2.1.11, "Exporting Resources"](#)
- [Section 2.1.12, "Generating an Effective WSDL"](#)
- [Section 2.1.13, "Modifying JAR Dependencies"](#)
- [Section 2.1.14, "Importing Resources"](#)
- [Section 2.1.15, "Creating Servers"](#)

- [Section 2.1.16, "Creating Service Account Resources"](#)
- [Section 2.1.17, "Creating Service Key Provider Resources"](#)
- [Section 2.1.18, "Creating SMTP Server Resources"](#)
- [Section 2.1.19, "Creating XQuery Transformations"](#)
- [Section 2.1.20, "Creating XSL Transformations"](#)

2.1.1 Editing Resources

Edit resources using the built-in editors. For example, edit a proxy service by double-clicking its name in the Project Explorer.

Do not manually edit resource files as text or XML files. This can result in unpredictable behavior. Do not manually edit these resource types:

- Alert Destination
- Business Service
- Custom Resources
- Proxy Service
- Service Account
- Service Key Provider
- Split-Join
- JNDI Provider
- SMTP Server
- Proxy Server
- UDDI Registry

2.1.2 Cloning Oracle Service Bus Projects and Folders

To clone Oracle Service Bus projects and folders:

1. In the Project Explorer, right-click the Oracle Service Bus project or folder you want to clone.
2. From the menu, select **Oracle Service Bus > Clone** to display the Select Clone Target Dialog. Enter information as appropriate.

2.1.3 Creating Oracle Service Bus Configuration Projects

In the Oracle Service Bus perspective, select **File > New > Oracle Service Bus Configuration Project** to display the New Oracle Service Bus Configuration Project Wizard. Enter information as appropriate.

2.1.4 Creating Oracle Service Bus Projects

In the Oracle Service Bus perspective, select **File > New > Oracle Service Bus Project** to display the New Oracle Service Bus Project Wizard. Enter information as appropriate.

Note: You can create an Oracle Service Bus project in an Oracle Service Bus configuration project only.

2.1.5 Creating Custom Resources

In the Oracle Service Bus perspective, select **File > New > Custom Resource** to display New Custom Resource Wizard. Enter information as appropriate.

Note: You can create a custom resource in an Oracle Service Bus project only.

2.1.6 Creating JNDI Provider Resources

In the Oracle Service Bus perspective, select **File > New > JNDI Provider** to display the New JNDI Provider Resource Wizard. Enter information as appropriate.

Note: You can create a JNDI provider resource in an Oracle Service Bus configuration project only.

2.1.7 Creating Proxy Server Resources

In the Oracle Service Bus perspective, select **File > New > Proxy Server** to display the New Proxy Server Resource wizard. Enter the information described in [Section 4.9, "Proxy Servers"](#) as appropriate.

Note: You can create a proxy server resource in an Oracle Service Bus configuration project only.

2.1.8 Creating Message Format Files

In the Oracle Service Bus perspective, select **File > New > MFL** to display the New Message Format File wizard. Enter information as appropriate.

Note: You can create a message format file in an Oracle Service Bus project only.

2.1.9 Editing JNDI Provider Resources

To edit JNDI provider resources:

1. In the Project Explorer, find the Oracle Service Bus configuration project containing the JNDI provider resource you want to edit.
2. Double-click the name of the JNDI provider to display the JNDI Provider Editor. Edit as appropriate.

2.1.10 Editing Proxy Server Resources

To edit proxy server resources:

1. In the Project Explorer, find the Oracle Service Bus configuration project containing the proxy server resource you want to edit.

2. Double-click the name of the proxy server to display the Proxy Server Editor. Edit the information described in [Section 4.9, "Proxy Servers"](#) as appropriate.

2.1.11 Exporting Resources

This section describes different ways to import resources into Oracle Service Bus. This section includes the following topics:

- [Section 2.1.11.1, "Using the Export Wizard"](#)
- [Section 2.1.11.2, "Using the Command Line or a Script to Export an Oracle Service Bus Configuration"](#)

2.1.11.1 Using the Export Wizard

In the Oracle Service Bus perspective, select **File > Export** to display the Export wizard. See:

- [Section 4.6.1, "Export Wizard - Oracle Service Bus Configuration JAR Export Page"](#)
- [Section 4.6.2, "Export Wizard - Export to Server - Select Resources Page"](#)

2.1.11.2 Using the Command Line or a Script to Export an Oracle Service Bus Configuration

This section describes scripting and command-line options for exporting an Oracle Service Bus configuration:

- [Section 2.1.11.2.2, "Exporting a Configuration Using Ant"](#)
- [Section 2.1.11.2.3, "Exporting a Configuration Using WLST"](#)
- [Section 2.1.11.2.4, "Exporting a Configuration Using the Command Line"](#)

2.1.11.2.1 Before You Begin Refer to the following prerequisites and guidance before you begin.

- Java 1.6 is required.
- Oracle Service Bus 10gR3 MP1 or later and Eclipse must be installed.
- You may see exception stack traces in out the output or the workspace log file if workspace files are read-only.
- An exit value of 0 means the export succeeded.

2.1.11.2.2 Exporting a Configuration Using Ant You can export an Oracle Service Bus configuration using an Ant buildfile. Following is a sample buildfile with accompanying properties file.

Ant Buildfile Example

```
<project name="ConfigExport">
  <property file="./build.properties"/>
  <property name="eclipse.home"
    value="${oracle.home}/OEPE/eclipse"/>
  <property name="weblogic.home" value= "${oracle.home}/wlserver_10.3"/>
  <property name="metadata.dir" value="${workspace.dir}/.metadata"/>
  <target name="export">
    <available file="${metadata.dir}" type="dir"
      property="metadata.dir.exists"/>
    <java dir="${eclipse.home}"
      jar="${eclipse.home}/plugins/org.eclipse.equinox.launcher_1.0.201.R35x_
v20090715.jar"
```

```

        fork="true"
        failonerror="true"
        maxmemory="768m">
        <arg line="-data ${workspace.dir}"/>
        <arg line="-application com.bea.alsb.core.ConfigExport"/>
        <arg line="-configProject ${config.project}"/>
        <arg line="-configJar ${config.jar}"/>
        <sysproperty key="weblogic.home" value="${weblogic.home}"/>
    </java>
<antcall target="deleteMetadata"/>
</target>
<target name="deleteMetadata" unless="metadata.dir.exists">
    <delete failonerror="false" includeemptydirs="true"
        dir="${metadata.dir}"/>
</target>
</project>

```

build.properties Example

```

oracle.home=c:/oracle
workspace.dir=c:/oracle/user_projects/workspaces/default
config.project="OSB Configuration"
config.jar=c:/sbconfig.jar

```

Running "ant export" results in exporting the project "OSB Configuration" from the default workspace to c:\sbconfig.jar.

2.1.11.2.3 Exporting a Configuration Using WLST You can export an Oracle Service Bus configuration using the WebLogic Scripting Tool (WLST). For more information, see the WLST import/export sample at http://www.oracle.com/technology/sample_code/products/osb/index.html.

2.1.11.2.4 Exporting a Configuration Using the Command Line Oracle Service Bus provides a ConfigExport class that you can configure and launch using the following command line arguments. Command line export is for more advanced users who need more flexibility.

```

java -Xms384m -Xmx768m
-Dosgi.bundlefile.limit=500
-Dosb.home=OSB_ORACLE_HOME
-Dweblogic.home=WEBLOGIC_HOME
-jar ECLIPSE_HOME/eclipse/plugins/org.eclipse.equinox.launcher_launcher_
version.jar
-data WORKSPACE_DIR
-application com.bea.alsb.core.ConfigExport
-configProject PROJECT_NAME
-configJar config_filename.jar

```

where

- **OSB_ORACLE_HOME** is the top-level Oracle Service Bus directory in the Oracle Fusion Middleware home.
- **WEBLOGIC_HOME** is the location of the installed Oracle WebLogic Server.
- **ECLIPSE_HOME** is the location of the installed Eclipse that is linked to the Oracle Service Bus IDE plug-ins.
- **launcher_version** is the version of the Eclipse launcher JAR.

- *WORKSPACE_DIR* is the location that contains Oracle Service Bus artifacts to be exported. For example, `c:/oracle/user_projects/workspaces/default`. If this location contains an Eclipse workspace, the workspace is used and the configuration jar is exported from the workspace projects. If this location does not contain a workspace, but instead contains only Eclipse Oracle Service Bus projects, the utility imports the projects into a temporary workspace for the configuration JAR export.
- *PROJECT_NAME* is the name of the Oracle Service Bus Configuration project to be exported. For example, "OSB Configuration." If you do not specify this argument, the first Oracle Service Bus Configuration Project found in the workspace is exported.
- *config_filename.jar* is the name and location of the Oracle Service Bus Configuration JAR to be exported. For example, `c:/sbconfig.jar`.

Following is an example of exporting an Oracle Service Bus Configuration from the command line.

```
java -Xms384m -Xmx768m
-Dosgi.bundlefile.limit=500
-Dosb.home=D:/oracle/Oracle_OSB1
-Dweblogic.home=D:/oracle/wlserver_10.3
-jar D:/oracle/oepe_11gR1PS1/eclipse/plugins/org.eclipse.equinox.launcher_
1.0.201.R35x_v20090715.jar
-data D:/oracle/user_projects/myWorkspace
-application com.bea.alsb.core.ConfigExport
-configProject config
-configJar sbconfig.jar
```

2.1.12 Generating an Effective WSDL

To generate an effective WSDL:

1. In the Project Explorer, find the proxy service or business service from which you want to generate the effective WSDL.
2. Right-click the name of the service and select **Oracle Service Bus > Generate Effective WSDL** from the menu.
3. Select a location and save the file.

2.1.13 Modifying JAR Dependencies

To modify JAR dependencies:

1. In the Project Explorer, find the JAR file whose dependencies you want to modify.
2. Right-click the name of the file and select **Oracle Service Bus > Modify JAR Dependencies** from the menu.
3. Make modifications in the Modify JAR Dependencies Dialog.

2.1.14 Importing Resources

This section describes different ways to import resources into Oracle Service Bus. This section includes the following topics:

- [Section 2.1.14.1, "Using the Import Wizard"](#)
- [Section 2.1.14.2, "Using the Command Line or a Script to Import an Oracle Service Bus Configuration"](#)

2.1.14.1 Using the Import Wizard

In the Oracle Service Bus perspective, select **File > Import** to display the Import wizard. See:

- [Section 4.7.1, "Import Wizard - Config JAR Import - Load Resources Page"](#)
- [Section 4.7.2, "Import Wizard - Config JAR Import - Review Resources Page"](#)
- [Section 4.7.3, "Import Wizard - Config ZIP Import - Load Resources Page"](#)
- [Section 4.7.4, "Import Wizard - Config ZIP Import - Review Resources Page"](#)
- [Section 4.7.5, "Import Wizard - URL Import - Load Resources Page"](#)
- [Section 4.7.6, "Import Wizard - URL Import - Review Resources Page"](#)

2.1.14.2 Using the Command Line or a Script to Import an Oracle Service Bus Configuration

You can use scripting and command-line options for importing an Oracle Service Bus configuration. Use the examples in [Section 2.1.11.2.2, "Exporting a Configuration Using Ant"](#) and [Section 2.1.11.2.4, "Exporting a Configuration Using the Command Line"](#) for guidance on importing.

2.1.15 Creating Servers

In the Oracle Service Bus perspective, select **File > New > Server** to display the New Server wizard.

2.1.16 Creating Service Account Resources

In the Oracle Service Bus perspective, select **File > New > Service Account** to display the New Service Account Resource Wizard.

Note: You can create a service account resource in an Oracle Service Bus project only.

For configuration details, see [Section 4.19, "Service Accounts."](#)

2.1.17 Creating Service Key Provider Resources

In the Oracle Service Bus perspective, select **File > New > Service Key Provider** to display the New Service Key Provider Resource Wizard.

Note: You can create a service key provider resource in an Oracle Service Bus project only.

2.1.18 Creating SMTP Server Resources

In the Oracle Service Bus perspective, select **File > New > SMTP Server** to display the New SMTP Server Resource Wizard. Enter information as appropriate. For more information, see [Section 4.13, "SMTP Servers."](#)

Note: You can create an SMTP server resource in an Oracle Service Bus configuration project only.

2.1.19 Creating XQuery Transformations

In the Oracle Service Bus perspective, select **File > New > XQuery Transformation** to display the XQuery/XSLT Expression Editor. Enter information as appropriate. For more information, see "Oracle XQuery Mapper" in this guide.

Note: You can create an XQuery transformation resource in an Oracle Service Bus project only.

2.1.20 Creating XSL Transformations

In the Oracle Service Bus perspective, select **File > New > XSL Transformation** to display the XPath Expression Editor.

Note: You can create an XSL transformation resource in an Oracle Service Bus project only.

2.2 Working with Business Services

The following topics describe how to create and work with business services in the Oracle Service Bus plug-ins.

- [Section 2.2.1, "Creating Business Services"](#)
- [Appendix 2.2.2, "Generating a JCA Business Service from an Outbound JCA File"](#)
- [Section 2.2.3, "Generating a Business Service from Oracle Enterprise Repository"](#)
- [Section 2.2.4, "Editing Business Services"](#)

2.2.1 Creating Business Services

In the Oracle Service Bus perspective, select **File > New > Business Service** to display the New Business Service wizard. See:

- [Section 4.2.1, "Business Service General Configuration Page"](#)
- [Section 4.2.2, "Business Service Message Type Configuration Page"](#)
- [Section 4.2.6, "Business Service - Service Policy Configuration Page"](#)
- [Section 4.2.3, "Business Service SOAP Binding Configuration Page"](#)
- [Section 4.2.4, "Business Service Transport Configuration Page"](#)

Note: You can create a business service in an Oracle Service Bus project only.

2.2.1.1 Generating a Business Service from an Existing Service

To generate a business service from an existing proxy or business service:

1. In the Project Explorer, right-click the existing service and select **Oracle Service Bus > Generate Business Service**.
2. Name and configure the service.

2.2.2 Generating a JCA Business Service from an Outbound JCA File

Oracle Service Bus lets you generate business services from outbound JCA files. JCA services, which use the Oracle Service Bus JCA transport, communicate with back-end Enterprise Information Systems (EIS) through a JCA adapter framework and JCA-compliant adapters. For example, you could update back-end database records using an Oracle Service Bus JCA business service that communicates with the Oracle Database Adapter.

To create a JCA business service in Oracle Service Bus:

1. In Oracle JDeveloper, create a JCA file, its associated abstract WSDL, and any other required resources, such as a TopLink mapping file. For more information, see the *Oracle Fusion Middleware User's Guide for Technology Adapters*.
2. Import the JCA resource files into an Oracle Service Bus project so that all references to dependencies are maintained. For more information, see [Section 2.1.14, "Importing Resources."](#)
3. In Eclipse, right-click the outbound JCA file and choose **Oracle Service Bus > Generate Service**.
4. In the JCA Generate Business Service window, select the folder location for the new service, and, if desired, change the default service name.
5. Click **OK**. Oracle Service Bus generates the business service and the concrete WSDL that is used by the business service.

For more information on the Oracle Service Bus JCA transport, see "JCA Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV910>.

2.2.3 Generating a Business Service from Oracle Enterprise Repository

You can generate business services from service artifacts in Oracle Enterprise Repository. For more information, see [Section 2.7.1, "Generating Business Services from Oracle Enterprise Repository."](#)

You can also upload Oracle Service Bus projects to Oracle Enterprise Repository with Harvester, described in [Section 2.7, "Working with Oracle Enterprise Repository and Harvester."](#)

2.2.3.1 Re-generating an Existing Business Service from Oracle Enterprise Repository

You can re-generate a business service you previously generated from Oracle Enterprise Repository by following the same process described in [Section 2.2.3, "Generating a Business Service from Oracle Enterprise Repository."](#) Re-generating lets you pick up service updates in your development environment.

When you re-generate a service, Oracle Service Bus merges the service definitions, updating the existing service with changes in the Oracle Enterprise Repository but retaining service account and large message support configurations you have made in the development environment.

2.2.4 Editing Business Services

To edit business services:

1. In the Project Explorer, find the business service you want to edit.
2. Double-click the name of the service.

3. Select the page containing the options you want to edit, and make changes as appropriate. See:
 - [Section 4.2.1, "Business Service General Configuration Page"](#)
 - [Section 4.2.2, "Business Service Message Type Configuration Page"](#)
 - [Section 4.2.6, "Business Service - Service Policy Configuration Page"](#)
 - [Section 4.2.3, "Business Service SOAP Binding Configuration Page"](#)
 - [Section 4.2.4, "Business Service Transport Configuration Page"](#)
 - [Section 4.2.5, "Business Service Message Handling Configuration Page"](#)

2.3 Working with Proxy Services

The following topics describe how to create and work with proxy services in the Oracle Service Bus plug-ins.

- [Section 2.3.1, "Creating Proxy Services"](#)
- [Section 2.3.2, "Generating a JCA Proxy Service from an Inbound JCA File"](#)
- [Section 2.3.3, "Consuming Oracle Service Bus Proxy Services in Oracle JDeveloper with WSIL"](#)
- [Section 2.3.4, "Editing Proxy Services"](#)

2.3.1 Creating Proxy Services

In the Oracle Service Bus perspective, select **File > New > Proxy Service** to display the New Proxy Service wizard. See:

- [Section 4.3.1, "Proxy Service General Configuration Page"](#)
- [Section 4.3.8, "Proxy Service Security Configuration Page"](#)
- [Section 4.3.2, "Proxy Service Message Type Configuration Page"](#)
- [Section 4.3.3, "Proxy Service Operation Selection Configuration Page"](#)
- [Section 4.3.4, "Proxy Service Message Handling Configuration Page"](#)
- [Section 4.3.7, "Proxy Service - Service Policy Configuration Page"](#)
- [Section 4.3.5, "Proxy Service SOAP Binding Configuration Page"](#)
- [Section 4.3.6, "Proxy Service Transport Configuration Page"](#)

Note: You can create a proxy service in an Oracle Service Bus project only.

2.3.1.1 Generating a Proxy Service from an Existing Service

To generate a proxy service from an existing business or proxy service:

1. In the Project Explorer, right-click the existing service and select **Oracle Service Bus > Generate Proxy Service**.
2. Name and configure the service.

For a proxy services generated from a business service, the message flow automatically includes a route node to the business service.

2.3.2 Generating a JCA Proxy Service from an Inbound JCA File

Oracle Service Bus lets you generate proxy services from inbound JCA files. JCA services, which use the Oracle Service Bus JCA transport, communicate with Enterprise Information Systems (EIS) through a JCA adapter framework and JCA-compliant adapters. For example, you could invoke an external service from an EIS application through Oracle Service Bus using JCA.

To create a JCA proxy service in Oracle Service Bus:

1. In Oracle JDeveloper, create a JCA file, its associated abstract WSDL, and any other required resources, such as a TopLink mapping file. For more information, see the *Oracle Fusion Middleware User's Guide for Technology Adapters*.
2. Import the JCA resource files into an Oracle Service Bus project so that all references to dependencies are maintained. For more information, see [Section 2.1.14, "Importing Resources."](#)
3. In Eclipse, right-click the inbound JCA file and choose **Oracle Service Bus > Generate Service**.
4. In the JCA Generate Proxy Service window, select the folder location for the new service, and, if desired, change the default service name.
5. Click **OK**. Oracle Service Bus generates the proxy service and the concrete WSDL that is used by the proxy service.

For more information on the Oracle Service Bus JCA transport, see "JCA Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV910>.

2.3.3 Consuming Oracle Service Bus Proxy Services in Oracle JDeveloper with WSIL

Oracle Service Bus makes its WSDL-based proxy services available through the Web Services Inspection Language (WSIL), letting you consume Oracle Service Bus WSDL proxy services in Oracle JDeveloper for service orchestration in Oracle SOA Suite.

The Oracle Service Bus WSIL servlet automatically registers WSDL-based proxy services deployed in the Oracle Service Bus run-time environment. By creating a WSIL connection in JDeveloper, you can access those proxy services through different URL patterns that map to different hierarchy levels, such as project, folder, and individual service. For example, when you connect to the Oracle Service Bus WSIL servlet with a project-level URL, you can see all the child folders and WSDL-based proxy services in that project in Oracle JDeveloper.

The following procedure guides you through the process of creating a WSIL connection in JDeveloper and generating Web service references out of Oracle Service Bus WSDL proxy services for use in SOA applications.

1. In Oracle JDeveloper, open or create a SOA application.
2. Create a new WSIL connection.

In the Resource Palette, click the **New** icon and choose **New Connection > WSIL**.

In the Create WSIL Connection window:

- Enter the connection name.
- Enter the credentials for one of the following Oracle Service Bus roles: IntegrationAdmin, IntegrationDeployer, IntegrationOperator, or IntegrationMonitor.
- Enter the URL to the Oracle Service Bus WSIL in one of the following formats:

- Domain (gets all projects, folders, and WSDL proxy services) –
`http://host:port/sbinspection.wsil`
- Project (gets all child folders and WSDL proxy services) –
`http://host:port/sbinspection.wsil?refpath=project_name`
- Folder (in a project, gets the folder, all child folders, and WSDL proxy services) –
`http://host:port/sbinspection.wsil?refpath=project_name/folder_path`. For example:
`http://localhost:7021/sbinspection.wsil?refpath=MortgageBroker/ProxyServices`
- Proxy Service (gets an individual WSDL proxy service) –
`http://host:port/sbinspection.wsil?refpath=project_name/folder_path/wsdl_proxy_service`. For example:
`http://localhost:7021/sbinspection.wsil?refpath=MortgageBroker/ProxyServices/loanGateway1`

In a cluster, the WSIL servlet is deployed on managed servers and not the admin server. Use a managed server host name and port in the URL.

When finished, click **OK**. The WSIL connection is displayed in the Resource Palette in the hierarchy determined by the URL you entered.

3. To use an Oracle Service Bus WSDL proxy service in your SOA application, create a Web service reference to it.

In the Component Palette, create a new Web service. In the Create Web Service window, click the **WSDL URL** browse icon.

In the SOA Resource Browser, select **Resource Palette**, and select the Oracle Service Bus WSDL proxy service in the WSIL connection created in the previous step.

When you create the Web service reference to an Oracle Service Bus WSDL proxy service, you can use it as an external reference in your SOA application.

The Oracle Service Bus WSIL servlet leverages the SBResource servlet. If the SBResource is undeployed, the WSIL connection is not available.

2.3.4 Editing Proxy Services

To edit proxy services:

1. In the Project Explorer, find the proxy service you want to edit.
2. Double-click the name of the service.
3. Select the page containing the options you want to edit, and make changes as appropriate. See:
 - [Section 4.3.1, "Proxy Service General Configuration Page"](#)
 - [Section 4.3.8, "Proxy Service Security Configuration Page"](#)
 - [Section 4.3.2, "Proxy Service Message Type Configuration Page"](#)
 - [Section 4.3.3, "Proxy Service Operation Selection Configuration Page"](#)
 - [Section 4.3.4, "Proxy Service Message Handling Configuration Page"](#)
 - [Section 4.3.7, "Proxy Service - Service Policy Configuration Page"](#)
 - [Section 4.3.5, "Proxy Service SOAP Binding Configuration Page"](#)
 - [Section 4.3.6, "Proxy Service Transport Configuration Page"](#)

2.4 Working with Proxy Service Message Flows

The following topics describe how to add and configure nodes and actions to proxy service message flows.

- [Section 2.4.1, "Constructing Proxy Service Message Flows"](#)
- [Section 2.4.2, "Adding and Configuring Alert Actions in Message Flows"](#)
- [Section 2.4.3, "Adding and Configuring Assign Actions in Message Flows"](#)
- [Section 2.4.4, "Adding and Configuring Conditional Branch Nodes in Message Flows"](#)
- [Section 2.4.6, "Adding and Configuring Dynamic Publish Actions in Message Flows"](#)
- [Section 2.4.7, "Adding and Configuring Dynamic Routing Actions in Message Flows"](#)
- [Section 2.4.8, "Adding and Configuring Error Handlers in Message Flows"](#)
- [Section 2.4.9, "Adding and Configuring For-Each Actions in Message Flows"](#)
- [Section 2.4.10, "Adding and Configuring If-Then Actions in Message Flows"](#)
- [Section 2.4.11, "Adding and Configuring Insert Actions in Message Flows"](#)
- [Section 2.4.12, "Adding and Configuring Java Callout Actions in Message Flows"](#)
- [Section 2.4.13, "Adding and Configuring Log Actions in Message Flows"](#)
- [Section 2.4.14, "Adding and Configuring MFL Transform Actions in Message Flows"](#)
- [Section 2.4.15, "Adding and Configuring Operational Branch Nodes in Message Flows"](#)
- [Section 2.4.16, "Adding and Configuring Pipeline Pair Nodes in Message Flows"](#)
- [Section 2.4.17, "Adding and Configuring Publish Actions in Message Flows"](#)
- [Section 2.4.18, "Adding and Configuring Publish Table Actions in Message Flows"](#)
- [Section 2.4.19, "Adding and Configuring Raise Error Actions in Message Flows"](#)
- [Section 2.4.20, "Adding and Configuring Rename Actions in Message Flows"](#)
- [Section 2.4.21, "Adding and Configuring Replace Actions in Message Flows"](#)
- [Section 2.4.23, "Adding and Configuring Report Actions in Message Flows"](#)
- [Section 2.4.21, "Adding and Configuring Replace Actions in Message Flows"](#)
- [Section 2.4.22, "Adding and Configuring Reply Actions in Message Flows"](#)
- [Section 2.4.24, "Adding and Configuring Resume Actions in Message Flows"](#)
- [Section 2.4.25, "Adding and Configuring Route Nodes in Message Flows"](#)
- [Section 2.4.26, "Adding and Configuring Routing Actions in Message Flows"](#)
- [Section 2.4.27, "Adding and Configuring Routing Options Actions in Message Flows"](#)
- [Section 2.4.28, "Adding and Configuring Routing Table Actions in Message Flows"](#)
- [Section 2.4.29, "Adding and Configuring Service Callout Actions in Message Flows"](#)

- [Section 2.4.30, "Adding and Configuring Skip Actions in Message Flows"](#)
- [Section 2.4.31, "Adding and Configuring Stages in Message Flows"](#)
- [Section 2.4.32, "Adding and Configuring Transport Headers Actions in Message Flows"](#)
- [Section 2.4.33, "Adding and Configuring Validate Actions in Message Flows"](#)

2.4.1 Constructing Proxy Service Message Flows

When you create a proxy service, a message flow is created by default, with an empty starting node. The process for constructing the message flow follows this general pattern:

1. Open the Message Flow Editor for the proxy service. To open the proxy service, double-click its name in Project Explorer. The Message Flow Editor appears as a tab in the proxy service view.
2. Open the Message Flow Design Palette. To open the palette, in the Oracle Service Bus perspective, select **Window > Show View > Design Palette**.
3. Open the Properties view, if it is not already open:
 - a. In the Oracle Service Bus perspective, select **Window > Show View > Other**.
 - b. In the Show View dialog, select **General > Properties**.
4. Drag nodes and actions from the Message Flow Design Palette to the Message Flow Editor.

Alternatively, you can right-click a node or action in the Message Flow Editor to display menus of nodes and actions that can be inserted in that location. The menu contains one or more the following:

- **Insert >** (list of nodes and actions)
 - **Insert Into >** (list of nodes and actions)
 - **Insert After >** (list of nodes and actions)
 - **Add Error Handler**
5. Configure nodes and actions:
 - a. In the Proxy Service Editor, select the node or action by clicking it.

Alternatively, you can select a node or an action from the Outline view. To open the Outline view, in the Oracle Service Bus perspective, select **Window > Show View > Outline**.
 - b. In the Properties view, set the properties, as appropriate for the selected node or action. For instructions on how to configure the nodes and actions, click the Properties view for a node or action, and press F1 for help.

2.4.2 Adding and Configuring Alert Actions in Message Flows

Use the alert action to generate alerts based on message context in a pipeline, to send to an alert destination.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add an alert action

1. In the Message Flow Design Palette, open the **Stage Actions > Reporting** list, if it is not already open.
2. Drag the alert action to the desired location in the message flow.

To configure the alert action

1. In the Message Flow Editor, click the alert action, if it is not already selected.
2. On the Alert Action Properties page, edit properties as appropriate.

2.4.3 Adding and Configuring Assign Actions in Message Flows

Use an assign action to assign the result of an XQuery expression to a context variable.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows"](#).

To add an assign action

1. In the Message Flow Design Palette, open the **Stage Actions > Message Processing** list, if it is not already open.
2. Drag the assign action to the desired location in the message flow.

To configure the assign action

1. In the Message Flow Editor, click the assign action, if it is not already selected.
2. On the Assign Action Properties page, edit properties as appropriate.

2.4.4 Adding and Configuring Conditional Branch Nodes in Message Flows

Use a conditional branch node to specify that message processing is to proceed along exactly one of several possible paths, based on a result returned by an XPath condition.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a conditional branch node

1. In the Message Flow Design Palette, open the **Oracle Service Bus Message Flow > Nodes** list, if it is not already open.
2. Drag the conditional branch node to the desired location in the message flow.

To configure the conditional branch node

1. In the Message Flow Editor, click the conditional branch node, if it is not already selected.
2. On the Conditional Branch Node Properties page, edit properties as appropriate.

2.4.5 Adding and Configuring Delete Actions in Message Flows

Use a delete action to delete a context variable or a set of nodes specified by an XPath expression.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a delete action

1. In the Message Flow Design Palette, open the **Stage Actions > Message Processing** list, if it is not already open.
2. Drag the delete action to the desired location in a stage action in the message flow.

To configure the delete action

1. In the Message Flow Editor, click the delete action, if it is not already selected.
2. On the Delete Action Properties page, edit properties as appropriate.

2.4.6 Adding and Configuring Dynamic Publish Actions in Message Flows

Use a dynamic publish action to publish a message to a service specified by an XQuery expression.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a dynamic publish action

1. In the Message Flow Design Palette, open the **Stage Actions > Communication** list, if it is not already open.
2. Drag the dynamic publish action to the desired location in the message flow.

To configure the dynamic publish action

1. In the Message Flow Editor, click the dynamic publish action, if it is not already selected.
2. On the Dynamic Publish Action Properties page, edit properties as appropriate.

2.4.7 Adding and Configuring Dynamic Routing Actions in Message Flows

Use a dynamic routing action to assign a route for a message based on routing information available in an XQuery resource.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a dynamic routing action

1. In the Message Flow Design Palette, open the **Route Actions > Communication** list, if it is not already open.
2. Drag the dynamic routing action to the route action in the message flow.

To configure the dynamic routing action

1. In the Message Flow Editor, click the dynamic routing action, if it is not already selected.
2. On the Dynamic Routing Action Properties page, edit properties as appropriate.

2.4.8 Adding and Configuring Error Handlers in Message Flows

Use an error handler to specify what should happen if an error occurs in a specific location in the message flow.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add an error handler

1. In the Message Flow Design Palette, open the **Oracle Service Bus Message Flow > Nodes** list, if it is not already open.
2. Drag the error handler to the desired location in the message flow.
3. Drag a stage node to the error handler.
4. Add actions to the stage, as appropriate, to define the error handler.

To configure the error handler

1. In the Message Flow Editor, click the error handler, if it is not already selected.
2. On the Error Handler Node Properties page, edit the properties.
3. Click the stage node, if it is not already selected.
4. On the Stage Node Properties page, edit the properties.
5. Select and edit any actions contained by the stage, as appropriate.

2.4.9 Adding and Configuring For-Each Actions in Message Flows

Use the for-each action to iterate over a sequence of values and execute a block of actions.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a for-each action

1. In the Message Flow Design Palette, open the **Stage Actions > Flow Control** list, if it is not already open.
2. Drag the for-each action to the desired stage action in the message flow.

To configure the for-each action

1. In the Message Flow Editor, click the for-each action, if it is not already selected.
2. On the For-Each Action Properties page, edit properties as appropriate.

2.4.10 Adding and Configuring If-Then Actions in Message Flows

Use an if-then action to perform an action or a set of actions conditionally, based on the Boolean result of an XQuery expression.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add an if-then action

1. In the Message Flow Design Palette, do one of the following:
 - For an if-then action in a route node, open the **Route Actions > Flow Control** list, if it is not already open.

- For an if-then action in a stage node, open the **Stage Actions > Flow Control** list, if it is not already open.
2. Drag the if-then action to the route node or to the desired stage action in the message flow.

To configure the if-then action

In the Message Flow Editor, click each if condition and else-if condition contained by the if-then action, and define the conditions in the Condition Editor, as described in [Section 4.11.9, "If-Then Action Properties."](#)

2.4.11 Adding and Configuring Insert Actions in Message Flows

Use an insert action to insert the result of an XQuery expression at an identified place relative to nodes selected by an XPath expression.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add an insert action

1. In the Message Flow Design Palette, open the **Stage Actions > Message Processing** list, if it is not already open.
2. Drag the insert action to the desired location in the message flow.

To configure the insert action

1. In the Message Flow Editor, click the insert action, if it is not already selected.
2. On the Insert Action Properties page, edit properties as appropriate.

2.4.12 Adding and Configuring Java Callout Actions in Message Flows

Use a Java callout action to invoke a Java method or an EJB business service from within the message flow.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add an Java callout action

1. In the Message Flow Design Palette, open the **Stage Actions > Message Processing** list, if it is not already open.
2. Drag the Java callout action to the desired location in the message flow.

To configure the Java callout action

1. In the Message Flow Editor, click the Java callout action, if it is not already selected.
2. On the Java Callout Action Properties page, edit properties as appropriate.

2.4.13 Adding and Configuring Log Actions in Message Flows

Use the log action to construct a message to be logged and to define a set of attributes with which it will be logged.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a log action

1. In the Message Flow Design Palette, open the **Stage Actions > Reporting** list, if it is not already open.
2. Drag the log action to the desired location in the message flow.

To configure the log action

1. In the Message Flow Editor, click the log action, if it is not already selected.
2. On the Log Action Properties page, edit properties as appropriate.

2.4.14 Adding and Configuring MFL Transform Actions in Message Flows

Use a MFL (Message Format Language) transform action to convert message content from XML to non-XML, or vice versa, in the message pipeline.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a MFL transform action

1. In the Message Flow Design Palette, open the **Stage Actions > Message Processing** list, if it is not already open.
2. Drag the MFL transform action to the desired location in the message flow.

To configure the MFL transform action

1. In the Message Flow Editor, click the MFL transform action, if it is not already selected.
2. On the MFL Transform Action Properties page, edit properties as appropriate.

2.4.15 Adding and Configuring Operational Branch Nodes in Message Flows

Use an operational branch node to configure branching based on operations defined in a WSDL.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add an operational branch node

1. In the Message Flow Design Palette, open the **Oracle Service Bus Message Flow > Nodes** list, if it is not already open.
2. Drag the operational branch node to the desired location in the message flow.

To configure the operational branch node

1. In the Message Flow Editor, click the operational branch node, if it is not already selected.
2. On the Operational Branch Node Properties page, edit properties as appropriate.

2.4.16 Adding and Configuring Pipeline Pair Nodes in Message Flows

Use a pipeline pair node to define request and response processing.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a pipeline pair node

1. In the Message Flow Design Palette, open the **Oracle Service Bus Message Flow > Nodes** list, if it is not already open.
2. Drag the pipeline pair node to the desired location in the message flow.

To configure the pipeline pair node

1. In the Message Flow Editor, click the pipeline pair node, if it is not already selected.
2. On the Pipeline Pair Node Properties page, edit properties as appropriate.

2.4.17 Adding and Configuring Publish Actions in Message Flows

Use a publish action to identify a statically specified target service for a message and to configure how the message is packaged and sent to that service.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a publish action

1. In the Message Flow Design Palette, open the **Stage Actions > Communication** list, if it is not already open.
2. Drag the publish action to the desired location in the message flow.

To configure the publish action

1. In the Message Flow Editor, click the publish action, if it is not already selected.
2. On the Publish Action Properties page, edit properties as appropriate.

2.4.18 Adding and Configuring Publish Table Actions in Message Flows

Use a publish table action to publish a message to zero or more statically specified services.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a publish table action

1. In the Message Flow Design Palette, open the **Stage Actions > Communication** list, if it is not already open.
2. Drag the publish table action to the desired location in the message flow.

To configure the publish table action

1. In the Message Flow Editor, click the publish table action, if it is not already selected.

2. On the Publish Table Action Properties page, click <Expression> to display the XQuery/XSLT Expression Editor. Create an XQuery expression, which at run time returns the value upon which the routing decision will be made.
3. In the Message Flow Editor, select a case action.
4. From the **Operator** list on the Publish Table Action Properties page, select a comparison operator. Then, in the **Value** field, enter a value against which the value returned from the XQuery expression will be evaluated.
5. In the Message Flow Editor, click one of the publish table's publish actions to select it.
6. On the Publish Action Properties page, click **Browse** to select a service. Select the service to which messages are to be published if the expression evaluates true for the value you specified. The Select a Service Resource dialog is displayed.
7. Select a service from the list, then click **OK**. This is the target service for the message.
8. If the service has operations defined, you can specify the operation to be invoked by selecting it from the **invoking** list.
9. If you want the outbound operation to be the same as the inbound operation, select the **Use inbound operation for outbound** check box.
10. In the **Request Actions** field, to configure how the message is packaged and sent to the service, click **Add an Action**, then select one or more actions that you want to associate with the service.

Note: There is a nesting limit of four cumulative levels in the stage editor. If you attempt to add a fifth level, nesting action is not displayed. Cumulative levels include all branching actions: if... then... conditions, publish tables, and route tables. For example, you can have 2 levels of conditionals, then a publish table with a route table inside of it, bringing the total to 4 levels. If you attempt to add another conditional (to the last publish table), the conditional is not displayed.

11. To insert a new case, click the **Case** icon, then select **Insert New Case**.
12. Repeat steps 4-8 for the new case.
13. Add additional cases as dictated by your business logic.
14. Click the **Case** icon of the last case you define in the sequence, then select **Insert Default Case** to add a default case at the end.
15. Configure the default case—the configuration of this case specifies the routing behavior in the event that none of the preceding cases is satisfied.

2.4.19 Adding and Configuring Raise Error Actions in Message Flows

Use the raise error action to raise an exception with a specified error code (a string) and description.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a raise error action

1. In the Message Flow Design Palette, open the **Stage Actions > Flow Control** list, if it is not already open.
2. Drag the raise error action to the desired location in the message flow.

To configure the raise error action

1. In the Message Flow Editor, click the raise error action, if it is not already selected.
2. On the Raise Error Action Properties page, edit properties as appropriate.

2.4.20 Adding and Configuring Rename Actions in Message Flows

Use the rename action to rename elements selected by an XPath expression without modifying the contents of the element.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add an rename action

1. In the Message Flow Design Palette, open the **Stage Actions > Message Processing** list, if it is not already open.
2. Drag the rename action to the desired location in the message flow.

To configure the rename action

1. In the Message Flow Editor, click the rename action, if it is not already selected.
2. On the Rename Action Properties page, edit properties as appropriate.

2.4.21 Adding and Configuring Replace Actions in Message Flows

Use a replace action to replace a node or the contents of a node specified by an XPath expression. The node or its contents are replaced with the value returned by an XQuery expression.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a replace action

1. In the Message Flow Design Palette, open the **Stage Actions > Message Processing** list, if it is not already open.
2. Drag the replace action to the desired location in the message flow.

To configure the replace action

1. In the Message Flow Editor, click the replace action, if it is not already selected.
2. On the Replace Action Properties page, edit properties as appropriate.

2.4.22 Adding and Configuring Reply Actions in Message Flows

Use the reply action to specify that an immediate reply be sent to the invoker.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a reply action

1. In the Message Flow Design Palette, open the **Stage Actions > Flow Control** list, if it is not already open.
2. Drag the reply action to the desired location in the message flow.

To configure the reply action

1. In the Message Flow Editor, click the reply action, if it is not already selected.
2. On the Reply Action Properties page, edit properties as appropriate.

2.4.23 Adding and Configuring Report Actions in Message Flows

Use the report action to enable message reporting for a proxy service.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a report action

1. In the Message Flow Design Palette, open the **Stage Actions > Reporting** list, if it is not already open.
2. Drag the report action to the desired location in the message flow.

To configure the report action

1. In the Message Flow Editor, click the report action, if it is not already selected.
2. On the Report Action Properties page, edit properties as appropriate.

2.4.24 Adding and Configuring Resume Actions in Message Flows

Use the resume action to resume message flow after an error is handled by an error handler. This action has no parameters and can only be used in error pipelines.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a resume action

1. In the Message Flow Design Palette, open the **Stage Actions > Flow Control** list, if it is not already open.
2. Drag the resume action to the desired location in the message flow.

To configure the resume action

1. In the Message Flow Editor, click the resume action, if it is not already selected.
2. On the Resume Action Properties page, edit properties as appropriate.

2.4.25 Adding and Configuring Route Nodes in Message Flows

Use the route node to handle request and response dispatching of messages to and from business services.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a route node

1. In the Message Flow Design Palette, open the **Oracle Service Bus Message Flow > Nodes** list, if it is not already open.
2. Drag the route node to the desired location in the message flow.

To configure the route node

1. In the Message Flow Editor, click the route node action, if it is not already selected.
2. On the Route Node Properties page, edit properties as appropriate.

2.4.26 Adding and Configuring Routing Actions in Message Flows

Use a routing action to identify a target service for the message and configure how the message is routed to that service.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a routing action

1. In the Message Flow Design Palette, open the **Route Actions > Communication** list, if it is not already open.
2. Drag the routing action to the desired location in the message flow.

To configure the routing action

1. In the Message Flow Editor, click the routing action, if it is not already selected.
2. On the Routing Action Properties page, edit properties as appropriate.

2.4.27 Adding and Configuring Routing Options Actions in Message Flows

Use a routing options action to modify any or all of the following properties in the outbound request: URI, Quality of Service, Mode, Retry parameters, Message Priority.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a routing options action

1. In the Message Flow Design Palette, open the **Stage Actions > Communication** list, if it is not already open.
2. Drag the routing options action to the desired location in the message flow.

To configure the routing options action

1. In the Message Flow Editor, click the routing options action, if it is not already selected.
2. On the Routing Options Action Properties page, edit properties as appropriate.

2.4.28 Adding and Configuring Routing Table Actions in Message Flows

Use a routing table to select different routes based upon the results of a single XQuery expression. A routing table action contains a set of routes wrapped in a switch-style condition table.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a routing table action

1. In the Message Flow Design Palette, open the **Route Actions > Communication** list, if it is not already open.
2. Drag the routing table action to the desired location in the message flow.

To configure the routing table action

1. In the Message Flow Editor, click the routing table action, if it is not already selected.
2. On the Routing Table Action Properties page, edit properties as appropriate.

2.4.29 Adding and Configuring Service Callout Actions in Message Flows

Use a service callout action to configure a synchronous (blocking) callout to an Oracle Service Bus-registered proxy or business service.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a service callout action

1. In the Message Flow Design Palette, open the **Stage Actions > Communication** list, if it is not already open.
2. Drag the service callout action to the desired location in the message flow.

To configure the service callout action

1. In the Message Flow Editor, click the service callout action, if it is not already selected.
2. On the Service Callout Action Properties page, edit properties as appropriate.

2.4.30 Adding and Configuring Skip Actions in Message Flows

Use the skip action to specify that at run time, the execution of the current stage is skipped and the processing proceeds to the next stage in the message flow.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a skip action

1. In the Message Flow Design Palette, open the **Stage Actions > Flow Control** list, if it is not already open.
2. Drag the skip action to the desired location in the message flow.

To configure the skip action

1. In the Message Flow Editor, click the skip action, if it is not already selected.
2. On the Skip Action Properties page, edit properties as appropriate.

2.4.31 Adding and Configuring Stages in Message Flows

Use a stage node as a container for actions in a message flow.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a stage

1. In the Message Flow Design Palette, open the **Oracle Service Bus Message Flow > Nodes** list, if it is not already open.
2. Drag the stage to the desired location in the message flow.
3. Add actions to the stage, as appropriate for your configuration.

To configure the stage

1. In the Message Flow Editor, click the stage, if it is not already selected.
2. On the Stage Node Properties page, edit properties as appropriate.

2.4.32 Adding and Configuring Transport Headers Actions in Message Flows

Use a transport header action to set header values in messages.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a transport headers action

1. In the Message Flow Design Palette, open the **Stage Actions > Communication** list, if it is not already open.
2. Drag the transport headers action to the desired location in the message flow.

To configure the transport headers action

1. In the Message Flow Editor, click the transport headers action, if it is not already selected.
2. On the Transport Headers Action Properties page, edit properties as appropriate, as described in [Section 4.11.34, "Transport Headers Action Properties."](#)

2.4.33 Adding and Configuring Validate Actions in Message Flows

Use a validate action to validate elements selected by an XPath expression against an XML schema element or a WSDL resource.

Before you begin

Display the message flow for the desired proxy service. See [Section 2.4.1, "Constructing Proxy Service Message Flows."](#)

To add a validate action

1. In the Message Flow Design Palette, open the **Stage Actions > Message Processing** list, if it is not already open.
2. Drag the validate action to the desired location in the message flow.

To configure the validate action

1. In the Message Flow Editor, click the validate action, if it is not already selected.

2. On the Validate Action Properties page, edit properties as appropriate.

2.5 Working with Alert Destinations

The following topics describe how to create and work with alert destinations in the Oracle Service Bus plug-ins.

- [Section 2.5.1, "Creating Alert Destinations"](#)
- [Section 2.5.2, "Editing Alert Destinations"](#)
- [Section 2.5.3, "Adding E-mail Recipients to Alert Destinations"](#)
- [Section 2.5.4, "Adding JMS Destinations to Alert Destinations"](#)

2.5.1 Creating Alert Destinations

To create alert destinations:

1. In the Project Explorer in the Oracle Service Bus perspective, right-click a project or folder in which you want to create an alert destination.
2. From the menu, select **File > New > Alert Destination** to display the New Alert Destination Resource wizard. Enter information as appropriate. See [Section 4.1.1, "Alert Destination Editor."](#)

Note: You can create an alert destination in an Oracle Service Bus project only.

2.5.2 Editing Alert Destinations

To edit alert destinations:

1. In the Project Explorer, find the project folder containing the alert destination you want to edit.
2. Double-click the name of the alert destination to display the Alert Destination Editor. Edit information, as desired.

2.5.3 Adding E-mail Recipients to Alert Destinations

To add e-mail recipients to alert destinations:

1. Create or edit an alert destination, as described in [Section 2.5.1, "Creating Alert Destinations"](#) and [Section 2.5.2, "Editing Alert Destinations"](#).
2. In the **E-mail Recipients** field of the Alert Destination Editor, click **Add** to display the Edit E-mail Recipient Page. Add information, as appropriate.

2.5.4 Adding JMS Destinations to Alert Destinations

To add JMS destinations to alert destinations:

1. Create or edit an alert destination, as described in [Section 2.5.1, "Creating Alert Destinations"](#) and [Section 2.5.2, "Editing Alert Destinations"](#).
2. In the **JMS Destinations** field of the Alert Destination Editor, click **Add** to display the Edit JMS Destination Page. Add information, as appropriate.

2.6 Working with MQ Connections

MQ connections are sharable resources that can be reused across multiple MQ proxy and business services. MQ proxy and business services must connect to a MQ queue manager before accessing an MQ queue. MQ Connection resources provide the connection required for connecting to an MQ queue manager.

Each MQ Connection resource has a connection pool. Every business or proxy service using a given MQ Connection resource to get a connection to a given queue manager uses the same connection pool that was created for that resource. Thus, multiple business services and proxy services using the same queue manager share a connection pool.

To learn more about Oracle Service Bus MQ Connection resources and the native MQ transport, see "MQ Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at

<http://www.oracle.com/pls/as1111/lookup?id=OSBDV1117>.

To learn more about WebSphere MQ Fundamentals, see

<http://www.redbooks.ibm.com/redbooks/SG247128/wwhelp/wwhimpl/java/html/wwhelp.htm>.

2.6.1 Adding MQ Connections

In Oracle Service Bus, MQ connections are created as custom resources. Therefore, to add an MQ connection, you must create it as a custom resource, as follows:

1. In the Oracle Service Bus perspective, select **File > New > Custom Resource** to display New Custom Resource Wizard.
2. On the Create a New Custom Resource page, in the **Resource Type** field, select **MQ Connection**.
3. Enter configuration information, as appropriate, on the Custom MQ Resource Configuration Page.

Note: Do not include spaces in the MQ Connection resource name.

2.6.2 Editing MQ Connections

To edit MQ connections:

1. In the Project Explorer, find the Oracle Service Bus configuration folder containing the MQ connection resource you want to edit.
2. Double-click the name of the MQ connection to display the Custom MQ Resource Configuration Page. Edit as appropriate.

2.7 Working with Oracle Enterprise Repository and Harvester

Oracle Enterprise Repository is an enterprise metadata repository for application resources and services. Using Oracle Enterprise Repository in conjunction with Oracle Service Registry, Oracle's UDDI solution, is a best-practice approach to locating, using, synchronizing, and governing enterprise-wide services through Oracle Service Bus.

Harvester is an Oracle Enterprise Repository tool that lets you harvest enterprise artifacts into Oracle Enterprise Repository from multiple sources, including Oracle Service Bus.

Oracle Service Bus provides native connectivity to Oracle Enterprise Repository and Harvester in the development environment, letting you both generate business services from the repository and harvest Oracle Service Bus projects to the repository. Harvester also provides command-line scripting for harvesting Oracle Service Bus projects in the run-time environment.

For more information on Oracle Enterprise Repository, see the *Oracle Fusion Middleware Quick Start Guide for Oracle Enterprise Repository* at <http://www.oracle.com/pls/as1111/lookup?id=OERQS>.

This section includes the following topics:

- [Section 2.7.1, "Generating Business Services from Oracle Enterprise Repository"](#)
- [Section 2.7.2, "Using Harvester"](#)
- [Section 2.7.3, "Performing Queries in Oracle Enterprise Repository from Eclipse"](#)

2.7.1 Generating Business Services from Oracle Enterprise Repository

You can generate business services in the Oracle Service Bus development environment from service assets in Oracle Enterprise Repository.

In a best-practices deployment scenario, service assets in Oracle Enterprise Repository have associated services stored in Oracle Service Registry. When you generate business services from Oracle Enterprise Repository assets, those services can be synchronized with the associated services stored in the registry through Oracle Service Bus UDDI resources.

This section provides instructions on using the development environment to generate business services from service assets in Oracle Enterprise Repository.

Note: To generate business services in Oracle Service Bus, Oracle Enterprise Repository assets must be SOAP/XML WSDL-based services using HTTP.

1. Perform a query in Oracle Enterprise Repository to locate the service from which you want to generate a business service. See [Section 2.7.3, "Performing Queries in Oracle Enterprise Repository from Eclipse."](#)
2. Right-click the service asset from which you want to generate the new business service, and choose **Oracle Service Bus > Generate Business Service**.
3. In the Generate Business Service window, select a location in Eclipse for the new business service, and select an endpoint.

Note: Service assets must have an endpoint in order to generate business services in Oracle Service Bus.

4. If the Oracle Enterprise Repository asset is associated with Oracle Service Registry, you can click Next to configure an Oracle Service Bus UDDI Registry resource that at run time connects to Oracle Service Registry (otherwise, skip to the next step). The Oracle Enterprise Repository asset automatically provides the Inquiry URL to Oracle Service Registry.

The Configure UDDI Registry window provides the following options: Select a matching UDDI Registry Resource, select a non-matching UDDI Registry resource or create a new one, or not use UDDI.

If you select the option to create a new UDDI Registry resource, see [Section 4.14, "UDDI Registry Configuration Page"](#) for guidance.

5. Click **Finish** to generate the business service.

A new HTTP business service and its associated WSDL and XSD are added to your development environment.

If the Oracle Enterprise Repository asset endpoint is associated with a service in Oracle Service Registry, the business service configuration contains UDDI connection details to the Oracle Service Registry instance.

6. If the Oracle Enterprise Repository asset references an Oracle Web Services Manager policy or a WS-Policy, the generated business services does not automatically include the policy. You must manually add Oracle Web Services Manager policies and WS-Policies. WSDL-based policies, however, are automatically included.

In the development environment, you do not establish a subscription to Oracle Service Registry. The subscription and service synchronization to Oracle Service Registry occurs when you deploy the business service to the Oracle Service Bus run-time environment. Service synchronization is determined by the UDDI Registry resource's Auto-Import setting.

For more information on UDDI, see "UDDI" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBAG392>.

2.7.2 Using Harvester

Harvester lets you submit Oracle Service Bus projects to Oracle Enterprise Repository from Eclipse, the command line, with an Ant task, or with a WSLT script).

Two concepts in harvesting are local harvesting and remote harvesting. Local harvesting, performed from Eclipse and from specific command-line scripts, submits projects from the file system to the repository. Remote harvesting, performed from command-line scripts, submits projects from a run-time environment.

When you harvest resources, the dependency relationships are maintained in Oracle Enterprise Repository.

[Table 2-1](#) describes differences between harvesting scenarios to help you decide which scenario is appropriate.

Table 2-1 Differences in Harvesting Scenarios

Eclipse Harvests	Local Harvests	Remote Harvests
You can harvest a single project or an Oracle Service Bus configuration containing multiple projects.	You can use the command-line scripts to harvest Oracle Service Bus artifacts from a configuration JAR on the file system. You can harvest all the projects in the configuration jar or selected projects.	You can use the command-line scripts to harvest Oracle Service Bus artifacts from a configuration JAR in the run-time environment. You can harvest all the projects in the configuration jar or selected projects.

Table 2–1 (Cont.) Differences in Harvesting Scenarios

Eclipse Harvests	Local Harvests	Remote Harvests
Proxy service endpoint assets are not created in the repository.	Proxy service endpoint assets are not created in the repository.	Proxy service endpoint assets are created in the repository. If you perform a remote harvest on the same services you harvested from Eclipse or a local harvest, the newly created service endpoints are added to the same services that were harvested locally.
Service WSDLs (effective WSDLs) are generated in the repository with a local addresses, such as localhost:7001.	Service WSDLs (effective WSDLs) are generated in the repository with a local addresses, such as localhost:7001.	Service WSDLs (effective WSDLs) are generated in the repository with the server run-time address.

Before using Oracle Service Bus with Harvester, be sure to install the Harvester Solution Pack, as described in "Configuring and Using Automated Harvesting in Design-time and Runtime Environments" in the *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository* at <http://www.oracle.com/pls/as1111/lookup?id=OERCG>.

The following sections describe Eclipse, local, and remote harvesting.

2.7.2.1 Using Harvester from Eclipse

The Oracle Service Bus plug-ins for Eclipse provide built-in functionality for Harvesting projects to Oracle Enterprise Repository.

For information on how the assets are harvested from Eclipse, see [Table 2–1](#).

To Harvest an Oracle Service Bus project from Eclipse:

1. In Project Explorer view, right-click the project you want to harvest and select **Submit Project to Enterprise Repository**.

You can also right-click an Oracle Service Bus configuration project to harvest all projects in the configuration.

2. In the Submit window, enter the connection details to the repository, including the repository URI and login credentials.

If you have previously configured a connection to Oracle Enterprise Repository, as described in [Section 2.7.3, "Performing Queries in Oracle Enterprise Repository from Eclipse,"](#) those connection details appear by default.

Optionally:

- Enter a namespace for the project and its artifacts. The namespace prepends the name of service assets in Oracle Enterprise Repository query results.
 - Select a Registration Status for the artifacts. For example, if you are harvesting the artifacts for testing, select an appropriate status, such as Unsubmitted.
 - Modify the default connection timeout in milliseconds.
 - Modify the default description of the project(s).
3. Click **Submit**. An activity bar scrolls while the upload proceeds. When the upload finishes, click **Close**.

4. To see your harvested resources, perform a query in Oracle Enterprise Repository, described in [Section 2.7.3, "Performing Queries in Oracle Enterprise Repository from Eclipse."](#)

2.7.2.2 Using Harvester from the Command Line or a Script

Oracle Service Bus provides built-in Harvester resources necessary to harvest Oracle Service Bus projects from the command line, using Ant tasks, or using WSLT scripts. These resources are located at:

`ORACLE_OSB_HOME/harvester`

The main configuration file in the directory is `HarvesterSettings.xml`, which contains placeholder configurations for the connection to Oracle Enterprise Repository, local harvest settings, and remote harvest settings. Command-line Harvester uses the configuration details in that file to perform harvests in conjunction with any override configuration switches you provide.

Following are basic descriptions of local and remote harvesting:

- **Local Harvesting** – A local harvest involves generating an Oracle Service Bus configuration JAR on the file system and targeting that JAR for harvest. All projects in the JAR are harvested.

For information on generating an Oracle Service Bus configuration JAR, see [Section 2.1.11, "Exporting Resources"](#) or "Exporting Resources" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBAG1320>.

- **Remote Harvesting** – A remote harvest involves pointing the script at a running Oracle Service Bus instance and targeting a single project or a full Oracle Service Bus configuration for harvest.

To help you decide which harvesting option you want to use, see [Table 2-1](#).

Oracle Service Bus provides guidance and resource files for command-line, Ant, and WSLT harvesting. For details, see the `ORACLE_OSB_HOME/harvester/README.TXT` file.

The `README.TXT` file provides Oracle Service Bus-specific guidance. Use that information and the Oracle Service Bus Harvester resources in conjunction with the full set of command-line and scripting instructions in "Configuring and Using Automated Harvesting in Design-time and Runtime Environments" in the *Oracle Fusion Middleware Configuration Guide for Oracle Enterprise Repository* at <http://www.oracle.com/pls/as1111/lookup?id=OERCG>. Use *only* the options mentioned in the `README.TXT` file, as those are the only supported options with Oracle Service Bus.

After you harvest projects, you can query Oracle Enterprise Repository in Eclipse to view the harvested resources. See [Section 2.7.3, "Performing Queries in Oracle Enterprise Repository from Eclipse."](#)

Note: Besides the Oracle Service Bus harvester, which harvests full projects, the command-line framework also supports built-in harvesters for individual resources such as WSDLs, XSDs, XSLTs, and WS-Policies. However, Oracle Service Bus does not support the use of those harvesters for Oracle Service Bus 11g and later resources.

For harvesting individual resources, such as those in Oracle SOA Suite, Oracle WebLogic Server, or plain WSDLs, download and use the Harvester Base Data Pack.

2.7.3 Performing Queries in Oracle Enterprise Repository from Eclipse

The following steps show you how to connect to and query Oracle Enterprise Repository in Eclipse. Querying lets you locate service assets from which you can generate business services, browse assets in the repository, and view asset details and relationships.

1. In Eclipse, choose **Window > Show View > Other**, then choose **Oracle Enterprise Repository > Enterprise Repository Access**.
2. In Enterprise Repository Access view, click the **Connect to enterprise repository** icon.
3. In the Enterprise Repository Connection window, enter the repository URL and login credentials, and click **Establish Connection**.
4. On successful connection, click **Finish**.
5. When connected to Oracle Enterprise Repository, you can query the repository to locate the services and assets you want. For example, if you want to find service assets, select **Service** in the All Types field of the Query screen, then click the **Perform query** icon.

The Enterprise Repository Access view switches to the Results screen to display query results.

You can also open the Enterprise Repository Asset Relationships view in Eclipse to see graphical relationships among selected assets: **Window > Show View > Other**, then choose **Oracle Enterprise Repository > Enterprise Repository Asset Relationships**.

For more information on working with Oracle Enterprise Repository in Eclipse, see the *Oracle Fusion Middleware Integration Guide for Oracle Enterprise Repository* at <http://www.oracle.com/pls/as1111/lookup?id=OERIN>.

2.8 Working with UDDI Registries

Universal Description, Discovery and Integration (UDDI) registries are used in an enterprise to share Web Services. UDDI provides a framework in which to classify your business, its services, and the technical details about the services you want to expose.

Publishing a service to a registry requires knowledge of the service type and the data structure representing that service in the registry. A registry entry has certain properties associated with it and these property types are defined when the registry is created. You can publish your service to a registry and make it available for other organizations to discover and use. Proxy services developed in Oracle Service Bus can be published to a UDDI registry. Oracle Service Bus can interact with any UDDI version 3.0-compliant registry.

See also "UDDI" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

2.8.1 Adding UDDI Registries

To add UDDI registries:

1. In the Oracle Service Bus perspective, select **File > New > UDDI Registry** to display the New UDDI Resource wizard.

Note: You can add a UDDI registry only to an Oracle Service Bus configuration project only.

2. After choosing a location and entering a name, configure the registry on the UDDI Registry Configuration Page, as described in [Section 4.14, "UDDI Registry Configuration Page."](#)

Note: You can add a UDDI registry in an Oracle Service Bus configuration project only.

2.8.2 Configuring UDDI Registries

To configure UDDI registries:

1. In the Project Explorer, find the UDDI registry you want to configure.
2. Double-click the name of the registry.
3. Set configure options on the UDDI Registry Configuration Page, described in [Section 4.14, "UDDI Registry Configuration Page."](#)

2.8.3 Importing Business Services From a UDDI Registry

Note: In addition to importing business services directly from a UDDI registry, you can generate business services from Oracle Enterprise Repository. For more information see [Section 2.2.3, "Generating a Business Service from Oracle Enterprise Repository."](#)

To import business services from a UDDI registry:

1. Create a business service, as described in [Section 2.2.1, "Creating Business Services."](#)
2. In the New Business Service wizard Business Service General Configuration Page, select **WSDL Web Service**. then click **Browse**.
3. In the Select a WSDL dialog, click **Consume**.
4. In the Service Consumption dialog, select **UDDI**.

2.9 Working with Split-Join

This section provides instructions for creating and configuring Split-Joins. Following are the primary topics in this section:

- [Section 2.9.1, "Introduction to Split-Join"](#)
- [Section 2.9.2, "Designing a Split-Join"](#)
- [Section 2.9.3, "Designing a Static Split-Join"](#)
- [Section 2.9.4, "Designing a Dynamic Split-Join"](#)

2.9.1 Introduction to Split-Join

The Split-Join is a mediation pattern that can be used by a transport typed business service in an Oracle Service Bus message flow. Split-Join allows you to send message requests to multiple services concurrently, thus enhancing performance in comparison to sending them sequentially. Split-Join achieves this task by splitting an input message into individual messages, routing them concurrently to their destinations, and then aggregating the responses into one overall message.

You design a Split-Join in the Eclipse Split-Join editor, then export it to the Oracle Service Bus console for testing and production.

Note: In the Oracle Service Bus console, a Split-Join is associated with a business service using the Flow transport protocol. Therefore, the Split-Join has a .flow file name extension in Eclipse even though it is always referred to simply as a "Split-Join" in this document.

There are two types of Split-Join pattern: static Split-Join and dynamic Split-Join, as described in [Section 2.9.2, "Designing a Split-Join."](#)

For more information on invoking a business service from an Oracle Service Bus message flow, see "Proxy Services: Message Flow" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBAG1102>.

2.9.1.1 Using Split-Join with Content in SOAP Headers

You can use Split-Join to enhance the performance of services that put message content in the SOAP header. Proxy services can pass SOAP headers into Split-Joins, and Split-Joins can pass SOAP headers to proxy and business services as an invocation or response.

To enable this capability, you must declare the header parts along with the body parts in a single request/response message in the Split-Join WSDL and in the WSDL of the proxy or business services invoked by the Split-Join. With the message parts declared in the WSDLs, SOAP header content is available to Split-Joins in the request/response message variables.

Following is an example of the message and binding definitions in the WSDL.

Message

```
<wsdl:message name="retrieveCustomerOverviewByIdRequestMessage">
  <wsdl:part name="retrieveCustomerOverviewByIdRequest"
    element="co:retrieveCustomerOverviewByIdRequest"/>
  <wsdl:part name="serviceContext" element="sc:serviceContext"/>
</wsdl:message>
```

Binding

```
<wsdl:input>
<soap:body use="literal" parts="retrieveCustomerOverviewByIdRequest"/>
```

```
<soap:header message="tns:retrieveCustomerOverviewByIdRequestMessage"
  part="serviceContext" use="literal"/>
</wsdl:input>
```

2.9.1.2 Transaction Support

Split-Joins provide support for propagating transactions. Many Split-Join operations provide an option for setting a specific quality of service (QoS) values, which control transaction support. The QoS value of **Exactly Once** on a Split-Join operation ensures the operation executes in the context of a transaction if one exists.

Setting QoS on individual operations gives you the flexibility to execute multiple operations in the context of a transaction and execute other operations outside of a transaction in a single Split-Join. Operations set with a QoS of **Exactly Once** are executed in the transaction. Operations set with a QoS of **Best Effort** do not execute in the context of a transaction.

Split-Joins do not handle transaction rollback in the case of exceptions. It is the responsibility service component that called the Split-Join to handle transaction exceptions and rollback.

The following Split-Join operations support transaction propagation:

- Invoke Service
- Assign
- Delete
- Insert
- Java Callout
- Replace

2.9.1.3 Security with Split-Joins

Split-Joins do not enforce security policies, which means you cannot create a Split-Join with a WSDL that includes policies, and from a Split-Join you cannot call a WSDL-based business service that contains WSDL policies.

To ensure security enforcement when using Split-Joins, use proxy services to handle security enforcement in the following ways:

- Use the inbound proxy, which invokes the Split-Join, to enforce policies.
- If the Split-Join needs to invoke a WSDL business service that contains policies, have the Split-Join call a local proxy (configured without the security policies), which in turn invokes the business service with the required policies.

2.9.2 Designing a Split-Join

There are two Split-Join patterns, the Static Split-Join and the Dynamic Split Join.

The Static Split-Join can be used to create a fixed number of message requests (as opposed to an unknown number). For instance, a customer places an order for a cable package that includes three separate services: internet service, TV service, and telephone service. In the Static use case, you could execute all three requests in separate parallel branches to improve performance time over the standard sequential execution.

The Dynamic Split-Join can be used to create a variable number of message requests. For instance, a retailer places a batch order containing a variable number of individual

purchase orders. In the Dynamic use case, you could parse the batch order and create a separate message request for each purchase. Like the Static use case, these messages can then be executed in parallel for improved performance.

2.9.2.1 Initial Setup

Split-Joins potentially include the following tasks as part of their initial setup:

2.9.2.1.1 Creating/Importing a WSDL Containing the Base Operation Every Split-Join is based upon a WSDL operation. When you first create a Split-Join, you will be asked to browse to the appropriate WSDL file and to select this operation as part of the creation process. You can create this WSDL file in Eclipse.

2.9.2.1.2 Creating/Importing a Business Service to Use the Split-Join Every Split-Join will be used by a transport typed business service, which, in turn, is invoked by a proxy service. You cannot export or test your Split-Join until you have created this business service. If it already exists, you can import it into Eclipse, or, if it does not exist, you can create it in Eclipse or the Oracle Service Bus console. If you want to get started on your Split-Join before you create the business service, you can generate the business service automatically after you create the Split-Join.

2.9.3 Designing a Static Split-Join

Suppose you want to design a new Split-Join called *Service Availability* that handles orders for a telco's cable service package including TV, phone, and internet service. The idea is for the Split-Join to receive an incoming package order and to reply with an order acknowledgement for each type of service. In this case, *Service Availability* is designed as a *Static* Split-Join because there are three message requests per order, one for each type of service. In this particular example the customer requests only TV and DSL service.

Creating the *Service Availability* Split-Join may include the following tasks:

[Section 2.9.3.1, "1. Creating a New Split-Join"](#)

[Section 2.9.3.2, "2. Adding an Assign"](#)

[Section 2.9.3.3, "3. Adding a Parallel Node"](#)

[Section 2.9.3.4, "4. Adding an Assign for Each Branch"](#)

[Section 2.9.3.5, "5. Adding an Invoke Service"](#)

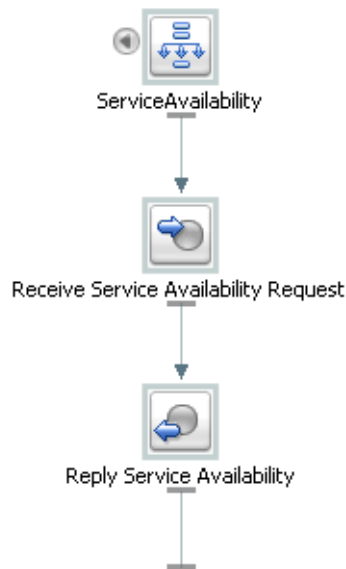
[Section 2.9.3.6, "6. Adding an Assign for Each Branch"](#)

[Section 2.9.3.7, "7. Exporting and Testing the Split-Join"](#)

2.9.3.1 1. Creating a New Split-Join

Create a new Split Join based on the WSDL operation you want to use for placing the order. In this case the WSDL operation we want is called "telecom."

After you select the WSDL operation, a skeleton of the newly created Split-Join appears in the Split-Join editor, as shown in the following figure. It consists of a Start Node, a Receive, a Reply. The labels are then edited in the general properties tab to better reflect the specific function of each node in this particular Split-Join.

Figure 2–1 New Split-Join

The Start Node contains both a Request Variable and a Response Variable that have been determined by the WSDL operation initially selected. The Receive receives the incoming request message (in this case for the three or fewer different kinds of cable service), and the Reply generates a response message and sends it back to the client.

Note: The Receive node requires no further configuration. Similarly the Reply requires no further configuration, unless to generate an error fault, which is not the case in this scenario (see [Section 2.9.15, "Configuring a Reply"](#) for more information on generating faults).

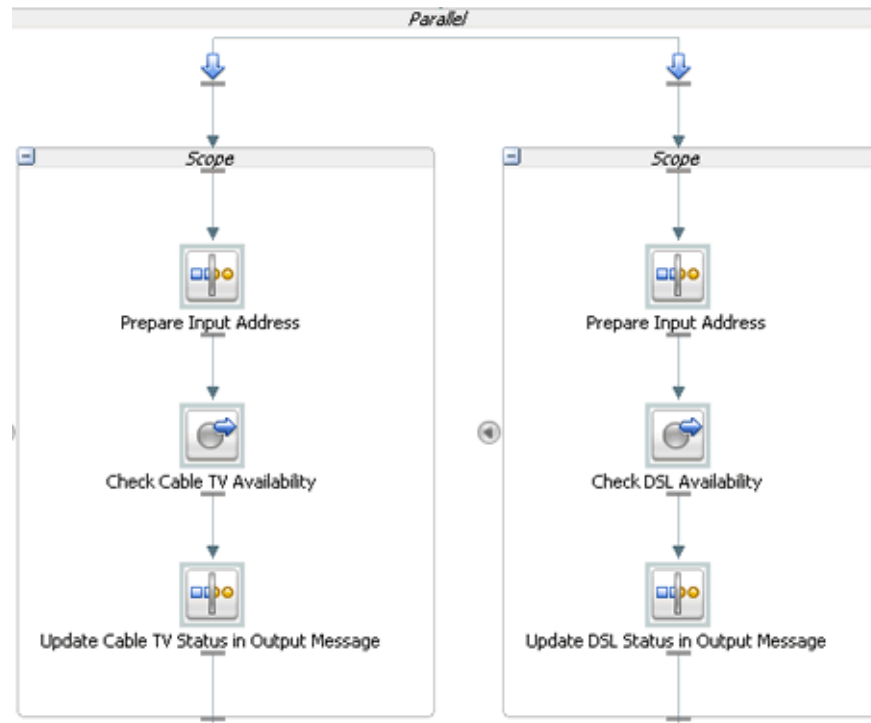
2.9.3.2 2. Adding an Assign

The first Assign, *Prepare Output Message*, contains an Assign operation that prepares the Response Variable in a form such that the later nodes can work on the data within it (that is, Copy/Insert/Assign/Replace/Delete/Java Callout/Log the data). This output message is relayed to the final Reply node in the Split-Join and, in turn, returned to the original client.

2.9.3.3 3. Adding a Parallel Node

The Parallel Node contains two main branches, one to check cable TV availability and one to check DSL availability. Each branch is composed of a number of actions, the sequence and general configuration being the same for both branches.

Figure 2–2 Parallel Node



2.9.3.4 4. Adding an Assign for Each Branch

The first Assign in each Parallel Branch, *Prepare Input Address*, copies the incoming customer address data into a Variable that is referenced to check the availability of the service at that location. The Assigns are the same for each branch and would be for additional branches as well.

2.9.3.5 5. Adding an Invoke Service

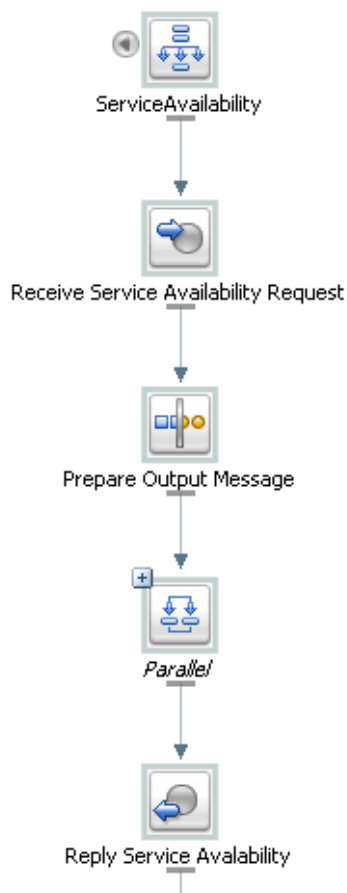
An External Service is then invoked to check whether the requested service type is available at the customer's location. Each branch contains one Invoke Service, *Check Cable TV Availability* and *Check DSL Availability*. Each invocation calls an External Service, which compares the customer's address (stored in the Variable initialized in the previous step) to the availability of the service at that location. The result is then stored in an output Variable that is passed on to the final Assign in the Branch below.

2.9.3.6 6. Adding an Assign for Each Branch

The final two Assigns, *Update Cable TV Status in Output Message* and *Update DSL Status in Output Message*, take the results of the external service invocations and put them into the output message using a Replace operation. The aggregated response are then sent to the original client in the final Reply node, which requires no further configuration.

2.9.3.7 7. Exporting and Testing the Split-Join

After you design the Split-Join, you can export it to the Oracle Service Bus console for testing and production.

Figure 2–3 Completed Split-Join Ready for Testing**Related Topics**

- [Section 2.9.5, "Creating a New Split-Join"](#)
- [Section 2.9.6, "Configuring the Start Node"](#)
- [Section 2.9.8, "Creating an Assign"](#)
- [Section 2.9.9, "Invoking a Service"](#)
- [Section 2.9.10, "Creating a Parallel"](#)
- [Section 2.9.17, "Exporting and Testing a Split-Join"](#)

2.9.4 Designing a Dynamic Split-Join

Suppose that you want to design a Split-Join that handles a batch order from a retailer containing a variable number of individual purchase orders (as opposed to a fixed number of orders). The idea is for the Split-Join to receive the batch order and to reply with an order acknowledgement for each order within. This would be a *Dynamic* Split-Join because the number of individual purchase order requests is variable and unknown at design time.

Creating this Split-Join may include the following tasks:

[Section 2.9.4.1, "1. Creating a New Split-Join"](#)

[Section 2.9.4.2, "2. Adding an Assign"](#)

[Section 2.9.4.3, "3. Adding a For Each"](#)

[Section 2.9.4.4, "4. Adding an Assign"](#)

[Section 2.9.4.5, "5. Adding an Invoke Service"](#)

[Section 2.9.4.6, "6. Adding an Assign"](#)

[Section 2.9.4.7, "7. Adding an Error Handler"](#)

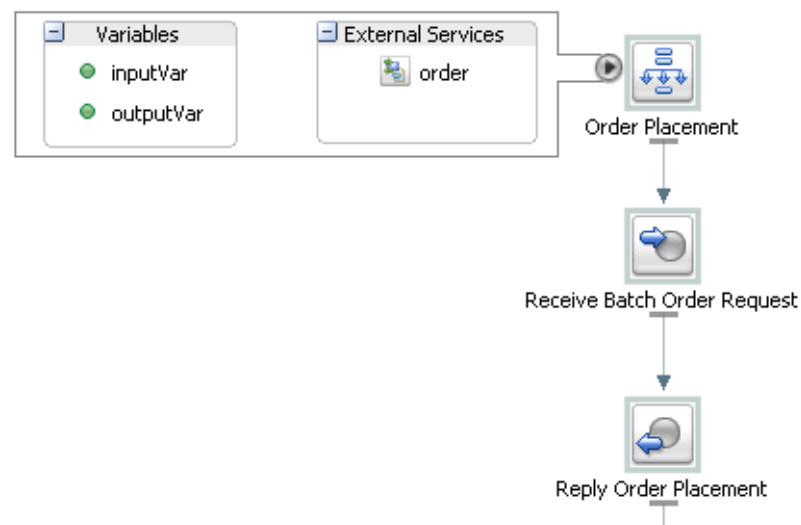
[Section 2.9.4.8, "8. Exporting and Testing the Split-Join"](#)

2.9.4.1 1. Creating a New Split-Join

Create a new Split Join based off of the WSDL operation you want to use for placing the order. In this case the WSDL operation we want is called "batchOrders."

After the operation is selected, a skeleton of the newly created Split-Join appears in the Split-Join editor consisting of a Start Node, a Receive, a Reply. The labels are then edited in the general properties tab to better reflect the specific function of each node in this particular Split-Join.

Figure 2-4 *New Split-Join With Edited Labels*



The Start Node, *Order Placement*, contains both a request variable, *inputVar*, and an response variable, *outputVar*. The Receive, *Receive Batch Order Request*, will initialize the contents of the Request Variable (in this case purchase orders), and the Reply, *Reply Order Placement*, will send a response, based on the order acknowledgements aggregated into the Response Variable, back to the client. In this example *Order Placement* also contains a callout to an External Service, "Order" that will be invoked to approve individual orders.

Note: The Receive node requires no further configuration. Similarly, the Reply requires no further configuration, unless you would like to generate an error fault—which is not the case in this scenario (see [Section 2.9.15, "Configuring a Reply"](#) for more information on generating faults).

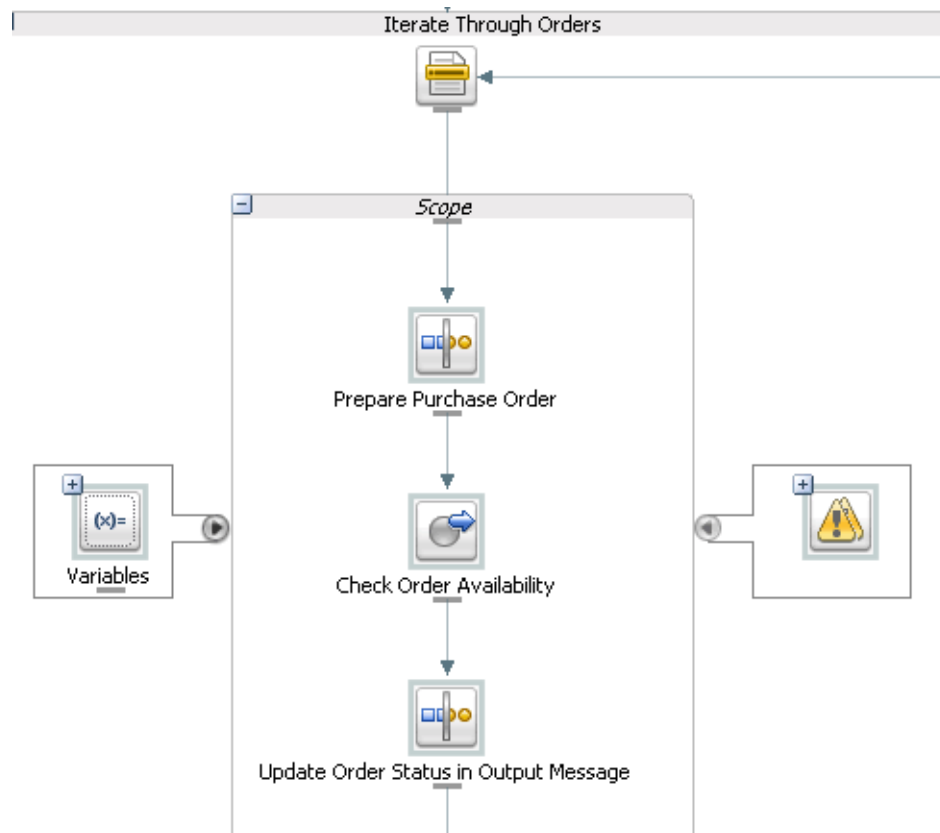
2.9.4.2 2. Adding an Assign

The first Assign, *Prepare Output Message*, contains an Assign operation that prepares the response variable (here labeled an "Output Message" for readability) in a form such that the later nodes can work on the data captured within it (that is, Copy/Insert/Assign/Replace/Delete into the Variable). In this case, that data would consist of order acknowledgments and/or errors. This Output Message is relayed to the final Reply node in the Split-Join and, in turn, returned to the original client.

2.9.4.3 3. Adding a For Each

The For Each, *Iterate Through Orders*, contains logic that will parse through each order in the batch, send it to an external proxy for approval, and capture an order acknowledgment in response. If there is a problem with an order, an error is sent from the invoked proxy and captured in the Error Handler. The following figure depicts the entire scope of the For Each logic.

Figure 2-5 For Each Node Labeled "Iterate Through Orders"



2.9.4.4 4. Adding an Assign

The Assign, *Prepare Purchase Order*, copies the incoming purchase order requests into a variable that is referenced to check approval of the order in the next step.

2.9.4.5 5. Adding an Invoke Service

An external service, *Check Order Availability*, is then invoked to approve each individual purchase order. If the order is accepted, the service responds with an order acknowledgment. If the order is not accepted, the service responds with an error. The

result is then stored in an output variable that is passed on to the final Assign in the next step.

2.9.4.6 6. Adding an Assign

The final Assign, *Update Order Status in Output Message*, takes the results of the external service invocation and copies them into the output message using an Insert operation. The aggregated response is then sent to the original client in the final Reply node, which requires no further configuration.

2.9.4.7 7. Adding an Error Handler

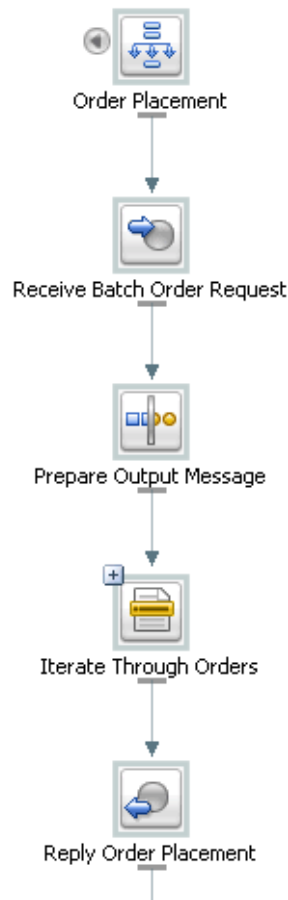
The Error Handler captures any Errors returned by the invoked service. It takes these errors and inserts them into the output message using an Assign operation.

Figure 2-6 Error Handler



2.9.4.8 8. Exporting and Testing the Split-Join

After you design the Split-Join, you can export it to the Oracle Service Bus console for testing and production.

Figure 2-7 Completed Split-Join Ready for Testing**Related Topics**

- [Section 2.9.5, "Creating a New Split-Join"](#)
- [Section 2.9.6, "Configuring the Start Node"](#)
- [Section 2.9.13, "Creating an Error Handler"](#)
- [Section 2.9.10, "Creating a Parallel"](#)
- [Section 2.9.15, "Configuring a Reply"](#)
- [Section 2.9.17, "Exporting and Testing a Split-Join"](#)

2.9.5 Creating a New Split-Join

In order to create a new Split/Join, you must have access to a WSDL containing the operation upon which to base the Split-Join. The Split Join must be created in an existing Oracle Service Bus project within an existing Oracle Service Bus configuration project.

To create a new Split-Join:

1. In the Oracle Service Bus perspective, select **File > New > Split-Join**. This opens the New Split-Join Wizard.
2. In the New Split-Join Wizard, type or select an Oracle Service Bus project location and enter a file name for the new Split-Join. When you have finished, click Next.

3. In the next screen, you must select a binding and then an operation on which to implement the Split-Join. There are two ways to make your selection:
 - a. Choose your operation from one of the WSDLs displayed in the Select Operation tree. All of the WSDLs in your current Oracle Service Bus configuration project are available.
 - b. Import your WSDL into the Oracle Service Bus configuration project using the Consume button. Consumption imports a new WSDL into your configuration from an outside source, as described in the following step.
4. If you choose to consume the base WSDL, go through the following steps:
 - a. Click **Consume**.
 - b. Browse for the location, or "Artifact Folder," wherein you wish to generate the consumed WSDL. The default artifact folder is your current Oracle Service Bus project.
 - c. If you want to overwrite existing local files, select the checkbox.
 - d. Choose the service resource in which the WSDL to be consumed resides: Enterprise Repository, File System, UDDI, URI, or Workspace.
 - e. Select The WSDL that you want to consume from that Service Resource. After you have made your selection the workspace will rebuild momentarily and the Service Consumption Status dialog will appear depicting the status of your consume. If it was successful, click **OK** to close the dialog.
 - f. The consumed WSDL is now in your Oracle Service Bus configuration project, and you can select an operation from it upon which to base your Split-Join.
5. Click **Finish**.

A basic Split-Join is created and visually represented as a diagram in the Design View. By default, it consists of a Start Node, a Receive, and a Reply. The Start Node contains the Request and Response Variables introspected from the WSDL operation. The Receive is used to receive incoming request messages. The Reply is used to send response messages.

Related Topics

- [Section 2.9.6, "Configuring the Start Node"](#)
- [Section 2.9.7, "Configuring a Receive"](#)
- [Section 2.9.10, "Creating a Parallel"](#)
- [Section 4.22.31, "Split-Join Wizard - New Split-Join"](#)

2.9.6 Configuring the Start Node

The Start Node is generated automatically whenever you create a new Split-Join. It is the starting point from which all the other nodes proceed. Configuring a Start Node can include the following tasks:

- Add General Information
- Define Global Variables
- View External Services

Related Topics

- [Section 2.9.9, "Invoking a Service"](#)

- [Section 4.22.2, "Global / Start Node Properties"](#)

2.9.6.1 Adding General Information

General information is useful for making a node more legible. It includes the ability to add a unique identifier, or Label, to the node and to supplement it with notes, or Documentation. General information is optional.

1. To add a Label to a node, select the General tab in the Properties view and enter a unique, identifying string in the Label field. The Label that you enter appears underneath the node in the Split-Join editor.
2. In the Documentation field enter any notes that you think are important.

2.9.6.2 Defining Global Variables

Variables in the Start Node store data that can be referenced globally, that is by any node in the Split-Join. By default, every Start Node is assigned both a request and a response variable when the Split-Join is initially created. From the Start Node, you can either create a new global variable or edit an existing global variable.

For information on the relationship between global and local variables, see [Section 2.9.16.1, "Scope and Variables."](#)

To create a new global variable

1. Right-click the Start Node and select **Create Variable** to open the Create Variable Dialog.
2. Enter a name for the variable.
3. Select the Variable Type (Builtin, Schema, or Message).
4. Choose a Variable.

Note: You may need to drill down into the hierarchy to select a Schema or Message Type variable.

5. Click **OK**.

If it is not already open, expand the content area to the left by clicking the arrow to the left of the Start Node icon. The newly created variable appears in the Variables field along with any other global variables. To view the details of any variable, simply select it and its structure will appear in the Properties view.

To edit an existing global Variable

1. Open the Edit Variable Dialog in one of the following ways:
 - Right-click the selected variable and select **Edit Variable** from the context menu.
 - Select a variable and click **Edit** in the variable Properties view.
2. Enter a name for the Variable.
3. Select the Variable Type: Builtin, Schema, or Message.
4. Choose a variable.

Note: You may need to drill down into the hierarchy to select a Schema or Message Type variable.

5. Click OK.

If it is not already open, expand the content area to the left of the Start Node icon. The newly created variable appears in the Variables field along with any other global variables. To view the details of any variable, simply select it and its structure will appear in the Properties view.

2.9.6.3 Viewing External Services

The External Services listed in the Start Node are those invoked outside of the context of the Split-Join. They are specified in an Invoke Service but listed here for convenience.

To view External Services, expand the content area to the left of the Start Node by clicking the arrow to the left of the Start Node icon. When an External Service is selected, a dashed blue arrow appears pointing to the Invoke Service associated with the service, and the service's location appears in the Properties view.

2.9.7 Configuring a Receive

A Receive is generated automatically whenever you create a new Split-Join. The purpose of the Receive is to place incoming request data in a variable and make the contents available for later nodes to use. Configuring a Receive can include the following tasks:

- View the Operation
- Define the Receive Variable
- Add General Information

Related Topics

[Section 4.22.24, "Receive Properties"](#)

2.9.7.1 Viewing the Operation

The Operation is based upon the initial WSDL selection for the overall Split-Join. It is displayed in the Properties View for reference.

2.9.7.2 Defining the Receive Variable

You must define the Incoming Message Variable the Receive will initialize.

1. Select the Receive operation.
2. Create a new Message Variable (following steps).

Note: If there are no available Message Variables associated with the previously chosen Operation, you must create a new Message Variable.

To create a new Message Variable, select **Create Message Variable** from the **Message Variable** menu, enter a variable name in the Create Variable dialog, and click **OK**.

Note that Message Type Namespace and Message Type are displayed automatically on the Properties page once the variable is defined.

2.9.7.3 Adding General Information

General information is useful for making a node more legible. It includes the ability to add a unique identifier, or Label, to the node and to supplement it with notes, or Documentation. General information is optional.

1. To add a Label to a node, select the General tab in the Properties view and enter a unique, identifying string in the Label field. The Label that you enter appears underneath the node in the Split-Join editor.
2. In the Documentation field enter any notes that you think are important.

2.9.8 Creating an Assign

The Assign is used for data-manipulation including initializing and updating a variable. It is composed of a set of one or more operations that can be added from the Assign toolbar. Configuring an Assign can include the following tasks:

- Add and Configure Assign Operations
- Add General Information

Related Topics

[Section 4.22.17, "Assign Properties"](#)

2.9.8.1 Adding and Configuring Assign Operations

Assign operations include Assign, Copy, Delete, Insert, Java Callout, Log, and Replace. Every Assign is composed of one or more of these operations, which you can add to the Assign using the Design Palette view.

Note: The Assign operations in the Split-Join editor are essentially the same as the corresponding actions in the Eclipse Message Flow editor. However, one important difference is that when you are using the XQuery\XSLT or XPath Editors to edit expressions in the Split-Join context, only variables and namespaces internal to the Split-Join are available.

A brief explanation of each operation follows:

- Assign the result of an XQuery Expression to a Variable.
For more information on configuring the Assign, see [Section 4.22.17, "Assign Properties"](#).
- Copy the information specified by an XPath expression from a source document to a destination document. (This operation is unique to Split-Join and has no corresponding Action in the Eclipse Message Flow editor, as described in [Section 2.9.8.1.2, "Adding a Copy Operation"](#)).
- Delete a set of nodes specified by an XPath expression.

Note: Unlike the Oracle Service Bus delete, only an XPath expression may be deleted in a Split-Join, not the entire variable.

For more information on configuring the Delete operation, see [Section 4.22.19, "Delete Properties."](#)

- Insert the result of an XQuery Expression at an identified place relative to nodes selected by an XPath expression.

For more information on configuring the Insert operation, see [Section 4.22.20, "Insert Properties."](#)

- **Java Callout** lets you invoke a Java method for processing such as validation, transformation, and logging.

For more information on configuring the Java Callout operation, see [Section 4.22.21, "Java Callout Properties."](#)

- **Log** lets you log Split-Join data at a specified severity to the server log file.

For more information on configuring the Log operation, see [Section 4.22.22, "Log Properties."](#)

- **Replace** a node or the contents of a node specified by an XPath expression.

For more information on configuring the Replace operation, see [Section 4.22.23, "Replace Properties."](#)

2.9.8.1.1 Adding an Operation to an Assign Adding an operation to the Assign involves the following steps:

1. Drag the operation from the Design Palette into the Assign node in the Split-Join editor.
2. Configure the operation in the Properties view.
3. Save the Split-Join.

You can edit an operation by selecting it and modifying the properties in the Properties view.

2.9.8.1.2 Adding a Copy Operation The Copy operation lets you copy the information specified by an XPath expression from a source document to a destination document. It is an operation unique to the Split-Join editor. Adding a Copy operation to the Assign involves the following steps:

1. Add a Copy to the Assign from the Design Palette.
2. In the Properties view, select a **Copy From** type and a **Copy To** type.
3. If the type is an expression, enter the expression manually or click **Browse** to launch the XQuery Expression Builder.
4. If the type is a variable, drill down to and select the desired element. The resulting query will be displayed in the Query field below.
5. If the Copy From type is a Literal, enter the literal in the text field.
6. If the Copy From type is an XML fragment, enter [or paste] the fragment in the text field.

2.9.8.2 Adding General Information

General information is useful for making a node more legible. It includes the ability to add a unique identifier, or Label, to the node and to supplement it with notes, or Documentation. General information is optional.

1. To add a Label to a node, select the General tab in the Properties view and enter a unique, identifying string in the Label field. The Label that you enter appears underneath the node in the Split-Join editor.

2. In the Documentation field enter any notes that you think are important.

2.9.9 Invoking a Service

The Invoke Service is used to invoke external, WSDL-based business services, WSDL-based proxy services, and Split-Joins. Configuring an Invoke Service can include the following tasks:

- Select an Operation
- Define Input and Output Variables
- Add General Information

Related Topics

- [Section 2.9.6, "Configuring the Start Node"](#)
- [Section 2.9.16, "About Scope"](#)
- [Section 4.22.5, "Invoke Service Properties"](#)

2.9.9.1 Selecting an Operation

An operation must be selected upon which to base the Invoke Service. You must select this operation before you can configure Input and Output variables. To select an operation:

1. Add an Invoke Service operation to the Split-Join from the Design Palette.
2. From the Operations tab in the Properties view, click **Browse** to launch the Service Browser.
3. In the Service Browser, drill down to the desired service and select a Binding, then an operation.
4. Click **OK**. The selected operation and its Service Location appear in the Properties view.

Note: Clicking a Service Location in the Properties view will open the external service file.

2.9.9.2 Defining Input and Output Variables

An Invoke Service requires both an Input Variable and an Output Variable, unless it is a one-way invocation. The procedure to configure these variables is essentially the same. Either type of variable can be global (that is, available within the entire Split-Join) or local (that is, available within a particular context Scope.) To define either an Input or Output variable:

In the Input Variable and Output Variable tabs in the Properties view, define the Message Variable for each. This can be done in two ways:

- Select a pre-existing variable from the Message Variable menu.
- Create a new Message Variable (following steps).

Note: If there are no available Message Variables associated with the previously chosen operation, you must create a new Message Variable.

To create a new Message Variable

1. Select **Create Message Variable** from the Message Variable menu. The Create Message Variable Dialog appears.
2. Provide a name for the variable.
3. Make the variable either global or local. Global variables are accessible within the entire Split-Join, whereas local variables are restricted to the current Scope.

Message Type Namespace and Message Type are displayed automatically on the Properties view once a variable is defined.

2.9.9.3 Adding General Information

General information is useful for making a node more legible. It includes the ability to add a unique identifier, or Label, to the node and to supplement it with notes, or Documentation. General information is optional.

1. To add a Label to a node, select the General tab in the Properties view and enter a unique, identifying string in the Label field. The Label that you enter appears underneath the node in the Split-Join editor.
2. In the Documentation field enter any notes that you think are important.

2.9.10 Creating a Parallel

The Parallel creates a fixed number of configured parallel branches. Each branch has its own Scope which in turn can contain any number of nodes. Configuring a Parallel can include the following tasks:

- Add Nodes
- Add General Information

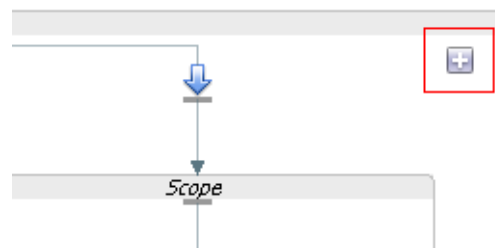
Related Topics

- [Section 2.9.16, "About Scope"](#)
- [Section 4.22.10, "Parallel Properties"](#)

2.9.10.1 Adding Nodes

The Parallel is essentially a placeholder for a fixed number of processing branches, each with its own scope. Two branches are automatically generated by default. An individual scope may contain unique processing logic according to your construction; simply drag the appropriate nodes into the Scope. You may add additional branches with the Add Scope button.

Figure 2-8 Add Scope Button



2.9.10.2 Adding General Information

General information is useful for making a node more legible. It includes the ability to add a unique identifier, or Label, to the node and to supplement it with notes, or Documentation. General information is optional.

1. To add a Label to a node, select the General tab in the Properties view and enter a unique, identifying string in the Label field. The Label that you enter appears underneath the node in the Split-Join editor.
2. In the Documentation field enter any notes that you think are important.

2.9.11 Creating a For Each

The For Each is used to create conditional logic for iterating through a variable number of requests. It is primarily used to create dynamic Split-Joins. Configuring a For Each can include the following tasks:

- Define the For Each Logic
- Add General Information

Related Topics

- [Section 2.9.9, "Invoking a Service"](#)
- [Section 4.22.7, "For Each Properties"](#)

2.9.11.1 Defining the For Each Logic

To define the For Each logic:

1. Add a For Each node to the Split-Join, and select it.
2. In the Properties view, select the **Counter Variables** tab, and set the Parallel property to yes or no. If you choose yes, individual branches will be processed in parallel. If you select no, they are processed sequentially.
3. Define the Counter Variable Name by clicking the **Counter Name** link.
4. Enter the **Start Counter Value**. If necessary, use the browse button to create a new value in the XPath Expression Builder.

Note: The lowest possible starting counter value is "1."

5. Enter the **Final Counter Value**. If necessary, use the browse button to create a new value in the XPath Expression Builder.

Note: The lowest possible starting counter value is "1."

2.9.11.2 Adding General Information

General information is useful for making a node more legible. It includes the ability to add a unique identifier, or Label, to the node and to supplement it with notes, or Documentation. General information is optional.

1. To add a Label to a node, select the General tab in the Properties view and enter a unique, identifying string in the Label field. The Label that you enter appears underneath the node in the Split-Join editor.
2. In the Documentation field enter any notes that you think are important.

2.9.12 Creating an If Activity

The If Activity is used to provide conditional logic within a Split-Join. It is composed of a number of nodes that determine the behavior for the overall If activity. Each node must be individually configured. When you create an If activity, an If and an Else are automatically generated within it. You can add an unlimited number of Else If nodes with the Add Else If button.

Figure 2–9 Add Else If Button



Configuring an If Activity can include the following tasks:

- Configure the If
- Add and Configure Else If
- Configure the Else

Related Topics

- [Section 2.9.16, "About Scope"](#)
- [Section 2.9.4, "Designing a Dynamic Split-Join"](#)
- [Section 4.22.8, "If Properties"](#)
- [Section 4.22.12, "Repeat Until Properties"](#)
- [Section 4.22.15, "While Properties"](#)

2.9.12.1 Configuring the If

The If provides a unit of conditional logic within the overall If activity. It is automatically generated when you create an If activity. Configuring an If can include the following tasks:

- Write the logic of the condition
- Add resulting nodes
- Add General Information

Related Topics

- [Section 4.22.9, "If and Else If Properties"](#)
- [Section 4.22.12, "Repeat Until Properties"](#)
- [Section 4.22.15, "While Properties"](#)

2.9.12.1.1 Writing the logic of the condition The If Activity executes conditional logic defined by an XPath expression. Enter this condition in the Condition text field of the Condition tab, or click the browse button to launch and write the expression in the expression builder.

2.9.12.1.2 Adding resulting nodes If the condition in the If logic is met, a subsequent node or string of nodes will result. Add and configure any resulting nodes by dragging them in sequential order to a drop point beneath the If icon.

2.9.12.1.3 Adding General Information General information is useful for making a node more legible. It includes the ability to add a unique identifier, or Label, to the node and to supplement it with notes, or Documentation. General information is optional.

1. To add a Label to a node, select the General tab in the Properties view and enter a unique, identifying string in the Label field. The Label that you enter appears underneath the node in the Split-Join editor.
2. In the Documentation field enter any notes that you think are important.

2.9.12.2 Adding and Configuring Else If

The Else If is used to provide additional logic within the context of an overall If. You can add an Else If every time you press the "Add Else If" button.

Figure 2–10 Add Else If Button



Configuring an Else If can include the following tasks:

- Write the Logic of the Condition
- Add Resulting Nodes
- Add General Information

Related Topics

- [Section 4.22.9, "If and Else If Properties"](#)
- [Section 4.22.12, "Repeat Until Properties"](#)
- [Section 4.22.15, "While Properties"](#)

2.9.12.2.1 Writing the Logic of the Condition The Else If uses conditional logic defined by an XPath expression. Enter this condition in the Condition text field of the Condition tab or click the browse button to launch and write the expression in the expression builder.

2.9.12.2.2 Adding Resulting Nodes If the condition in the Else If logic is met, a subsequent node or string of nodes will result. Add and configure any resulting nodes by dragging them in sequential order to a drop point beneath the Else If icon.

2.9.12.2.3 Adding General Information General information is useful for making a node more legible. It includes the ability to add a unique identifier, or Label, to the node and to supplement it with notes, or Documentation. General information is optional.

1. To add a Label to a node, select the General tab in the Properties view and enter a unique, identifying string in the Label field. The Label that you enter appears underneath the node in the Split-Join editor.
2. In the Documentation field enter any notes that you think are important.

2.9.12.3 Configuring the Else

The Else provides a final case of logic within the context of an overall If. It is automatically generated when an If is created. Configuring an Else can include the following tasks:

- Add Resulting Nodes
- Add General Information

Related Topics

- [Section 4.22.9, "If and Else If Properties"](#)
- [Section 4.22.12, "Repeat Until Properties"](#)
- [Section 4.22.15, "While Properties"](#)

2.9.12.3.1 Adding Resulting Nodes As the final case in the If's logic, the Else requires no conditions to be met in order to execute. It will automatically execute resulting activities when invoked. Add and configure any resulting nodes by dragging them in sequential order to a drop point beneath the Else icon.

2.9.12.4 Adding General Information

General information is useful for making a node more legible. It includes the ability to add a unique identifier, or Label, to the node and to supplement it with notes, or Documentation. General information is optional.

1. To add a Label to a node, select the General tab in the Properties view and enter a unique, identifying string in the Label field. The Label that you enter appears underneath the node in the Split-Join editor.
2. In the Documentation field enter any notes that you think are important.

2.9.13 Creating an Error Handler

The Error Handler receives and handles errors. If it is attached to a Start Node, it is a "global" Error Handler and serves as a catch-all for the output of all local Raise Error nodes. If it is attached to a Scope, it only handles errors raised locally. To create an Error Handler:

1. Select the start node or Scope node to which the Error Handler will be added.
2. Right-click the selected icon and select **Add Catch** or **Add CatchAll**.
3. If you are invoking a SOAP Fault, in the Catch All tab of the Properties View, select **SOAP Fault Variable Name** to define a SOAP Fault variable associated with the Error Handler.

The basic Error Handler is now configured, but you may need to add additional Assign, If, and/or Reply nodes to it depending on whether you wish to execute logic upon the received faults before sending a response.

Related Topics

- [Section 2.9.8, "Creating an Assign"](#)

- [Section 2.9.12, "Creating an If Activity"](#)
- [Section 2.9.10, "Creating a Parallel"](#)
- [Section 2.9.16, "About Scope"](#)
- [Section 4.22.4, "Error Handler Properties"](#)

2.9.14 Creating a Raise Error

The Raise Error generates an error that causes the Split-Join to stop normal processing. If the error is not handled using an Error Handler, the Split-Join will terminate and a Fault will be sent to the Oracle Service Bus message flow. Configuring a Raise Error can optionally include documenting the nature of the error in the General Information tab.

You can also add a Re-Raise Error operation to an Error Handler. Configuration involves modifying Label and adding Documentation.

Related Topics

[Section 4.22.11, "Raise Error Properties"](#)

General information is useful for making a node more legible. It includes the ability to add a unique identifier, or Label, to the node and to supplement it with developer notes, or Documentation. General information is optional.

1. To add a Label to a node, open the Properties view and enter a unique, identifying string in the **Label** field. The Label that you enter appears underneath the node Icon in the Canvas area.
2. In the Documentation field enter any notes that you think are important.

2.9.15 Configuring a Reply

A global Reply is generated automatically whenever you create a new Split-Join. The purpose of the global Reply is to send a response back to the invoking Oracle Service Bus message flow. However, you may also create a Reply elsewhere in the Split-Join, including within Error Handlers. Configuring the Reply can include the following tasks:

- View the Operation
- Define the Reply Variable
- Add General Information

Related Topics

[Section 4.22.6, "Reply Properties"](#)

2.9.15.1 Viewing the Operation

The operation is based upon the initial WSDL selection for the overall Split-Join. It is displayed in the Properties view for reference.

2.9.15.2 Defining the Reply Variable

The Reply can either send a Response or a Fault back to the client depending on how you configure the variable. The Fault options available vary depending upon whether the Reply is global or local.

- A global Reply (that is, a Reply in a Split-Join outside of an Error Handler) can never have a SOAP Fault but can have a WSDL Fault. This is why the SOAP Fault option is always disabled.
- A local Reply (that is, a Reply attached to an Error Handler) can have either a WSDL Fault or a SOAP Fault. WSDL Faults will be available only if they were defined in the WSDL upon which the Split-Join is based. The SOAP Fault option will always be available provided one has been previously defined in the Error Handler.

Note: Switching back and forth between the Response and Fault buttons will clear either configuration. For instance, if you have previously selected "Propagate SOAP Fault" for Fault configuration and you then switch to the "Response" configuration, "Propagate SOAP Fault" will be deselected.

Given the available options as outlined above, select either a Response or a Fault for your Reply Variable.

If you select Response, you must define the Message Variable the Response will be assigned to. This can be done in two ways:

1. Select the Reply, and in the Properties view, select the Variable tab.
2. Select a pre-existing variable from the Message Variable menu.
3. Create a new Message Variable (following steps).

Note: If there are no available Message Variables associated with the previously chosen operation, you must create a new Message Variable.

To create a new Message Variable, select **Create Message Variable** from the Message Variable menu. The Create Message Variable Dialog appears:

1. Provide a name for the variable.
2. Click **OK**.

Note that Message Type Namespace and Message Type are displayed automatically in the Properties view once the variable is defined.

If you select **Fault**, you must specify either a **WSDL Fault** or propagate a **SOAP Fault**.

Note: In some circumstances, no Faults or only a SOAP Fault will be available. See previous notes.

If you select a WSDL Fault, you must specify the Fault by name and define the Message Variable that it will be assigned to.

1. Select **WSDL Fault Name** and choose a name from those available.

Note: There may only be one name available in which case no choice is necessary.

2. Define a Message Variable. This can be done in two ways:

- a. Select a pre-existing variable from the Message Variable menu.
- b. Create a new Message Variable (following steps).

Note: If there are no available Message Variables associated with the previously chosen operation, you *must* create a new Message Variable.

To create a new Message Variable, select **Create Message Variable** from the Message Variable menu. The Create Message Variable Dialog opens:

1. Provide a name for the variable.
2. Click **OK**.

Message Type Namespace and Message Type are displayed automatically on the properties page once a variable is defined.

If you select Propagate SOAP Fault, the SOAP Fault specified in the parent Error Handler will automatically be propagated in the Reply. There is nothing else to configure.

2.9.15.3 Adding General Information

General information is useful for making a node more legible. It includes the ability to add a unique identifier, or Label, to the node and to supplement it with notes, or Documentation. General information is optional.

1. To add a Label to a node, select the General tab in the Properties view and enter a unique, identifying string in the Label field. The Label that you enter appears underneath the node in the Split-Join editor.
2. In the Documentation field enter any notes that you think are important.

2.9.16 About Scope

A Scope is a container that groups various elements together. The container creates a context that influences the behavior of its enclosed elements. Local Variables and the Error Handler defined within the Scope are restricted to this context. However, some nodes within the scope may operate both locally (that is, within the Scope) and globally (that is, outside of the Scope.) For instance, an Invoke Service within a certain Scope might call upon an service external to the Scope's context.

2.9.16.1 Scope and Variables

Although variables are visible in the scope in which they are defined and in all scopes nested within that scope, a variable declared in an outer scope is hidden when you declare a variable with an identical name in an inner scope. For example, if you define variable myVar in an outer scope (So) and then define variable myVar again in an inner scope (Si) which is contained by scope So, then you can only access the myVar you defined in the inner scope Si. This myVar overrides the myVar you defined in scope So.

Related Topics

[Section 4.22.14, "Scope Properties"](#)

2.9.17 Exporting and Testing a Split-Join

You can export and test a Split-Join on an Oracle Service Bus server provided that it is associated with a transport typed business service. Exporting and testing a Split-Join can include the following tasks:

- Creating a Transport Typed Business Service
- Exporting the Split-Join Files
- Testing the Split-Join in the Test Console

2.9.17.1 Creating a Transport Typed Business Service

A Split-Join is used by a particular transport typed business service. If you do not have an appropriate business service, you must create one before you can export or test your Split-Join. There are two ways to create a business service:

1. Create the business service manually in Eclipse or the Oracle Service Bus console.
2. Generate the business service automatically from the Split-Join (.flow) menu:
 - a. Right click on the Split-Join (.flow) file in the Project Explorer to open the Split-Join menu.
 - b. Select **Oracle Service Bus**.
 - c. Select **Generate Business Service**.
 - d. Name and save the new service in a project.

After you create the business service, you can export the Split-Join provided that it has no errors.

Note: It is a helpful practice to place the associated business service in the same Oracle Service Bus project as the Split-Join. It can also be useful to give the business service the same name as the Split-Join so that they are easily correlated.

2.9.17.2 Exporting the Split-Join Files

Split-Joins without errors can be exported to an Oracle Service Bus server.

Note: Errors appear in the Problems view of the Split-Join editor. If you try to export a Split-Join with errors, the export fails.

There are three ways to export a Split-Join:

1. Export from the Business Service Menu
2. Auto-export
3. Manual export

2.9.17.2.1 Exporting from the Business Service Menu It is possible to export a Split-Join directly from the Business Service menu. However, because exporting by this method automatically launches the Oracle Service Bus Test Console, it is useful if you want to both export and test. Exporting from the Business Service menu involves the following steps:

1. In the Project Explorer, right click on the Business Service (.biz file) to be exported/tested.
2. Select **Run as**.
3. Select **Run on server**. The Run on Server Dialog opens.
4. Select an existing server or define a new one and go to the next step.
5. In the Add and Remove Projects window, the Oracle Service Bus project containing the business service and any other dependent files have been pre-selected for configuration/export. They can not be removed because the business service can not be tested without its dependent files. The entire Split-Join will therefore be exported.
6. Select **Finish**, and the Oracle Service Bus Test Console will launch. You can now test the business service.

2.9.17.2.2 Auto-export A Split-Join can be auto-exported to an Oracle Service Bus server. If you use this method, you must manually launch the Oracle Service Bus console in order to test the exported files. Auto-exporting involves the following steps:

1. Select **File > Export**.
2. Select **Oracle Service Bus**.
3. Select **Resources to Oracle Service Bus Server**. This will export the resources to the Oracle Service Bus server, but it will not launch the Oracle Service Bus Test Console. You must launch the Test Console manually within the Oracle Service Bus console application.

2.9.17.2.3 Manual export A Split-Join can be manually exported to an Oracle Service Bus server. If you use this method, you must manually launch the Oracle Service Bus console to test the exported files. Manually exporting involves the following steps:

1. Select **File > Export**.
2. Select **Oracle Service Bus**.
3. Select **Resources as Configuration JAR** and go to the next step.
4. In the Oracle Service Bus Configuration JAR Export window, configure the following options:
 - a. Select the Oracle Service Bus Configuration file containing the files to be exported.
 - b. Set the Export Level to **Project** or **Resource** depending upon whether you wish to export entire projects or individual files. The selection available in the tree below will change based upon the Export Level.
 - c. Select the projects and/or resources to be exported in the configuration JAR.
 - d. Select **Include Dependencies** if you want to export any file dependencies associated with the selected files.
 - e. Browse to a destination for the exported JAR file.
 - f. Click **Finish** to export the JAR file.
5. Import the JAR file via the Oracle Service Bus console.

Note: A quick way to access the Oracle Service Bus console from Eclipse is to right-click the server and select **Launch Service Bus Console**.

2.9.17.3 Testing the Split-Join in the Test Console

You can test a Split-Join by executing the business service that uses it in the Oracle Service Bus Test Console. This can either be done within the Split-Join editor or by exporting the Split-Join to an Oracle Service Bus server. To test the Split-Join within the IDE, you need to export the files using the menu for the business service that uses the Split-Join.

2.9.17.3.1 Exporting from the Business Service Menu You can export and test a Split-Join directly from the Business Service menu. If you use this method, the export happens in the background while the Oracle Service Bus Test Console launches. Exporting from the Business Service menu involves the following steps:

1. In the Project Explorer, right click on the Business Service (.biz file) to be exported/tested.
2. Select **Run as**.
3. Select **Run on server**. The Run on Server Dialog appears.
4. Select an existing server or define a new one and go to the next step.
5. In the Add and Remove Projects window, the Oracle Service Bus project containing the business service and any other dependent files have been pre-selected for configuration/export. The dependent files cannot be removed because the business service cannot be tested without its dependent files.

Click **Finish**, and the Oracle Service Bus Test Console will launch. You can now test the business service.

Note: Although only the Oracle Service Bus Test Console is displayed at this point, the entire Split-Join has been exported to the Oracle Service Bus server.

2.10 Using the Oracle Service Bus Debugger

Oracle Service Bus extends the Eclipse debugging framework to provide debugging functionality for proxy service message flows and Split-Joins.

The Oracle Service Bus Debugger can handle Java callouts and supports multi-threaded debugging on Split-Joins that use parallel processing. You can also perform debugging on remote servers.

You can use the Oracle Service Bus Debugger in two different ways:

- [Section 2.10.2, "Using Standard Debugging"](#) – Provides automated service debugging features for debugging on a local machine.
- [Section 2.10.3, "Using the Oracle Service Bus Debugger Launch Configuration"](#) – Provides more manual control of the Oracle Service Bus Debugger environment.

2.10.1 Enabling Debugging

Oracle Service Bus debugging is enabled automatically on a server running in development mode. When you create a new domain, the following entries are included in the <domain>/bin/setDomainEnv script:

```
set ALSB_DEBUG_FLAG=true
set ALSB_DEBUG_PORT=7453
```

If you want to turn Oracle Service Bus debugging functionality off, set `ALSB_DEBUG_FLAG=false`.

If you start the server in production mode, Oracle Service Bus debugging is automatically disabled.

2.10.2 Using Standard Debugging

To debug a proxy service or a Split-Join, you must execute it while in debug mode (unless you are using the debugger launch configuration described in [Section 2.10.3, "Using the Oracle Service Bus Debugger Launch Configuration"](#).)

To debug a proxy service or Split-Join:

1. Set breakpoints in your message flow or Split-Join to automatically pause the test execution at those points. Right-click an action in the flow editor and choose **Toggle Breakpoint**.
2. Start the server in debug mode. On server startup, Eclipse automatically switches to the Debug perspective, shown in [Figure 2–11](#).
3. With the proxy or business service file open and active, select **Run > Debug As > Debug on Server** in Eclipse.

In the Debug on Server window, select the server on which your Oracle Service Bus configuration is deployed, or is to be deployed, and complete the steps in the wizard.

The Debug As option automatically enables the Java debugger in order to handle Java callouts in services.

4. The Oracle Service Bus Test Console appears. Execute your test, looking at and interacting with the execution threads in the Debug view. As you move through the test, the debugger highlights the current stage of the test in the service's flow view, as shown in [Figure 2–11](#).

You are not required to use the Test Console to execute a service test and use the debugger. You can also execute your service in other ways, such as posting a JMS message or dropping a file in the directory of a file proxy service.

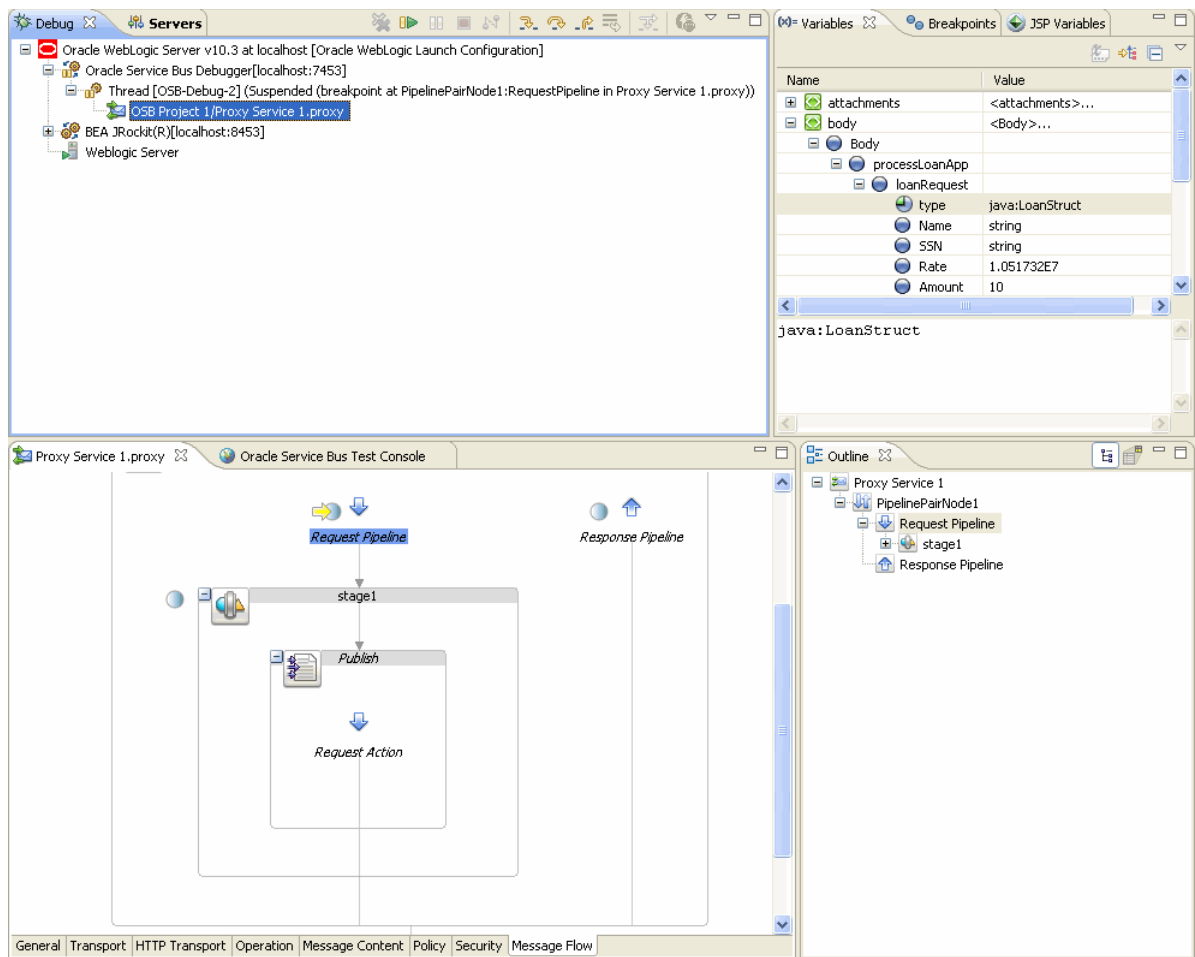
For debugging on remote servers, execute the service on the remote server and step through the test in Debug view. If the remote server is running in normal mode rather than debug mode, use the debugger launch configuration, as described in [Section 2.10.3, "Using the Oracle Service Bus Debugger Launch Configuration"](#).

See the [Section 2.10.2.1, "Debug Views"](#) section for descriptions of different debug views for Oracle Service Bus Debugger.

2.10.2.1 Debug Views

This section describes the different views you can use while debugging a service, illustrated in [Figure 2–11](#).

Figure 2–11 Debugging a Proxy Service



The Debug perspective provides the following key views for debugging a service:

- **Debug view** – The Debug view shows the Oracle Service Bus call stack, displaying the current execution thread. The Debug view also provides a toolbar that lets you resume the test from its current location or perform step actions (Step Into, Step Over, Step Return). As you step through a test, if you are testing a local service, the current operation is highlighted in the service’s flow view. Errors appear in the Console view.
- **Console view** – The Console view, which displays server messages, shows any run-time errors that occur as your service runs.
- **Variables view** – The Variables view shows the variable names and values in your service at each stage of the test. Variables are read-only while debugging. Select the call stack at a specific breakpoint to see the current variables and values.
- **Breakpoints view** – The Breakpoints view lists the breakpoints you have manually inserted into your service’s flow. The test stops at each breakpoint, letting you view the state of the service at that point. You can enable and disable breakpoints in the Breakpoints view. Disabling a breakpoint by deselecting it keeps the breakpoint in place, but the debugger ignores it.
- **Service editor, flow view** – As the test runs, if you are testing a local service, the test progress is graphically highlighted in the service’s flow view. The flow view

also shows the breakpoints you have set (by right-clicking actions in the service's flow view editor and choosing Toggle Breakpoint).

- **Oracle Service Bus Test Console** – The Test Console lets you execute local proxy and business services. For more detailed information on using the Test Console, see "Using the Test Console" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBAG357>.

2.10.2.2 Step Actions and Breakpoints

The Eclipse debugging framework lets you debug incrementally by performing step actions to debug code. See "Execution Control Commands" in the Eclipse help system (*Java Development User Guide*).

If you use step actions to debug, the Oracle Service Bus Debugger still stops at each breakpoint you have set, ignoring the current step action being performed.

2.10.3 Using the Oracle Service Bus Debugger Launch Configuration

If you want more manual control of the Oracle Service Bus Debugger environment, use the Oracle Service Bus Debugger launch configuration. Launch configurations are an Eclipse feature. The Oracle Service Bus Debugger launch configuration removes the automation of running the debugger with the Debug As option (Debug perspective launching, server restarting in debug mode, Java debugger starting), letting you connect and disconnect the debugger as needed on a server running in either normal or debug mode.

If you want to use the Java debugger in conjunction with the Oracle Service Bus Debugger to handle Java callouts, the server must be running in debug mode.

To use the Oracle Service Bus Debugger launch configuration:

1. Set breakpoints in your proxy service message flow or Split-Join.
2. Start the server in normal or debug mode. However, if you want to use the Java debugger to handle Java callouts from your flow, start the server in debug mode.
3. In the Eclipse menu, select **Run > Debug Configurations**.
4. Double-click **Oracle Service Bus Debugger**. A "New_configuration" sub-item appears. You need only perform this step once for each debug configuration you want to create.

The right pane displays the default server and port. Make sure the port matches the value of the ALSB_DEBUG_PORT in your domain's setDomainEnv script.

You can also rename the launch configuration in the right pane, as well as add the debug configuration to the Debug toolbar item as a shortcut.

If your services use Java callouts, enable the Java debugger by double-clicking **Remote Java Application** and configuring the "New_configuration."

5. With your launch configuration selected, click **Debug** in the Debug dialog. The debugger is connected. If you are using the Java debugger as well, select it and click Debug.
6. Open the Debug perspective or the debug views you want, such as Debug, Breakpoints, and Variables.
7. Run the service.
 - To execute the service in the Test Console, use **Run As > Run on Server**.

- You can also execute your service in other ways, such as through a custom test client.

8. Step through the test.

To disconnect the debugger, click the Disconnect icon in the Debug view. Reconnect the debugger in the Debug dialog by selecting the launch configuration and clicking Debug, or by creating a shortcut in the Debug toolbar item as described in the previous steps.

2.10.3.1 Remote Debugging

If you are debugging remote services, you can select a remote server in the server configuration window (for Debug As) or set the remote server and port in the launch configuration. After you connect the debugger to the remote server, execute the services on the remote server and step through the test in your local Debug perspective.

2.10.3.2 Debugging Oracle Service Bus Running Stand-Alone on a Managed Server

If you want to debug on an Oracle Service Bus instance that is running stand-alone on a managed server in a non-clustered environment, follow these steps:

1. In Eclipse, in Servers view, add a server for the domain to which the Oracle Service Bus managed server belongs.
2. Make sure that both the domain's admin server and the Oracle Service Bus managed server are running.

The admin server must be running in normal mode, not debug mode.

3. Follow the debug configuration and debug steps [Section 2.10.3, "Using the Oracle Service Bus Debugger Launch Configuration."](#)

If you set up a Remove Java Application configuration to test Java callouts, be sure to target the configuration to the managed server.

2.10.3.3 Server Sharing

If multiple users share a single server instance for debugging, only one user at a time can have the debugger connected. Other users trying to connect a debugger will get a connection refused error.

Transport Configuration

When you configure a business service or a proxy service, you must configure the transport used by the service. Each transport is configured on its own configuration page.

3.1 Protocol-Specific Transport Configuration Pages

Each transport available for business services and proxy services has its own configuration page. Oracle Service Bus provides the following transports:

- [Section 3.1.1, "BPEL-10g Transport Configuration Page \(Business Services\)"](#)
- [Section 3.1.2, "DSP Transport Configuration Page \(Business Services\)"](#)
- [Section 3.1.3, "EJB Transport Configuration Page \(Business Services\)"](#)
- [Section 3.1.4, "E-Mail Transport Configuration Page \(Business Services\)"](#)
- [Section 3.1.5, "E-Mail Transport Configuration Page \(Proxy Services\)"](#)
- [Section 3.1.6, "File Transport Configuration Page \(Business Services\)"](#)
- [Section 3.1.7, "File Transport Configuration Page \(Proxy Services\)"](#)
- [Section 3.1.8, "FTP Transport Configuration Page \(Business Services\)"](#)
- [Section 3.1.9, "FTP Transport Configuration Page \(Proxy Services\)"](#)
- [Section 3.1.10, "HTTP Transport Configuration Page \(Business Services\)"](#)
- [Section 3.1.11, "HTTP Transport Configuration Page \(Proxy Services\)"](#)
- [Section 3.1.12, "JCA Transport Configuration Page \(Proxy and Business Services\)"](#)
- [Section 3.1.13, "JEJB Transport Configuration Page \(Business Services\)"](#)
- [Section 3.1.14, "JEJB Transport Configuration Page \(Proxy Services\)"](#)
- [Section 3.1.15, "JMS Transport Configuration Page \(Business Services\)"](#)
- [Section 3.1.16, "JMS Transport Configuration Page \(Proxy Services\)"](#)
- [Section 3.1.17, "MQ Transport Configuration Page \(Business Services\)"](#)
- [Section 3.1.18, "MQ Transport Configuration Page \(Proxy Services\)"](#)
- [Section 3.1.19, "SB Transport Configuration Page \(Business Services\)"](#)
- [Section 3.1.20, "SB Transport Configuration Page \(Proxy Services\)"](#)
- [Section 3.1.21, "SFTP Transport Configuration Page \(Business Services\)"](#)
- [Section 3.1.22, "SFTP Transport Configuration Page \(Proxy Services\)"](#)

- [Section 3.1.23, "SOA-DIRECT Transport Configuration Page \(Business Services\)"](#)
- [Section 3.1.24, "Tuxedo Transport Configuration Page \(Business Services\)"](#)
- [Section 3.1.25, "Tuxedo Transport Configuration Page \(Proxy Services\)"](#)
- [Section 3.1.26, "WS Transport Configuration Page \(Business Services\)"](#)
- [Section 3.1.27, "WS Transport Configuration Page \(Proxy Services\)"](#)

3.1.1 BPEL-10g Transport Configuration Page (Business Services)

Use this page to configure transport settings for a business service using the BPEL-10g (Oracle BPEL Process Manager) transport protocol. For more information on using Oracle Service Bus with Oracle BPEL Process Manager, see the "Oracle BPEL Process Manager Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV1147>.

Table 3–1 BPEL-10g Transport Configuration Options for Business Services

Property	Description
Role	<p>The BPEL transport is used to send request messages from Oracle Service Bus to Oracle BPEL Process Manager. The business service can serve one of the following roles:</p> <ul style="list-style-type: none"> ■ Synchronous Client For synchronous communication with an Oracle Service Bus client, the only location information that is required is the BPEL address. This address is captured statically as the endpoint URI and/or dynamically through URI rewriting. ■ Asynchronous Client For asynchronous communication with an Oracle Service Bus client, a callback from Oracle BPEL Process Manager to Oracle Service Bus is sent on a different connection than the request, and you must configure Oracle Service Bus to provide the correct callback address. ■ Service Callback If the business service is designed to be a service callback to Oracle BPEL Process Manager (where Oracle BPEL Process Manager is calling an external service through Oracle Service Bus), the callback address is known only at run time. Use an Endpoint URI of <code>bpel://callback</code>. If you configure the business service with the marker URI, configure your pipeline logic to dynamically set the URI on \$outbound; for example, using the <code>TransportHeader</code> action. <p>Note: A Service Callback business service does not support load balancing or failover.</p>
Callback Proxy	<p>This optional field is available only for the Asynchronous Client role. This field lets you select the proxy service (must be either an SB or HTTP proxy of type Any SOAP) that will be used to route callbacks to the Oracle Service Bus client that made the request.</p>
Service Account	<p>For JNDI context security, used to access the Oracle BPEL Process Manager delivery service. Click Browse and select a service account. If no service account is specified, an anonymous subject is used.</p> <p>There is no restriction on the type of service account that can be configured, such as static or pass-through, but the run time must be able to access a user name and password.</p>

Table 3–1 (Cont.) BPEL-10g Transport Configuration Options for Business Services

Property	Description
Suspend Transaction	<p>Selecting Suspend Transaction makes the business service non-transactional even if the business service is invoked by a transaction.</p> <p>If you do not select Suspend Transaction:</p> <ul style="list-style-type: none"> ■ If the protocol indicates an Oracle WebLogic Server-supported protocol (t3, iiop, http), the transaction is propagated. ■ If the protocol indicates an OC4J server (ormi, opmn), the BPEL transport throws an exception, since OC4J does not support transaction propagation. <p>The BPEL transport ignores the Suspend Transaction option in the following situations:</p> <ul style="list-style-type: none"> ■ The business service is called with quality of service (QoS) "best-effort." The BPEL transport automatically suspends any transaction that does not support QoS. ■ The business service is called with QoS set to "exactly-once" and there is no transaction.
Dispatch Policy	<p>Select the instance of Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists.</p> <p>For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112.</p>

3.1.2 DSP Transport Configuration Page (Business Services)

Note: The DSP transport is being deprecated and will be removed from future releases.

Use this page to configure transport settings for a business service using the DSP (Oracle Data Service Integrator) transport protocol. For more information on accessing Oracle Data Service Integrator from Oracle Service Bus, see "DSP and Oracle Data Service Integrator Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV1229>.

Table 3–2 DSP Transport Configuration Options for Business Services

Option	To create or edit...
Debug Level	<p>Specify one of the following</p> <ul style="list-style-type: none"> ■ 0 - for no debug information ■ 1 - to output information on the request message ■ 3 - to output information on the request and the response message
Service Account	<p>Click Browse and select a service account from the list displayed. If no service account is specified, an anonymous subject is used.</p>

Table 3–2 (Cont.) DSP Transport Configuration Options for Business Services

Option	To create or edit...
Dispatch Policy	Select the instance of Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists. The Work Manager is used to post the reply message for response processing. See "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112 .

3.1.3 EJB Transport Configuration Page (Business Services)

Use this page to configure transport settings for a business service using the EJB transport protocol.

For more information on the EJB transport, see "EJB Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV1009>.

Table 3–3 EJB Transport Configuration Options for Business Services

Option	To create or edit...
Pass Caller's Subject	Select this check box to have Oracle Service Bus pass the authenticated subject from the proxy service when invoking the EJB and no service accounts are configured. Note that the Service Account field is disabled when this option is selected.
Service Account	Click Browse and select a service account from the list displayed. If no service account is specified, an anonymous subject is used. This option is not available if you use the Pass Caller's Subject option.
Supports Transaction	Select this check box to specify transaction support.
Client Jar	Click Browse and select an EJB client JAR resource from the list displayed.
Converter Jar	Click Browse and select an EJB converter class JAR resource from the list displayed.
Home Interface	EJB 2.1 only – Select the required EJBHome interface from the options populated by the JAR. The JNDI name in this URI sample must be associated with the EJBHome interface you select here. If the EJB is not of the required type or an EJBHome interface is not specified in the client-jar, Oracle Service Bus displays a warning.
Remote Interface	EJB 2.1 only – This field is automatically populated depending on the configuration of the Home Interface.
Business Interface	EJB 3.0 only – Select the business interface in the client JAR that you want to invoke.
Target Namespace	This field is populated by information picked up from the JAR.

Table 3–3 (Cont.) EJB Transport Configuration Options for Business Services

Option	To create or edit...
Style	<p>Select Document Wrapped or RPC according to your requirements. If two or more methods of your stateless session EJB have the same number and data type of parameters, and you want the operations to be document-oriented, you must specify that they be document-wrapped.</p> <p>The style is important because when routing or publishing to the EJB, <code>\$body</code> must have content that matches the style. Also when calling out to an EJB, the style affects the parameter contents, especially for document wrapped. Secondly one usage pattern is to define an EJB business service and then create a proxy service with the same WSDL that routes to the EJB. In this case care must be taken on the style of the WSDL because the client tool used to invoke the proxy might have limitations on the style of the WSDL.</p>
Encoding	Select Encoded or Literal .
Methods	<p>The methods displayed are those of the EJB remote or business interface you selected. Select the required methods (you can select multiple methods). Click + to expand the method. Edit the default parameter values and select a converter if provided (or required).</p> <p>You must exclude the methods with parameters or return types that are not supported by the JAX-RPC stack or you must associate such arguments with converter classes.</p> <p>You can change the default operation name for a given method. (By default, the operation name is the method name.) If an EJB contains methods with same name, you must change the operation names so that they are unique—WSDLs require unique operation names.</p> <p>Note: If the credentials or transaction settings are different between the methods for a given EJB, you can customize the methods for a given business service, and create a business service per method. This gives you fine-grained control over transactions and credentials.</p>
Exceptions	<p>This field appears if a method throws a business exception. If an EJB method throws an exception that has data types not supported by Java Web Services (JWS), such as an <code>ArrayList</code>, use the Exceptions field to select a converter class that converts the exception to a type supported by JWS.</p> <p>Your converter class must implement <code>com.bea.wli.sb.transports.ejb.ITypeConverter</code>. Converter classes can only be configured for checked exceptions and not for run-time exceptions.</p> <p>Package the converter class and the converted exception class in the client or converter JAR.</p> <p>For more information, see "EJB Transport" in the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBDV1009.</p>

3.1.4 E-Mail Transport Configuration Page (Business Services)

Use this page to configure transport settings for a business service using the e-mail transport protocol.

Table 3–4 E-Mail Transport Configuration Options for Business Services

Option	To create or edit...
SMTP Server	Select the default SMTP Server to use for endpoint URI entries of name@domain_name.com. If you provide SMTP server parameters in the endpoint URI, as described in Table 4–6 , those server resources are used instead of this SMTP Server setting. Do not select an SMTP server if you use the Mail Session option. You must first create the SMTP Server resource. For more information, see Section 2.1.18, "Creating SMTP Server Resources."
Mail Session	Enter the JNDI name of the configured mail session to use for endpoint URI entries of name@domain_name.com. If you provide JNDI mail session parameters in the endpoint URI, as described in Table 4–6 , those mail sessions are used instead of this Mail Session setting. Do not enter a Mail Session if you use the SMTP Server option.
From Name	Enter a display name for the originating e-mail account for this service.
From Address	Enter the originating e-mail account for this service.
Reply To Name	Enter a display name for the reply to e-mail account.
Reply To Address	Enter an e-mail address to reply to.
Connection Timeout	Enter the timeout interval, in seconds, before the connection is dropped. If you enter 0, there is no timeout.
Request Encoding	Accept the default ISO-8859-1 as the character set encoding for requests in e-mail transports, or enter a different character set encoding.

3.1.5 E-Mail Transport Configuration Page (Proxy Services)

Use this page to configure transport settings for a proxy service using the e-mail transport protocol.

Table 3–5 E-Mail Transport Configuration Options for Proxy Services

Option	To create or edit...
Service Account	Enter a service account name, or click Browse to select service accounts from a browser.
Managed Server	Select the Managed Server for polling in a clustered domain. This field is available <i>only</i> in a clustered domain.
Polling Interval	Enter a polling interval, in seconds.
E-Mail Protocol	Select POP3 or IMAP as the server type for the e-mail account.
Read Limit	Specify the maximum number of messages to read per polling sweep. Enter 0 to specify no limit.
Pass By Reference	Select this check box to stage the file in the archive directory and pass it as a reference in the headers. By default when you create a new service, the Pass By Reference option is selected and you must specify the archive directory location.
Pass Attachments by Reference	Select this check box to stage the attachments in the archive directory and pass them as a reference in the headers. By default, when the Pass By Reference option is selected, the Pass Attachments By Reference option is implicitly true and you must specify the archive directory location.

Table 3–5 (Cont.) E-Mail Transport Configuration Options for Proxy Services

Option	To create or edit...
Post Read Action	Select what happens to a message after it has been read: <ul style="list-style-type: none"> ■ Archive - The message is archived. ■ Delete - The message is deleted. ■ Move - The message is moved. Move is only available with the IMAP protocol.
Attachments	Select how attachments are handled: <ul style="list-style-type: none"> ■ Archive - Attachments are saved to the archive directory. ■ Ignore - Attachments are ignored.
IMAP Move Folder	Enter the folder to which the message is moved if the Post Read Action field is set to Move .
Download Directory	Enter a temporary location for downloading e-mails.
Archive Directory	Specify the path to the archive location if the Post Read Action field is set to Archive . This field is required if the Pass By Reference or Pass Attachments By Reference option is selected.
Error Directory	Enter the file system directory path to write the message and any attachments if there is a problem.
Request Encoding	Accept the default ISO-8859-1 as the character set encoding for requests in E-mail transports, or enter a different character set encoding.

3.1.6 File Transport Configuration Page (Business Services)

Use this page to configure transport settings for a business service using the file transport protocol.

Table 3–6 File Transport Configuration Options for Business Services

Option	To create or edit...
Prefix	Enter a prefix to be prepended to the file name. Do not enter * in this field. This character causes a run-time exception.
Suffix	Enter a suffix to be appended to the file name. This is a required field. Do not enter * in this field. This character causes a run-time exception.
Request Encoding	Accept the default utf-8 as the character set encoding for requests in File transports, or enter a different character set encoding.

3.1.7 File Transport Configuration Page (Proxy Services)

Use this page to configure transport settings for a proxy service using the file transport protocol.

Table 3–7 File Transport Configuration Options for Proxy Services

Option	To create or edit...
File Mask	Specify the files that the proxy service should poll. If the URI is a directory and you specify * . *, the service polls for all the files in the directory. Only the wildcard characters * and ? are allowed in the File Mask. Regular expressions are not supported.

Table 3–7 (Cont.) File Transport Configuration Options for Proxy Services

Option	To create or edit...
Managed Server	Select the Managed Server for polling in a clustered domain. This field is available <i>only</i> in a clustered domain.
Polling Interval	Enter a polling interval, in seconds. The default is 60.
Read Limit	Specify the maximum number of messages to read per polling sweep. Enter 0 to specify no limit. The default is 10.
Sort By Arrival	Select this check box to specify that events are delivered in the order of arrival. Note that when this option is selected for a proxy service that is executed in a clustered environment, messages are always sent to the same server. In other words, load balancing across servers is ignored when this option is selected.
Scan Subdirectories	Select this check box to recursively scan all the directories.
Pass By Reference	Select this check box to stage the file in the archive directory and pass it as a reference in the headers.
Post Read Action	Select what happens to a message after it has been read: <ul style="list-style-type: none"> ■ Archive - The message is archived. ■ Delete - The message is deleted.
Stage Directory	Enter an intermediate directory to temporarily stage the files while processing them. Do not put the stage directory inside of the polling directory (the directory identified in the URL of the file transport proxy service; for example, file://c:/dir1/dir2).
Archive Directory	Specify the path to the archive location if the Post Read Action option is set to Archive . The Archive Directory field is also a required field if you have selected the Pass By Reference field. Do not put the archive directory inside of the polling directory.
Error Directory	Enter the location where messages and attachments are posted if there is a problem. Do not put the error directory inside of the polling directory.
Request Encoding	Accept the default UTF-8 as the character set encoding for requests in file transports, or enter a different character set encoding.

3.1.8 FTP Transport Configuration Page (Business Services)

Use this page to configure transport settings for a business service using the ftp transport protocol.

Table 3–8 FTP Transport Configuration Options for Business Services

Option	To create or edit...
User Authentication	Select anonymous if the user of the FTP server is anonymous, or select external user if the user of the FTP server is an externally configured account.
Identity (E-mail id)	This field is available only if the User Authentication option is set to anonymous . Enter the mail ID for the anonymous user.

Table 3–8 (Cont.) FTP Transport Configuration Options for Business Services

Option	To create or edit...
Service Account	This field is available only if the User Authentication option is set to external user . Enter the service account for the external user.
Timeout	Enter the socket timeout interval, in seconds, before the connection is dropped. The default is 60 seconds.
Prefix for destination File Name	Enter a prefix for the file name under which the file is stored on the remote server. Do not enter * in this field. This character causes a run-time exception.
Suffix for destination File Name	Enter a suffix for the file name under which the file is stored on the remote server. Do not enter * in this field. This character causes a run-time exception.
Transfer Mode	Select ASCII or binary as the transfer mode.
Request Encoding	Accept the default UTF-8 as the character set encoding for requests in ftp transports, or enter a different character set encoding.

3.1.9 FTP Transport Configuration Page (Proxy Services)

Use this page to configure transport settings for a proxy service using the ftp transport protocol.

Table 3–9 FTP Transport Configuration Options for Proxy Services

Option	To create or edit...
User Authentication	Select anonymous if the user of the FTP server is anonymous, or select external user if the user of the FTP server is an externally configured account.
Identity (E-Mail ID)	This field is available only if the User Authentication option is set to anonymous . Enter the mail ID for the anonymous user.
Service Account	This field is available only if the User Authentication option is set to external user . Enter the service account for the user. This is a required field when the User Authentication option is set to external user .
Pass By Reference	Select this check box to stage the file in the archive directory and pass it as a reference in the headers.
Remote Streaming	Select this check box to stream the FTP files directly from the remote server at the time of processing. When you select this option, the archive directory is the remote directory on the remote FTP server machine. Therefore, you should specify the archive directory as relative to the FTP user directory.
File Mask	Enter the regular expression for the files to be picked. The default is *. *.
Managed Server	Select the Managed Server for polling in a clustered domain. This field is available <i>only</i> in a clustered domain.
Polling Interval	Enter a polling interval, in seconds. The default is 60.
Read Limit	Specify the maximum number of messages to read per polling sweep. Enter 0 to specify no limit. The default is 10.

Table 3–9 (Cont.) FTP Transport Configuration Options for Proxy Services

Option	To create or edit...
Post Read Action	Select what happens to a message after it has been read. <ul style="list-style-type: none"> ■ Archive - The message is archived. ■ Delete - The message is deleted.
Transfer Mode	Select ASCII or binary as the transfer mode.
Archive Directory	Specify the path to the archive location if the Post Read Action option is set to Archive . This field is required if the Pass By Reference option is selected. <p>Note: The Archive, Download, and Error directories are absolute paths, and they are automatically created. If you specify a relative path, the files are created relative to the Java process that starts the Oracle WebLogic Server.</p>
Download Directory	Enter the directory on your local machine where files are downloaded during the file transfer. <p>Note: The Archive, Download, and Error directories are absolute paths, and they are automatically created. If you specify a relative path, the files are created relative to the Java process that starts the Oracle WebLogic Server.</p>
Error Directory	Enter the location where messages are posted if there is a problem. <p>Note: The Archive, Download, and Error directories are absolute paths, and they are automatically created. If you specify a relative path, the files are created relative to the Java process that starts the Oracle WebLogic Server.</p>
Request Encoding	Accept the default UTF-8 as the character set encoding for requests in FTP transports.
Scan Subdirectories	Select this check box to recursively scan all directories
Sort By Arrival	Select this check box to deliver events in the order of arrival.
Timeout	Enter the socket timeout interval, in seconds, before the connection is dropped. If you enter 0, there is no timeout.
Retry Count	Specify the number of retries for FTP connection failures.

3.1.10 HTTP Transport Configuration Page (Business Services)

Use this page to configure transport settings for a business service using the HTTP transport protocol. The HTTP transport supports both HTTP and HTTPS endpoints.

Table 3–10 HTTP Transport Configuration Options for Business Services

Option	To create or edit...
Read Timeout	Enter the read timeout interval in seconds. A zero (0) value indicates no timeout.
Connection Timeout	Enter the connection timeout interval in seconds. If the timeout expires before the connection can be established, Oracle Service Bus raises a connection error. A zero (0) value indicates no timeout.

Table 3–10 (Cont.) HTTP Transport Configuration Options for Business Services

Option	To create or edit...
HTTP Request Method	<p data-bbox="647 260 1398 310">This parameter lets you to use one of the following HTTP methods in a request:</p> <ul style="list-style-type: none"> <li data-bbox="647 327 1419 457">■ POST – Passes all its data, of unlimited length, directly over the socket connection as part of its HTTP request body. The exchange is invisible to the client, and the URL does not change. For REST-based requests, POST tells the transport to perform a create/replace operation or perform an action with the request. <li data-bbox="647 474 1419 680">■ GET – You can include as part of the request some of its own information that better describes what to get. This information is passed as a sequence of characters appended to the request URL in a query string. You can use GET in a business service with a Service Type of "Any XML Service," or with a Service Type of "Messaging Service" when the Request Message Type is set to "None." For REST-based requests, GET retrieves a representation of a remote resource. <li data-bbox="647 697 1419 806">■ PUT – You can use PUT in a business service with a Service Type of "Any XML Service" or "Messaging Service." PUT tells the transport to perform a create/replace operation with a REST-based request, such as uploading a file to a known location. <li data-bbox="647 823 1419 953">■ HEAD – You can use HEAD in a business service with a Service Type of "Any XML Service," or with a Service Type of "Messaging Service" when the Response Message Type is set to "None." HEAD tells the transport to get header information for a remote resource rather than getting a full representation of the resource in a REST-based request. <li data-bbox="647 970 1419 1045">■ DELETE – You can use PUT in a business service with a Service Type of "Any XML Service" or "Messaging Service." DELETE tells the transport to perform a delete operation with a REST-based request. <p data-bbox="647 1062 1435 1140">Note: If a method is already set in the \$outbound/transport/request/http:http-method variable, that value takes precedence over any method you select for HTTP Request Method.</p>

Table 3–10 (Cont.) HTTP Transport Configuration Options for Business Services

Option	To create or edit...
Authentication	<p>Select one of the following:</p> <ul style="list-style-type: none"> ■ None - Specifies that authentication is not required to access this service. ■ Basic - Specifies that basic authentication is required to access this service. <p>Basic authentication instructs Oracle WebLogic Server to authenticate the client using a user name and password against the authentication providers configured in the security realm, such as a Lightweight Directory Access Protocol (LDAP) directory service and Windows Active Directory. The client must send its user name and password on the HTTP request header.</p> <p>Basic authentication is strongly discouraged over HTTP because the password is sent in clear text. However, it is safe to send passwords over HTTPS because HTTPS provides an encrypted channel.</p> <p>Caution: By default, all users (authorized and anonymous) can access a business service. To limit the users who can access a business service, create a transport-level authorization policy. can access a proxy service, create a transport-level authorization policy. See "Editing Transport-Level Access Policies" in the <i>Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBAG1208.</p> <p>Client Certificate - Specifies encrypted communication and strong client authentication (two-way SSL). To learn more, see "Configuring Transport-Level Security" in the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBDV1557.</p>
Service Account	<p>Enter a service account. A service account is an alias resource for a user name and password. This is a required field if you selected the Basic Authentication Required field.</p>
Dispatch Policy	<p>Select the instance of Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists.</p> <p>For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112.</p>
Request Encoding	<p>Accept the default <code>iso-8859-1</code> as the character set encoding for requests in HTTP transports, or enter a different character set encoding.</p>
Response Encoding	<p>Accept the default <code>iso-8859-1</code> as the character set encoding for responses in HTTP transports, or enter a different character set encoding.</p>
Proxy Server	<p>Enter a proxy server resource or click Browse to choose an entry from the list of configured proxy server resources.</p>
Follow HTTP redirects	<p>Select this check box to specify that HTTP redirects (which are requests with a response code 3xx) should be automatically followed. A redirect occurs when you send an outbound request to the URL of a business service, and that service returns a response code (for example, 302) that says the URL is no longer valid and this request needs to be sent to another URL. If you select the Follow HTTP Redirects check box, Oracle Service Bus automatically re-sends the request to the new URL without any action on your part. Deselect this check box if you do not want the HTTP redirects to be automatically followed.</p>

Table 3–10 (Cont.) HTTP Transport Configuration Options for Business Services

Option	To create or edit...
Use Chunked Streaming Mode	<p>Select this option if you want to use HTTP chunked transfer encoding to send messages.</p> <p>Note: Do not use chunked streaming with if you use the Follow HTTP Redirects option. Redirection and authentication cannot be handled automatically in chunked mode.</p>

3.1.11 HTTP Transport Configuration Page (Proxy Services)

Use this page to configure transport settings for a proxy service using the HTTP transport protocol. The HTTP transport supports both HTTP and HTTPS endpoints.

Table 3–11 HTTP Transport Configuration Options for Proxy Services

Option	To create or edit...
HTTPS required	<p>Select this check box for inbound HTTPS endpoints.</p> <p>To learn more, see "Configuring Transport-Level Security" in the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBDV1557.</p>
Authentication	<p>Select one of the following:</p> <ul style="list-style-type: none"> ■ None - Specifies that authentication is not required. ■ Basic - Specifies that basic authentication is required to access this service. <ul style="list-style-type: none"> Basic authentication instructs Oracle WebLogic Server to authenticate the client using a user name and password against the authentication providers configured in the security realm, such as a Lightweight Directory Access Protocol (LDAP) directory service and Windows Active Directory. The client must send its user name and password on the HTTP request header. Basic authentication is strongly discouraged over HTTP because the password is sent in clear text. However, it is safe to send passwords over HTTPS because HTTPS provides an encrypted channel. Caution: By default, all users (authorized and anonymous) can access a proxy service. To limit the users who can access a proxy service, create a transport-level authorization policy. See "Editing Transport-Level Access Policies" under "Security Configuration" in the <i>Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBAG1176. ■ Client Certificate - Specifies encrypted communication and strong client authentication (two-way SSL). To learn more, see "Configuring Transport-Level Security" in the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBDV1557. ■ Custom Authentication - Specifies that an authentication token is contained in an HTTP header. The client's identity is established through the use of this client-supplied token. You must configure an Identity Assertion provider that maps the token to an Oracle Service Bus user. <ul style="list-style-type: none"> The custom authentication token can be of any active token type supported by a configured Oracle WebLogic Server Identity Assertion provider.

Table 3–11 (Cont.) HTTP Transport Configuration Options for Proxy Services

Option	To create or edit...
Dispatch Policy	<p>Select a dispatch policy for this endpoint. Leave blank to use the default dispatch policy.</p> <p>Dispatch policy refers to the instance of WLS Work Manager that you want to use for the service endpoint.</p> <p>For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112.</p>
Request Encoding	<ul style="list-style-type: none"> ■ For HTTP inbound transports: <p>If the character set encoding parameter of the <code>Content-Type</code> header is not specified in Client Request, enter a character set encoding parameter. If you do not enter a value, the field defaults to ISO-8859-1.</p> ■ For HTTP outbound transports: <p>If you have not configured a request encoding, the Oracle Service Bus run time decides the most appropriate encoding while it makes a request to the business service. In the case of a non-passthrough scenario, the default character encoding is UTF-8 at run time. However if it is a passthrough scenario, the run time will pass through the encoding received with the outbound response.</p>
Response Encoding	Accept the default ISO-8859-1 as the character set encoding for responses in HTTP transports, or enter a different character set encoding.
Authentication Header	<p>Enter the HTTP header (any except <code>Authorization</code>) from which Oracle Service Bus is to extract the token. This field is available only if you selected the Custom Authentication check box.</p> <p>For example, <code>client-xyz-token</code>.</p>
Authentication Token Type	Select an authentication token type. Only the active token types configured for an Identity Assertion provider are available. This field is available only if you selected the Custom Authentication check box.

3.1.12 JCA Transport Configuration Page (Proxy and Business Services)

Use this page to configure transport settings using the JCA transport protocol. For more information on using the JCA transport, see "JCA Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV910>.

Table 3–12 JCA Transport Configuration Options

Option	Description
JCA File	<p>Click Browse to select a JCA resource. The JCA resource defines different aspects of the service, such as details about the adapter used, a binding to the WSDL and TopLink or EclipseLink mapping file, and the activation/interaction spec properties required by the service.</p> <p>Once you select a valid JCA resource, the remaining transport configuration fields become available.</p>
Adapter Name	A read-only value showing the name of the adapter that the JCA service will use.
Adapter Type	A read-only value showing the adapter type.

Table 3–12 (Cont.) JCA Transport Configuration Options

Option	Description
Dispatch Policy	<p>Select the instance of Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists.</p> <p>For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112</p>
JNDI Service Account	<p>JNDI Service Account is for JNDI context security, used to access the EIS adapter managed connection factory. Click Browse and select a service account. If no service account is specified, an anonymous subject is used.</p> <p>For JCA business services, there is no restriction on the type of JNDI service account that can be configured, such as static or pass-through, but the run time must be able to access a user name and password. JCA proxy services can use only static JNDI service accounts.</p> <p>For more information on JNDI service accounts, see "Security" in the "JCA Transport" chapter of the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBDV916</p>
EndPoint Properties	<p>This field lets you assign values to endpoint properties such as retries for the type of adapter the service uses.</p> <p>For a list of supported endpoint properties, see "Endpoint Properties in the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBDV1841.</p>
Dynamic EndPoint Properties	<p>This option lets you pass request parameters to JCA-compliant services. For example, you can use a dynamic endpoint property to pass database query parameters to the Oracle JCA Adapter for Database.</p> <p>For more information on querying with parameters, see "Oracle JCA Adapter for Database" in the <i>Oracle Fusion Middleware User's Guide for Technology Adapters</i> at http://www.oracle.com/pls/as1111/lookup?id=TKADP2117.</p> <p>Enter a name/value pair for each dynamic endpoint property you want to provide. The endpoint property key matches the query parameter name.</p>
Always use configuration from JCA file	<p>This option determines whether or not Activation Spec Properties (proxy services) and Interaction Spec Properties (business services) are always used from the JCA file.</p> <p>If this option is selected (default), the JCA transport interacts with the JCA framework using the activation/interaction spec properties in the JCA file.</p> <p>If this option is deselected, you can override the Activation/Interaction Spec Properties.</p> <p>For the redeployment impact of using this option, see "Endpoint Redeployment" in the "JCA Transport" chapter of the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBDV940</p>
Operation Name	<p>Displays a read-only name of the selected WSDL operation. An operation can have its own activation/interaction spec properties, shown in the Activation/Interaction Spec Properties field.</p>

Table 3–12 (Cont.) JCA Transport Configuration Options

Option	Description
Activation/Interaction Spec Properties	<p>Activation Spec Properties is the field name for proxy services; Interaction Spec Properties is the field name for business services.</p> <p>If this service is an inbound service invoked by an EIS application, this field displays the activation spec properties for the JCA inbound operation shown in Operation Name field.</p> <p>You can override the activation/interaction spec properties if you deselect Always use configuration from JCA file.</p> <p>Note: For Oracle Adapter Suite adapters, activation/interaction spec properties are displayed as read-only. The Oracle Adapter Suite adapters store their own configurations, which you must change in the Oracle Adapter Suite management tools.</p>
Connection properties (legacy)	<p>For legacy JCA services that use non-managed mode connection properties (deprecated in this release), see the connection configuration options at http://download.oracle.com/docs/cd/E13159_01/osb/docs10gr3/jcatransport/transport.html#wp1105451.</p>

For more information on endpoint and activation/interaction spec properties, see the *Oracle Fusion Middleware User's Guide for Technology Adapters* at <http://www.oracle.com/pls/as1111/lookup?id=TKADP>.

3.1.13 JEJB Transport Configuration Page (Business Services)

Use this page to configure transport settings for a business service using the JEJB transport protocol.

Table 3–13 JEJB Transport Configuration for Business Services

Option	Description
Dispatch Policy	<p>Select the instance of the Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists.</p> <p>For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112.</p>
EJB Spec Version	Select the EJB version of the remote EJB interface.
Pass XMLBeans by value	<p>Select this option if you want the transport to generate an "inlined" XML representation of POJO arguments (an XMLObject) whose parameters you can access and manipulate with XQuery expressions.</p> <p>Note: Type information is not available inline for XMLObjects passed by value. If you use this option, you cannot pass the typed XMLObject as the argument in a Java Callout in a proxy service pipeline.</p> <p>Do not select this option if you want to pass the POJO by reference, which also results in better performance.</p>
Pass Caller's Subject	As an alternative to selecting a Service Account, select this option to have Oracle Service Bus pass the authenticated subject from the proxy service when invoking the EJB.

Table 3–13 (Cont.) JEJB Transport Configuration for Business Services

Option	Description
Service Account	Click Browse and select a JNDI service account from the list displayed. If no service account is specified, an anonymous subject is used. For more information, see Section 4.19, "Service Accounts."
Client JAR	Click Browse and select an EJB client JAR resource from the list displayed. The client JAR contains the remote or business interface for the remote service. The Client JAR is registered as a generic Archive Resource.
Home Interface	EJB 2.1 only – Select the required EJBHome interface from the options populated by the JAR.
Remote Interface	EJB 2.1 only – This field is automatically populated based on the configuration of the Home Interface.
Business Interface	EJB 3.0 only – Select the business interface from the client JAR that you want to invoke.
Target Namespace	This field is populated by information picked up from the JAR.
Methods	Select the required methods. Click + to expand the method, which lets you edit the default parameter values. You can change the default operation name for a given method. By default, the operation name is the method name. If an EJB contains methods with same name (overloaded), you must change the operation names so that they are unique. WSDLs require unique operation names.

3.1.14 JEJB Transport Configuration Page (Proxy Services)

Use this page to configure transport settings for a proxy service using the JEJB transport protocol.

Table 3–14 JEJB Transport Configuration for Proxy Services

Option	Description
Dispatch Policy	Select the instance of the Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists. For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112 .
EJB Spec Version	Select the EJB version of the remote EJB interface.
Pass XMLBeans by value	Select this option if you want the transport to generate an "inlined" XML representation of POJO arguments (an XMLObject) whose parameters you can access and manipulate with XQuery expressions. Note: Type information is not available inline for XMLObjects passed by value. If you use this option, you cannot pass the typed XMLObject as the argument in a Java Callout in a proxy service pipeline. Do not select this option if you want to pass the POJO by reference, which also results in better performance.

Table 3–14 (Cont.) JEEB Transport Configuration for Proxy Services

Option	Description
Transaction Attribute	<p>Select one of the following options for handling transactions:</p> <ul style="list-style-type: none"> ▪ Supports – The transport accepts an incoming transaction. Quality of service is exactly-once if the operation is invoked in a transaction and best-effort if the operation is invoked outside of a transaction. ▪ Required – The transport accepts an incoming transaction. If no ongoing transaction exists, the transport starts one. Quality of service is exactly-once. ▪ RequiresNew – The transport always starts a new transaction, suspending an ongoing transaction. Quality of service is exactly-once. ▪ Mandatory – The transport invokes the method in the existing transaction. Quality of service is exactly-once. ▪ NotSupported – The transport suspends an existing transaction and resumes it on invocation. Quality of service is best-effort. ▪ Never – The transport does not invoke the method in a transaction. Quality of service is best-effort.
Client JAR	Click Browse and select an EJB client JAR resource from the list displayed. The client JAR contains the remote or business interface for the remote EJB. The Client JAR is registered as a generic Archive Resource.
Home Interface	EJB 2.1 only – Select the required EJBHome interface from the options populated by the client JAR.
Remote Interface	EJB 2.1 only – This field is automatically populated based on the configuration of the Home Interface.
Business Interface	EJB 3.0 only – Select the business interface from the client JAR that you want to invoke.
Target Namespace	This field is populated by information picked up from the JAR.
Methods	<p>Select the required methods. Click + to expand the method, which lets you edit the default parameter values.</p> <p>You can change the default operation name for a given method. By default, the operation name is the method name. If an EJB contains methods with same name (overloaded), you must change the operation names so that they are unique. WSDLs require unique operation names.</p>

3.1.15 JMS Transport Configuration Page (Business Services)

Use this page to configure transport settings for a business service using the JMS transport protocol. For more information, see "JMS Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV1037>.

Table 3–15 JMS Transport Configuration Options for Business Services

Option	To create or edit...
Destination Type	<p>Select a type for the JMS bridge destination:</p> <ul style="list-style-type: none"> ▪ Queue (for point-to-point) ▪ Topic (for publish/subscribe)

Table 3–15 (Cont.) JMS Transport Configuration Options for Business Services

Option	To create or edit...
Message Type	Select one of the following: <ul style="list-style-type: none"> ■ Bytes (for a stream of uninterpreted bytes) ■ Text (for text messages)
Response Queues	This option is available only when Queue is selected for the Destination Type. Select one of the following response options: <ul style="list-style-type: none"> ■ None – No response is expected. Set this option for one-way operations. ■ One for all Request URIs – Lets you enter a single URI to handle the response, as well as set other response configuration details such as encoding and timeout, and optionally select a JMS Service Account for passing JMS/JNDI credentials. ■ One per Request URI – This option provides response failover, letting you enter a response URI or destination for each request URI. You can optionally select a service account for JMS/JNDI credentials on each request/response pairing.
Response Pattern	This option is available only when you select a response option in the Response Queue field. Select one of the following: <ul style="list-style-type: none"> ■ Select JMSCorrelationID for all services other than JAX-RPC services running on Oracle WebLogic Server. ■ Select JMSMessageID for JAX-RPC services running on Oracle WebLogic Server. For more information, see "Message ID and Correlation ID Patterns for JMS Request/Response" in the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBDV1051 .
Dispatch Policy	Select the instance of Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists. For example, if the business service has a JMS transport protocol, the business service endpoint is an MDB (message-driven bean) JAR file that you can associate with the specific dispatch policy. For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112 .
Request Encoding	Enter the character set for encoding requests. The default is UTF-8.
Response Encoding	This option is available only when one of the response options is selected in the Response Queues field. Enter the character set for encoding responses. The default is UTF-8.
Response Timeout	This option is available only when one of the response options is selected in the Response Queues field. Enter the amount of time, in seconds, to wait for the response before dropping the connection. The default, zero (0), means the response never times out.

Table 3–15 (Cont.) JMS Transport Configuration Options for Business Services

Option	To create or edit...
Response URI	<p>This option is available when you select the One for all Request URIs response option and the JMSCorrelationID response pattern.</p> <p>Enter a response URI in the format:</p> <pre>jms://host:port/connection_factory/jndi_destination</pre> <p>To target multiple servers, use the following format:</p> <pre>jms://host1:port,host2:port/connection_factory/jndi_destination</pre> <p>You can also omit the host and port in the response URI. For example:</p> <pre>jms:///connection_factory/jndi_destination</pre> <p>When you omit host and port, the connection factory/destination lookup occurs on the local server. This is useful, for example, if the request URI goes to a foreign connection factory/destination, but you want the response sent to the local server.</p>
Request Responses	<p>This option is available when you select the One per Request URI response option for the JMSCorrelationID pattern to provide response failover.</p> <p>For each request URI entered on the generic Transport page, enter a Response URI, as described in the previous Response URI field.</p> <p>You can select an optional Service Account for JMS/JNDI credentials that the service uses for both the request and response queues.</p>
Target - Responses	<p>This option is available when you select the One for all Request URIs response option for the JMSMessageID pattern.</p> <p>Enter the name of the Target server that is to receive responses, and enter a Response URI, as described in the Response URI field.</p>
Request Connections	<p>This option is available when you select the One per Request URI response option for the JMSMessageID pattern to provide response failover.</p> <p>For each request URI, identified sequentially by number in the Seq.No field, optionally enter a JMS Connection Factory name. If you do not enter a name, the JMS transport uses the connection factory from the request URI.</p> <p>You can select an optional Service Account for JMS/JNDI credentials that the service uses for both the request and response queues.</p>
Target - Destinations	<p>This option is available when you select the One per Request URI response option for the JMSMessageID pattern to provide response failover. Use this field in conjunction with Request Connections.</p> <p>Each Target server listed (determined by the servers in the current domain), such as managed servers in a cluster, lists the request URIs by Seq.No, which correspond to those in the Request Connections field. For each request URI on each target, enter the Destination queue on that target server that receives responses.</p> <p>Note: Because the Oracle Service Bus development environment supports only a one-server environment, only one Target is listed. To configure this field in a multi-server environment, deploy this service to the run-time environment and complete the service configuration in the Oracle Service Bus Console.</p>

Table 3–15 (Cont.) JMS Transport Configuration Options for Business Services

Option	To create or edit...
JMS Service Account	<p>This option is available when you use the None or One for all Request URIs option in the Response Queues field.</p> <p>Select a service account to use for the JMS resource managed by the JMS server. A service account is an alias resource for a user ID and password, used for both the request and response. The same service account is used for both JMS and JNDI purposes.</p> <p>This field is mutually exclusive with the Pass Caller's Subject option. Use one or the other for JMS/JNDI authentication.</p> <p>For more information, see Section 2.1.16, "Creating Service Account Resources."</p>
Use SSL	<p>Select this option only if the requests are made over a TLS/SSL connection.</p> <p>TLS/SSL (Secure Sockets Layer) provides secure connections by allowing two applications connecting over a network to authenticate the other's identity and by encrypting the data exchanged between the applications. Authentication allows a server, and optionally a client, to verify the identity of the application on the other end of a network connection. Additionally, if the administrator has restricted access to individual JMS destinations (queues or topics) by setting access control on the JNDI entry for the destination, the service must authenticate when looking up the entry in the JNDI tree. Authenticate using a Service Account or the Pass Caller's Subject option.</p>
Expiration	<p>The time interval in milliseconds after which the message expires. Default value is 0, which means that the message never expires.</p>
Enable Message Persistence	<p>The JMS delivery mode the service uses, which lets you balance reliability with throughput. Select this option (default) for guaranteed message delivery. Deselect this option to improve throughput if an occasional lost message is tolerable.</p>
Unit of Order	<p>Enter a message unit-of-order. Message unit-of-order enables message producers to group messages into a single unit with respect to the processing order. This single unit-of-order requires that all messages in that unit be processed sequentially in the order they were created.</p>
Pass Caller's Subject	<p>Select this check box to have Oracle Service Bus pass the authenticated subject when sending a message.</p> <p>When you enable this field and the business service targets JMS resources in a different domain, enable global trust on both domains. See "Configuring Security for a WebLogic Domain" in <i>Oracle Fusion Middleware Securing Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=SECMG402.</p> <p>This field is mutually exclusive with the Service Account option. Use one or the other for JMS/JNDI authentication.</p>
JNDI Timeout	<p>The JNDI connection timeout (in seconds) used while looking up the destination or connection factory in the JNDI tree.</p> <p>The default, zero (0), means the connection never times out.</p>

3.1.16 JMS Transport Configuration Page (Proxy Services)

Use this page to configure transport settings for a proxy service using the JMS transport protocol. For more information, see "JMS Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV1037>.

Table 3–16 JMS Transport Configuration Options for Proxy Services

Option	To create or edit...
Destination Type	Select one of the following: <ul style="list-style-type: none"> ■ Queue (for a point-to-point destination type) ■ Topic (for a publish/subscribe destination)
Is Response Required	This option is available only when Queue is selected for the Destination Type. Select this option to specify that a response is expected after an outbound message is sent.
Response Pattern	This option is available only when the Is Response Required check box is selected. Select one of the following: <ul style="list-style-type: none"> ■ Select JMSMessageID for JAX-RPC services running on Oracle WebLogic Server. ■ Select JMSCorrelationID for all other services. When you select this option, you must also enter a Response URI.
Response Message Type	This option is available only when the Is Response Required check box is selected. Select one of the following: <ul style="list-style-type: none"> ■ Bytes (for a stream of uninterpreted bytes) ■ Text (for text messages)
Dispatch Policy	Select the instance of Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists. For example, if the business service has a JMS transport protocol, the business service endpoint is an MDB (message-driven bean) JAR file that you can associate with the specific dispatch policy. For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112 .
Request Encoding	Enter the character set for encoding requests. The default is UTF-8.
Response Encoding	This option is available only when the Is Response Required check box is selected. Enter the character set for encoding responses. The default is UTF-8.
Client Response Timeout	This option is available only when the Is Response Required check box is selected. Enter the number of seconds to wait for a server response before dropping the connection. This only applies if the client is another proxy service in the same domain.

Table 3–16 (Cont.) JMS Transport Configuration Options for Proxy Services

Option	To create or edit...
Response URI	<p>This option is available only when JMSCorrelationID is selected for the Response Correlation Pattern.</p> <p>Enter a response URI in the format:</p> <pre>jms://host:port/connection_factory/jndi_destination</pre> <p>To target multiple servers, use the following format:</p> <pre>jms://host1:port,host2:port/connection_factory/jndi_destination</pre> <p>You can also omit the host and port in the response URI. For example:</p> <pre>jms:///connection_factory/jndi_destination</pre> <p>When you omit host and port, the connection factory/destination lookup occurs on the local server. This is useful, for example, if the request URI goes to a foreign connection factory/destination, but you want the response sent to the local server.</p>
Response Connection Factory	<p>This option is available only when JMSMessageID is selected for the Response Correlation Pattern.</p> <p>Enter a response connection factory URI.</p> <p>If a connection factory is not specified, the connection factory for the request is used for the response.</p>
JMS Service Account	<p>Select a service account to use for the JMS resource managed by the JMS server. A service account is an alias resource for a user ID and password, used for both the request and response. The same service account is used for both JMS and JNDI purposes.</p> <p>For more information, see Section 2.1.16, "Creating Service Account Resources."</p>
Use SSL	<p>Select this option only if the requests are made over a TLS/SSL connection.</p> <p>TLS/SSL (Secure Sockets Layer) provides secure connections by allowing two applications connecting over a network to authenticate the other's identity and by encrypting the data exchanged between the applications. Authentication allows a server, and optionally a client, to verify the identity of the application on the other end of a network connection. Additionally, if the administrator has restricted access to individual JMS destinations (queues or topics) by setting access control on the JNDI entry for the destination, the service must authenticate when looking up the entry in the JNDI tree.</p>
Message Selector	<p>Enter a message selector expression.</p> <p>Only messages with properties matching the expression are processed</p>
Durable Subscription	<p>This option is available only if Topic is selected for the Destination Type.</p> <p>Select this check box if the subscription is durable or leave it blank if the subscription is not durable</p>
Retry Count	<p>Enter the number of delivery retries a message can have before it is moved to the error destination. This field only applies to Oracle WebLogic Server JMS destinations.</p>
Retry Interval	<p>Enter the amount of time, in milliseconds, before rolled back or recovered messages are redelivered. This field only applies to Oracle WebLogic Server JMS destinations.</p>

Table 3–16 (Cont.) JMS Transport Configuration Options for Proxy Services

Option	To create or edit...
Error Destination	Enter the name of the target destination for messages that have reached their redelivery limit. This field only applies to Oracle WebLogic Server JMS destinations.
Expiration Policy	Select an Expiration Policy to use when an expired message is encountered on a destination. This field only applies to Oracle WebLogic Server JMS destinations.
Is XA Required	Select this check box if your connection factory is XA. This value is into account when the remote connection factory is unavailable. If your connection factory is available and this value is true, make sure that the connection factory is defined as transactional.
JNDI Timeout	The JNDI connection timeout (in seconds) used while looking up the destination or connection factory in the JNDI tree.

3.1.17 MQ Transport Configuration Page (Business Services)

Before you use the MQ transport, you must configure a MQ Connection resource. See [Section 2.6, "Working with MQ Connections."](#) For more information about the MQ transport, see "MQ Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV1117>.

Use this page to configure transport settings for a proxy service using the native MQ transport protocol.

Table 3–17 MQ Transport Configuration Options for Business Services

Option	To create or edit...
Message Type	Select one of the following: <ul style="list-style-type: none"> ■ Bytes (for a stream of uninterpreted bytes) ■ Text (for text messages)
Is Response Required	Select this option to specify that a response is expected after an outbound message is sent.
Polling Interval	This option is available only when the Is Response Required check box is selected. Enter a polling interval, in milliseconds. The default is 1000.
Response Correlation Pattern	This option is available only when the Is Response Required check box is selected. Specify whether the response correlation pattern should be based on: <ul style="list-style-type: none"> ■ MessageID ■ CorrelationID ■ Dynamic Queue – Select this option if your WebSphere MQ implementation uses dynamic queues for response correlation. The MQ transport supports only temporary dynamic queues.
Auto-generate Correlation Value	This option is available only when the Is Response Required check box is selected. Select this check box to automatically generate a CorrelationID or MessageID.

Table 3–17 (Cont.) MQ Transport Configuration Options for Business Services

Option	To create or edit...
Model Queue	For Dynamic Queue Response Correlation Pattern only. Enter the name of the model queue used to generate the dynamic queue.
MQ Response URI	<p>This option is available only when the Is Response Required option is selected.</p> <p>The destination to which the response should be published. Enter a response URI in the same format as the endpoint URI: <code>mq://local_queue_name?conn=mq_connection_resource</code> or, if you are using dynamic queues: <code>mq://dynamic_queue_prefix?conn=mq_connection_resource</code></p> <p>The <i>dynamic_queue_prefix</i>, which is limited to 32 characters, is used to create the dynamic queue on the MQ server. The queue name becomes the prefix plus a unique ID. For example, if the <i>dynamic_queue_prefix</i> is <code>example</code>, the dynamic queue would be named something like <code>example123129083821</code>.</p> <p>You can also use an asterisk (*) as a wildcard in the dynamic queue response URI. For example: <code>mq://dynamic_queue_prefix*?conn=mq_connection_resource</code> <code>mq://dynamic_queue_prefix*</code> <code>mq://*</code></p> <p>If you do not provide a <i>dynamic_queue_name</i> in the URI, the transport uses the dynamic queue name generated by the MQ server. If you do not provide an explicit <i>mq_connection_resource</i> in the URI (best practice), the transport uses the <i>mq_connection_resource</i> from the endpoint URI.</p> <p>For more detailed information, see "MQ Transport" in the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBDV1117.</p>
Response Timeout	<p>This option is available only when the Is Response Required checkbox is selected.</p> <p>Enter the number of seconds to wait for a response before dropping the connection.</p>
Dispatch Policy	<p>Select a dispatch policy for this endpoint.</p> <p>Dispatch policy refers to the instance of WLS Work Manager that you want to use for the service endpoint.</p> <p>For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112.</p>
Dynamic Queue Pooling	<p>For Dynamic Queue Response Correlation Pattern only. Select this option if you want the service to use pooled connections to dynamic queues.</p> <p>If you want to use a separate connection pool for dynamic queues, consider configuring a dedicated MQ Connection resource for the dynamic queues.</p> <p>Do not select this option if you want to create a new dynamic queue instance on each request (and destroy the queue after the response).</p>

Table 3–17 (Cont.) MQ Transport Configuration Options for Business Services

Option	To create or edit...
Endpoint URI 'PUT' options	<p data-bbox="643 260 1308 287">Enter the MQ PUT message options from among the following:</p> <ul style="list-style-type: none"> <li data-bbox="643 302 1235 329">■ MQC.MQPMO_ALTERNATE_USER_AUTHORITY <li data-bbox="643 340 1101 367">■ MQC.MQPMO_DEFAULT_CONTEXT <li data-bbox="643 378 1097 405">■ MQC.MQPMO_FAIL_IF QUIESCING <li data-bbox="643 415 1070 443">■ MQC.MQPMO_LOGICAL_ORDER <li data-bbox="643 453 1070 480">■ MQC.MQPMO_NEW_CORREL_ID <li data-bbox="643 491 1026 518">■ MQC.MQPMO_NEW_MSG_ID <li data-bbox="643 529 1029 556">■ MQC.MQPMO_NO_CONTEXT <li data-bbox="643 567 1055 594">■ MQC.MQPMO_NO_SYNCPOINT <li data-bbox="643 604 937 632">■ MQC.MQPMO_NONE <li data-bbox="643 642 1105 669">■ MQC.MQPMO_PASS_ALL_CONTEXT <li data-bbox="643 680 1170 707">■ MQC.MQPMO_PASS_IDENTITY_CONTEXT <li data-bbox="643 718 1097 745">■ MQC.MQPMO_RESOLVE_LOCAL_Q <li data-bbox="643 756 1092 783">■ MQC.MQPMO_SET_ALL_CONTEXT <li data-bbox="643 793 1159 821">■ MQC.MQPMO_SET_IDENTITY_CONTEXT <li data-bbox="643 831 1005 858">■ MQC.MQPMO_SYNCPOINT <li data-bbox="643 869 992 896">■ MQC.MQPMO_VERSION_1 <li data-bbox="643 907 992 934">■ MQC.MQPMO_VERSION_2 <p data-bbox="643 947 1292 1024">You can use either " " or " + " to separate multiple options. For example, you can specify the following: MQC.MQPMO_LOGICAL_ORDER MQC.MQPMO_NEW_MSG_ID</p> <p data-bbox="643 1037 1357 1094">The MQ PUT message options are applied when the message is placed in the outbound queue.</p>
MQ Unrecognized Response URI	<p data-bbox="643 1178 1349 1283">Enter the URI representing the queue to which unrecognized response messages should be sent. Note that this setting is enabled only when the Auto-generate Correlation Value check box is selected.</p> <p data-bbox="643 1295 1336 1352">If you do not specify a value for this field, unrecognized response messages are deleted.</p>
Process RFH2 Headers	<p data-bbox="643 1367 1325 1451">Select this option to parse WebSphere MQ RFH2 headers from a message payload and automatically generate an RFH2Headers transport header containing the RFH2 data.</p> <p data-bbox="643 1463 1325 1514">If you do not select this option, the payload is passed through as received.</p>

3.1.18 MQ Transport Configuration Page (Proxy Services)

Before you use the MQ transport, you must configure a MQ Connection resource. See [Section 2.6, "Working with MQ Connections."](#) For more information about the MQ transport, see "MQ Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV1117>.

Use this page to configure transport settings for a proxy service using the native MQ transport protocol.

Table 3–18 MQ Transport Configuration Options for Proxy Services

Option	To create or edit...
Polling Interval	Enter a polling interval, in milliseconds. The default is 1000.
Is Response Required	Select this option to specify that a response is expected after an outbound message is sent.
Response Correlation Pattern	This option is available only when the Is Response Required check box is selected. Specify whether the response correlation pattern should be based on MessageID or CorrelationID .
MQ Response URI	This option is available only when the Is Response Required check box is selected. The destination to which the response should be published. Enter a response URI in the same format as the endpoint URI: mq://<local-queue-name>?conn=<mq-connection-resource-ref>
Response Message Type	This option is available only when the Is Response Required check box is selected. Select one of the following: <ul style="list-style-type: none"> ■ Bytes (for a stream of uninterpreted bytes) ■ Text (for text messages)
Client Response Timeout	This option is available only when the Is Response Required check box is selected. Enter the number of seconds to wait for a response before dropping the connection.
Dispatch Policy	Select a dispatch policy for this endpoint. Dispatch policy refers to the instance of WLS Work Manager that you want to use for the service endpoint. For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112 .
Backout Threshold	Enter a value representing the number of times the pipeline should retry a message before redirecting the message to the queue specified in the Dead Letter URI field. If you do not specify a value for this field, the message is redirected to the dead letter queue without attempting any retries.
MQ Dead Letter URI	Enter the URI of the dead letter queue to which request messages should be redirected after attempting the number of retries specified in the Backout Threshold field. If you do not specify a value for this field, the message is discarded after retrying the number of times specified in the Backout Threshold field. The Dead Letter URI uses the same format as the EndPoint URI.

Table 3–18 (Cont.) MQ Transport Configuration Options for Proxy Services

Option	To create or edit...
Endpoint URI 'GET' options	<p data-bbox="651 260 1321 287">Enter the MQ GET message options from among the following:</p> <ul style="list-style-type: none"> <li data-bbox="651 302 1198 329">■ MQC.MQGMO_ACCEPT_TRUNCATED_MSG <li data-bbox="651 338 1149 365">■ MQC.MQGMO_ALL_MSGS_AVAILABLE <li data-bbox="651 373 1052 401">■ MQC.MQGMO_BROWSE_FIRST <li data-bbox="651 409 1052 436">■ MQC.MQGMO_BROWSE_NEXT <li data-bbox="651 445 1073 472">■ MQC.MQGMO_COMPLETE_MSG <li data-bbox="651 480 992 508">■ MQC.MQGMO_CONVERT <li data-bbox="651 516 1110 543">■ MQC.MQGMO_FAIL_IF_QUIESCING <li data-bbox="651 552 943 579">■ MQC.MQGMO_LOCK <li data-bbox="651 588 1081 615">■ MQC.MQGMO_LOGICAL_ORDER <li data-bbox="651 623 1154 651">■ MQC.MQGMO_MARK_BROWSE_CO_OP <li data-bbox="651 659 1146 686">■ MQC.MQGMO_MARK_SKIP_BACKOUT <li data-bbox="651 695 1065 722">■ MQC.MQGMO_NO_SYNCPOINT <li data-bbox="651 730 951 758">■ MQC.MQGMO_NONE <li data-bbox="651 766 987 793">■ MQC.MQGMO_NO_WAIT <li data-bbox="651 802 1016 829">■ MQC.MQGMO_SYNCPOINT <li data-bbox="651 837 1203 865">■ MQC.MQGMO_SYNCPOINT_IF_PERSISTENT <li data-bbox="651 873 984 900">■ MQC.MQGMO_UNLOCK <li data-bbox="651 909 1195 936">■ MQC.MQGMO_UNMARK_BROWSE_CO_OP <li data-bbox="651 945 1203 972">■ MQC.MQGMO_UNMARK_BROWSE_HANDLE <li data-bbox="651 980 1198 1008">■ MQC.MQGMO_UNMARKED_BROWSE_MSG <li data-bbox="651 1016 1003 1043">■ MQC.MQGMO_VERSION_1 <li data-bbox="651 1052 1003 1079">■ MQC.MQGMO_VERSION_2 <li data-bbox="651 1087 1003 1115">■ MQC.MQGMO_VERSION_3 <li data-bbox="651 1123 938 1150">■ MQC.MQGMO_WAIT <p data-bbox="651 1255 1300 1312">You can use either " " or "+" to separate multiple options. For example, you can specify the following:</p> <p data-bbox="651 1320 1354 1377">MQC.MQGMO_ACCEPT_TRUNCATED_MSG MQC.MQGMO_LOCK</p> <p data-bbox="651 1386 1360 1440">The MQ GET message options are applied when reading a message from the inbound queue.</p>
Process RFH2 Headers	<p data-bbox="651 1457 1328 1539">Select this option to parse WebSphere MQ RFH2 headers from a message payload and automatically generate an RFH2Headers transport header containing the RFH2 data.</p> <p data-bbox="651 1547 1328 1602">If you do not select this option, the payload is passed through as received.</p>

3.1.19 SB Transport Configuration Page (Business Services)

Use this page to configure transport settings for a proxy service using the SB (Service Bus) transport protocol. For more information about the SB transport, see "SB Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV997>.

Table 3–19 SB Transport Configuration Options for Business Services

Option	To create or edit...
Dispatch policy	Select the instance of Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists. See "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112 .
Timeout	The amount of time in seconds it takes the service to time out. Note: The timeout will be ignored when the quality of service is Exactly-Once.
Service Account	Click Browse and select a service account from the list displayed. If no service account is specified, an anonymous subject is used.

3.1.20 SB Transport Configuration Page (Proxy Services)

Use this page to configure transport settings for a proxy service using the SB (Service Bus) transport protocol. For more information about the SB transport, see "SB Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV997>.

Table 3–20 SB Transport Configuration Options for Proxy Services

Option	To create or edit...
Dispatch Policy	Select a dispatch policy for this endpoint or use the default dispatch policy. Dispatch policy refers to the instance of WLS Work Manager that you want to use for the service endpoint to process the request. See "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112 .
Use SSL	When specified, requests must be sent over an SSL connection. However, unsecured connections are not forbidden. The administrator must close all unsecured protocols on the server (for example, t3 or http) to strictly enforce secured client connections.

3.1.21 SFTP Transport Configuration Page (Business Services)

Use this page to configure transport settings for a proxy service using the sftp transport protocol. For more information about the SFTP transport, see "SFTP Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV975>.

Table 3–21 SFTP Transport Configuration Options for Business Services

Option	To create or edit...
User Authentication	Select one of the following: <ul style="list-style-type: none"> ▪ Username Password Authentication - Specifies that a static service account is associated with this authentication method and the client is authenticated using the provided credentials. ▪ Host Based Authentication - Specifies that a user name and service key provider is required to use this authentication method. Any user connecting from a known host is authenticated using the private key of the host. ▪ Public Key Authentication - Specifies that a user name and service key provider is required to use this authentication method. Every user has their own private key.
Service Account	Enter the service account for the user, or click Browse to select service accounts from a browser.
Service Key Provider	This option is available only when Host Based or Public Key Authentication is selected. Enter a service key provider in the Service Key Provider field. You can click Browse to select service key providers from a browser. This is a required field.
Username	This option is available only when Host Based or Public Key Authentication is selected. Enter the user name.
Timeout	Enter the socket timeout interval, in seconds, before the connection is dropped. If you enter 0, there is no timeout. The default value is 60.
Prefix for destination File Name	Enter the prefix for the file name under which the file is stored on the remote server.
Suffix for destination File Name	Enter the suffix for the file name under which the file is stored on the remote server.
Request Encoding	Accept the default UTF-8 as the character set encoding for requests in SFTP transports.

3.1.22 SFTP Transport Configuration Page (Proxy Services)

Use this page to configure transport settings for a proxy service using the sftp transport protocol. For more information about the SFTP transport, see "SFTP Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV975>.

Table 3–22 SFTP Transport Configuration Options for Proxy Services

Option	To create or edit...
User Authentication	Select one of the following: <ul style="list-style-type: none"> ▪ Username Password Authentication - Specifies that a static service account is associated with this authentication method and the client is authenticated using the provided credentials. ▪ Host Based Authentication - Specifies that a user name and service key provider is required to use this authentication method. Any user connecting from a known host is authenticated using the private key of the host. ▪ Public Key Authentication - Specifies that a user name and service key provider is required to use this authentication method. Every user has their own private key.

Table 3–22 (Cont.) SFTP Transport Configuration Options for Proxy Services

Option	To create or edit...
Service Account	Enter the service account for the user, or click Browse to select service accounts from a browser.
Service Key Provider	This option is available only when Host Based or Public Key Authentication is selected. Enter a service key provider in the Service Key Provider field. You can click Browse to select service key providers from a browser. This is a required field.
Username	This option is available only when Host Based or Public Key Authentication is selected. Enter the user name.
Pass By Reference	Select this check box to stage the file in the archive directory and pass it as a reference in the headers.
Remote Streaming	Select this check box to stream the SFTP files directly from the remote server at the time of processing. When you select this option, the archive directory is the remote directory on the remote SFTP server machine. Therefore, you should specify the archive directory as relative to the SFTP user directory.
File Mask	Enter the regular expression for the files to be picked. The default is *. *.
Managed Server	This field is available only in a clustered domain. Select the Managed Server to act as the polling server. All of the Managed Servers can process the message, but only one can poll for the message.
Polling Interval	Enter the interval in seconds at which the file is polled from the specified location. The default is 60.
Read Limit	Specify the maximum number of messages to read per polling sweep. Enter 0 to specify no limit. The default is 10.
Post Read Action	Select what happens to a message after it has been read. <ul style="list-style-type: none"> ■ Archive - The message is archived. ■ Delete - The message is deleted.
Archive Directory	Specify the path to the archive location if the Post Read Action option is set to Archive . This field is required if the Pass By Reference option is selected. Note: The Archive, Download, and Error directories are absolute paths, and they are automatically created. If you specify a relative path, the files are created relative to the Java process that starts the Oracle WebLogic Server.
Download Directory	Enter the directory on your local machine where files are downloaded during the file transfer. Note: The Archive, Download, and Error directories are absolute paths, and they are automatically created. If you specify a relative path, the files are created relative to the Java process that starts the Oracle WebLogic Server.
Error Directory	Enter the location where messages are posted if there is a problem. Note: The Archive, Download, and Error directories are absolute paths, and they are automatically created. If you specify a relative path, the files are created relative to the Java process that starts the Oracle WebLogic Server.

Table 3–22 (Cont.) SFTP Transport Configuration Options for Proxy Services

Option	To create or edit...
Request Encoding	Accept the default UTF–8 as the character set encoding for requests in SFTP transports.
Scan Subdirectories	Select this check box to recursively scan all directories
Sort By Arrival	Select this check box to deliver events in the order of arrival.
Timeout	Enter the socket timeout interval, in seconds, before the connection is dropped. If you enter 0, there is no timeout. The default value is 60.
Retry Count	Specify the number of retries for SFTP connection failures.

3.1.23 SOA-DIRECT Transport Configuration Page (Business Services)

Table 3–23 describes the transport-specific configuration options for the SOA-DIRECT transport.

Table 3–23 SOA-DIRECT Transport Configuration

Property	Description
JNDI Service Account	Optional. Specifies the security credentials for the JNDI lookup of the target SOA service. The service account must be static. Click Browse and select a service account. If you do not specify a service account, an anonymous subject is used.
Role	<p>Required. Identifies the communication pattern the service uses. Select one of the following options:</p> <ul style="list-style-type: none"> ■ Synchronous Client (default) – In this role, because there is no asynchronous callback, the Callback Proxy option is disabled. The WS-Addressing Version option is also disabled. ■ Asynchronous Client – In this role, because asynchronous callback is usually required, you can identify a Callback Proxy, and you must select a WS-Addressing Version. The asynchronous option is enabled only when the WSDL service is of type SOAP. ■ Service Callback – This role is for returning the asynchronous callback to an SOA service after an external service invocation. <p>There is no load balancing or failover for Callback services.</p>
Callback Proxy	<p>Optional. Enabled only for the Asynchronous Client role.</p> <p>This option lets you specify the proxy service that receives callbacks. When you select a callback proxy, if no WS-Addressing is provided by the request or the proxy service pipeline, Oracle Service Bus automatically populates the ReplyTo property in the SOAP header. You must select a WSDL proxy service that uses the SB transport (for RMI), and the callback proxy service must understand WS-Addressing.</p> <p>WS-Addressing properties that are sent in the request or set in the proxy service pipeline are used instead of the callback proxy you set in this option.</p> <p>If you do not specify a Callback Proxy, and the request does not contain ReplyTo properties, you must provide ReplyTo properties in the SOAP header through the proxy service pipeline.</p>
Fault Proxy	This option is not currently supported. You must configure your callback services to handle faults in an asynchronous pattern.

Table 3–23 (Cont.) SOA-DIRECT Transport Configuration

Property	Description
WS-Addressing Version	<p>Required. Enabled only for the Asynchronous Client role.</p> <p>Specify the default WS-Addressing version to use when no WS-Addressing is provided in the request or the proxy service pipeline. WS-Addressing properties that are sent in the request or set in the proxy service pipeline are used instead of the WS-Addressing version you set in this option.</p> <p>For WS-Addressing version mismatches between environments, perform any necessary transformations in the proxy service pipeline.</p>
Dispatch Policy	<p>Select the instance of Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists.</p> <p>For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112.</p>
Pass Caller's Subject	<p>Optional. Select this option to have Oracle Service Bus pass the authenticated subject from the proxy service when invoking the SOA service. The Invocation Service Account option, an alternative to Pass Caller's Subject, is disabled when you select this option.</p> <p>Note: Make sure that domain trust is enabled between client and target server if they are in different domains. For more information, see "Important Information Regarding Cross-Domain Security Support" in <i>Oracle Fusion Middleware Securing Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=SECMG403.</p>
Invocation Service Account	<p>Optional. Alternative to Pass Caller's Subject, which lets you specify custom security credentials by selecting a service account for RMI invocation. You can specify any type of service account (Pass Through, Static, Mapping).</p> <p>Click Browse and select a service account. If you do not specify a service account, an anonymous subject is used.</p>

3.1.24 Tuxedo Transport Configuration Page (Business Services)

Use this page to configure transport settings for a proxy service using the Tuxedo transport protocol. For more information about the Tuxedo transport, see "Tuxedo Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV1200>.

Table 3–24 Tuxedo Transport Configuration Options for Business Services

Option	To create or edit...
Field Table Classes	Enter the name of the class or classes describing the FML/FML32 buffer received. These are used for the FML/FML32-to-XML conversion routines to map field names to element names. This is a space separated list of fully qualified class names.

Table 3–24 (Cont.) Tuxedo Transport Configuration Options for Business Services

Option	To create or edit...
View Classes	Enter the name of the class or classes describing the VIEW/VIEW32 buffer received or sent. These are used for the VIEW-to-XML or VIEW32-to-XML conversion routines to map field names to element names. This is a space separated list of fully qualified class names.
Classes Jar	Select a JAR Resource that contains a JAR file with the FML/FML32 or VIEW/VIEW32 classes necessary for this endpoint operation.
Remote Access Point(s)	<p>Select a remote access point from the drop down list that is associated with the Import. The list contains remote access points configured in WTC. A business service cannot be created if there is no associated remote access point.</p> <p>If no remote access points exist or to create a new one, select New. Enter the corresponding Access Point Name and Network Address in the adjacent fields. Upon validation of the endpoint, the access point is added to the WTC configuration for each WTC server. If no WTC server exists, one is created.</p> <p>If more than one URI has been specified, there will be one remote access point field per URI and the URI displays for informative purposes. If more than one URI exists, each requires a different remote access point. If the URI specified already corresponds to an existing WTC resource, the corresponding remote access point displays, but cannot be modified.</p>
Local Access Point(s)	<p>This field appears only when you select New in the Remote Access Point field.</p> <p>From the list, select a local access point to be associated with the newly created remote access point. If none exist or to create a new one, select New. Enter the corresponding Local Access Point Name and Local Network Address in the adjacent fields.</p>
Request Buffer Type	Select the type of buffer that the remote Tuxedo service will receive.
Request Buffer Subtype	This option is enabled if the previous Request Buffer Type value is VIEW or VIEW32. Enter the buffer subtype with which to associate the request buffer.
Response Required?	Select this check box to indicate a bidirectional call. If not checked, the underlying <code>tpcall</code> is invoked with <code>TPNOREPLY</code> flag, and a null response is posted asynchronously.
Suspend Transaction?	Select this check box to suspend the transaction, if it exists. This is useful when the remote service does not support transactions.
Dispatch Policy	<p>Select the instance of Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists.</p> <p>This Work Manager is used to asynchronously post a null reply in the case of a one-way invocation. See "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112.</p>
Request Encoding	Specify a character set encoding for requests in Tuxedo transports.
Response Encoding	Specify a character set encoding for responses in Tuxedo transports.
Timeout	Specify the maximum amount of time (in seconds) that the transport run time waits for replies; an integer value that is greater than or equal to 0. If not specified or set to zero (default), replies will time out at <code>BLOCKTIME</code> , the maximum number of seconds that the local WTC access point allows for a blocking call.

Table 3–24 (Cont.) Tuxedo Transport Configuration Options for Business Services

Option	To create or edit...
Transformation Style	<p>Select one of the following:</p> <ul style="list-style-type: none"> ■ None - (default) The order of fields may not be respected. ■ Ordered - The fields are presented with all their occurrences in the correct order. ■ Ordered and Grouped - If the fields are logically structured as records, the fields are ordered by occurrence and grouped by record.

3.1.25 Tuxedo Transport Configuration Page (Proxy Services)

Use this page to configure transport settings for a proxy service using the Tuxedo transport protocol. For more information about the Tuxedo transport, see "Tuxedo Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV1200>.

Table 3–25 Tuxedo Transport Configuration Options for Proxy Services

Option	To create or edit...
Field Table Classes	Enter the name of the class or classes describing the FML/FML32 buffer received. These are used for the FML/FML32-to-XML conversion routines to map field names to element names. This is a space separated list of fully qualified class names.
View Classes	<p>Enter the name of the class or classes describing the VIEW/VIEW32 buffer received or sent. These are used for the VIEW-to-XML or VIEW32-to-XML conversion routines to map field names to element names. This is a space separated list of fully qualified class names.</p> <p>X_C_TYPE and X_COMMON Tuxedo buffer types are handled in the same manner as VIEW/VIEW32 buffers.</p> <p>If an incoming request contains a VIEW, then the corresponding VIEW class should be specified in the Oracle Service Bus CLASSPATH.</p>
Classes Jar	Select a JAR resource that contains a JAR file with the FML/FML32 or VIEW/VIEW32 classes necessary for this endpoint operation.
Local Access Point	<p>Select a local access point from the list that is associated with the export. The list contains local access points configured in WTC. A proxy service cannot be created if there is not an associated local access point.</p> <p>If no local access points exist or to create a new one, select New. Enter the corresponding Local Access Point Name and Local Network Address in the adjacent fields. Upon validation of the endpoint, the access point is added to the WTC configuration for each WTC server. If no WTC server exists, one is created.</p> <p>You can enter an existing access point name after selecting the New option. This causes the existing information to be updated with the new parameters. You can change only the host name and port number.</p>

Table 3–25 (Cont.) Tuxedo Transport Configuration Options for Proxy Services

Option	To create or edit...
Remote Access Point	<p>This field appears only when you select New in the Local Access Point field.</p> <p>From the list, select a remote access point to be associated with the newly created local access point. If none exist or to create a new one, select New. Enter the corresponding Access Point Name and Network Address in the adjacent fields.</p> <p>You can enter an existing access point name after selecting the New option. This causes the existing information to be updated with the new parameters. You can change only the host name and port number.</p> <p>The remote access point will also be the authentication principal for the WTC connection for inbound requests. Optionally, you can create a user with the same access point ID in the default security realm to allow incoming calls. To do so, select Yes from the Create User? list. The password will be randomly generated using a temporary variable to avoid security issues.</p>
Reply Buffer Type	<p>This option is available only if the Response Required? field is selected. Select the type of buffer that the remote Tuxedo client will receive.</p>
Reply Buffer Subtype	<p>This option is available only when the Response Required? option is selected and the Reply Buffer Type value is VIEW or VIEW32. Enter the buffer subtype with which to associate the reply buffer.</p>
Response Required?	<p>Select this check box if this service is expected to send a response. The default status is that this option is selected.</p> <p>This option is cleared and the unavailable if the service type is Messaging Service and the response message type is None.</p>
Request Encoding	Specify a character set encoding for requests in Tuxedo transports.
Response Encoding	Specify a character set encoding for responses in Tuxedo transports.
Transformation Style	<p>Select one of the following:</p> <ul style="list-style-type: none"> ■ None - (default) The order of fields may not be respected. ■ Ordered - The fields are presented with all their occurrences in the correct order. ■ Ordered and Grouped - If the fields are logically structured as records, the fields are ordered by occurrence and grouped by record.

3.1.26 WS Transport Configuration Page (Business Services)

Use this page to configure transport settings for a proxy service using the WS transport protocol. For more information about the WS transport, see "WS Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV1083>.

Table 3–26 WS Transport Configuration Options for Business Services

Option	To create or edit...
Response Timeout	<p>Enter the number of seconds to wait for a response.</p> <p>Leaving this field blank indicates that there is no response timeout. The business service will wait for the duration of the sequence timeout configured in the RM policy.</p> <p>If you enter a zero (0) value, there is no timeout; as such, it will never time out.</p>
Service Account	<p>Enter a service account or click Browse to select one from the list displayed.</p> <p>The service account specifies the credentials to use when there is an HTTP basic authentication policy on this service.</p>
Queue Error Messages	Select the check box to enable sending error messages to the configured error queue.
Error Queue URI	<p>This option is available only when the Queue Error Messages check box is selected.</p> <p>Enter the URI of JMS queue for storing error messages, in the following format:</p> <pre>jms://host:port/connFactoryJndiName/queueJndiName</pre>
JMS Error Queue Service Account	<p>This option is available only when the Queue Error Messages check box is selected.</p> <p>Enter a service account or click Browse to select one from the list displayed.</p> <p>The service account specifies the credentials to use for JNDI lookups and sending error messages to the error queue.</p>
Use SSL for Error Queue	<p>This option is available only when the Queue Error Messages check box is selected.</p> <p>Select the check box to use SSL for connecting to the error queue.</p>

3.1.27 WS Transport Configuration Page (Proxy Services)

Use this page to configure transport settings for a proxy service using the WS transport protocol. For more information about the WS transport, see "WS Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV1083>.

Table 3–27 WS Transport Configuration Options for Proxy Services

Option	To create or edit...
Dispatch Policy	<p>Select a dispatch policy for this endpoint or use the default dispatch policy.</p> <p>Dispatch policy refers to the instance of WLS Work Manager that you want to use for the service endpoint. See "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> at http://www.oracle.com/pls/as1111/lookup?id=CNFGD112.</p>
Retry Count	<p>The number of times to retry delivery of a message to the pipeline.</p> <p>If an unhandled exception occurs in the request pipeline of a proxy service, the incoming WS transport message will be redelivered to the pipeline up to the number of times specified by the retry count. This value is important for reliably processing WS transport messages.</p>

Table 3–27 (Cont.) WS Transport Configuration Options for Proxy Services

Option	To create or edit...
Retry Delay	The number of seconds the system pauses before retrying to send a message to the pipeline after an error.

User Interface Reference

This section describes all the views, dialogs, wizards, and other user interface objects in the Oracle Service Bus plug-ins.

- [Section 4.1, "Alerts"](#)
- [Section 4.2, "Business Service Configuration"](#)
- [Section 4.3, "Proxy Service Configuration"](#)
- [Section 4.4, "Oracle Service Bus Configurations and Projects"](#)
- [Section 4.5, "Custom Resources"](#)
- [Section 4.6, "Export Wizard"](#)
- [Section 4.7, "Import Wizard"](#)
- [Section 4.8, "JNDI Providers"](#)
- [Section 4.9, "Proxy Servers"](#)
- [Section 4.10, "Message Flow Design Palette"](#)
- [Section 4.11, "Message Flow Editor"](#)
- [Section 4.12, "Modify JAR Dependencies Dialog"](#)
- [Section 4.13, "SMTP Servers"](#)
- [Section 4.14, "UDDI Registry Configuration Page"](#)
- [Section 4.15, "Outline view - Oracle Service Bus"](#)
- [Section 4.16, "Resource Management"](#)
- [Section 4.17, "New Service Key Provider Resource"](#)
- [Section 4.18, "New WS-Policy"](#)
- [Section 4.19, "Service Accounts"](#)
- [Section 4.20, "Expression Editors"](#)
- [Section 4.21, "New XSL Transformation"](#)
- [Section 4.22, "Split-Join User Interface Reference"](#)

4.1 Alerts

The following pages are provided for managing alerts:

- [Section 4.1.1, "Alert Destination Editor"](#)

- [Section 4.1.2, "Edit E-mail Recipient Page"](#)
- [Section 4.1.3, "Edit JMS Destination Page"](#)

4.1.1 Alert Destination Editor

An alert destination is a destination address for alert notifications in Oracle Service Bus. Use this page to configure an alert destination resource.

Table 4–1 Alert Destination Editor Options

Option	Description
Description	Enter a description for this alert destination.
SNMP Trap	If you specify SNMP Trap , alerts are sent as SNMP traps, and can be processed by any third-party enterprise monitoring systems (ESM).
Reporting	If you specify Reporting , alerts are sent to the Oracle Service Bus Reporting module and can be captured using a custom reporting provider that can developed using the reporting APIs provider by Oracle Service Bus. This allows third-parties to receive and process alerts in custom Java code.
Alert Logging	If you select Yes, alerts sent to this alert destination are logged to the alert log.

Continue in the Edit Alert Destination page. Click **Add** to add e-mail and JMS recipients to an alert destination. See:

- [Section 4.1.2, "Edit E-mail Recipient Page"](#)
- [Section 4.1.3, "Edit JMS Destination Page"](#)

4.1.2 Edit E-mail Recipient Page

Use this page to configure the destination target for an alert sent via e-mail (that is, using the e-mail transport).

Table 4–2 E-mail Recipient Options

Option	Description
Mail Recipients	<p>Enter an e-mail recipient in the format: mailto:username@hostname</p> <p>You can specify multiple e-mail recipients by entering the user names and hostnames in a comma-separated list. For example, mailto:username@hostname [,username_1@hostname_1]...[,username_n@hostname_n]</p> <p>Only the first mail recipient needs to be prefixed with the text "mailto:". However doing so is optional; the code will add it if it is missing.</p>
SMTP Server	Select the name of the SMTP server for the outgoing e-mail. This field is not required if Mail Session is selected in the next field. These fields are mutually exclusive; it is an error to configure both.
Mail Session	Select an available mail session. This field is not required if an SMTP Server is selected in the previous field. These fields are mutually exclusive; it is an error to configure both.

Table 4–2 (Cont.) E-mail Recipient Options

Option	Description
From Name	Provide a sender's name for the alert notification. This field is optional.
From Address	Provide a valid e-mail address. This field is required if a value for the From Name field is specified.
Reply To Name	Provide a name to which a reply may be addressed. This field is optional.
Reply To Address	Provide an e-mail address to which a reply may be sent. This field is required if a value for the Reply To Name field is specified.
Connection Timeout	Enter the number of seconds a connection must wait for a response from the server before timing out. The default value is 0.
Request Encoding	Enter a character set encoding value. The default encoding value is iso-8859-1.

4.1.3 Edit JMS Destination Page

Use this page to configure the destination target for an alert sent via JMS (that is, using the JMS transport).

Table 4–3 JMS Destination Options

Option	Description
Destination URI	Enter a JMS destination URI in the format: <code>jms://host:port/connection_factory/jndi_destination</code>
Destination Type	Select Queue or Topic .
Message Type	Select Bytes or Text .
Request Encoding	Enter a character set encoding value. The default encoding value is iso-8859-1.

4.2 Business Service Configuration

You configure business services in the business service editor after you create the business service. You can view and modify those settings in the Business Service editor: The business service configuration pages are:

- [Section 4.2.1, "Business Service General Configuration Page"](#)
- [Section 4.2.2, "Business Service Message Type Configuration Page"](#)
- [Section 4.2.3, "Business Service SOAP Binding Configuration Page"](#)
- [Section 4.2.4, "Business Service Transport Configuration Page"](#)
- [Section 4.2.5, "Business Service Message Handling Configuration Page"](#)
- [Section 4.2.6, "Business Service - Service Policy Configuration Page"](#)
- [Section 4.2.7, "Business Service Security Configuration Page"](#)

4.2.1 Business Service General Configuration Page

Use the Business Service General Configuration page to specify general configuration settings for a business service.

4.2.1.1 Business Service Editor Options

The following table describes the options in the editor:

Table 4–4 Business Service Editor Options

Option	Description
Description	Enter or modify a description for this service.
Service Type (editor only)	<p>This option shows the service type of the business service. You can change only some of the properties of some of the service types:</p> <ul style="list-style-type: none"> ■ WSDL Web Service - Select this option to create a business service based on a WSDL. Then, enter the WSDL name, qualified by its path (for example, myProject/myFolder/myWSDL). Alternatively, click Browse to select a WSDL resource. You can also specify a different WSDL, but by doing so, you are effectively creating a new service and you will have to configure it as if it were a new service. <p>(port or binding) - Enter the name of a port (defined in the WSDL) to describe an actual transport address, or enter the name of a binding (defined in the WSDL) to map to a transport address. If you use Browse to select a WSDL, as described above, the Select a WSDL Definition dialog lists any ports and bindings defined in the WSDL. When you choose a port or a binding on that page, the (port or binding) field is filled with the selected name.</p> <ul style="list-style-type: none"> ■ Transport Typed Service - Select this option to create a service that uses the EJB, JEJB, or Flow (Split-Join) transport. ■ Messaging Service - Select this option to create a service that exchanges messages of different content-types. These exchanges can be either request/response or one-way. They can also have a response with no request when used with the HTTP 'GET' option for the HTTP transport. Unlike Web Services, the content-type of the request and response need not be the same. ■ Any SOAP Service - Select this option to create a SOAP service that does not have an explicitly defined, concrete interface. You can change the SOAP version (SOAP 1.1 or SOAP 1.2) ■ Any XML Service - Select this option to create an XML service that does not have an explicitly defined, concrete interface. <p>HTTP GET is only supported for messaging services and this service type.</p>

4.2.2 Business Service Message Type Configuration Page

Use the Business Service Message Type Configuration page to configure message types for a business service whose type is Messaging Service.

The binding definition for messaging services consists of configuring the content-types of the messages that are exchanged. The content-type for the response does not need to be the same as for the request; therefore, the response is configured separately (for example, the service could accept an MFL message and return an XML acknowledgment receipt).

Note: E-mail, File, FTP, or SFTP transport business services whose type is Messaging Service support one-way messaging *only*; the **Response Message Type** should be **None**. If you select an option other than **None**, the file, ftp, or sftp protocol will not be available on the Transport Configuration page.

Table 4–5 Message Type Business Service Options

Option	Description
Request Message Type	<p>Select a message type for the request message:</p> <ul style="list-style-type: none"> ■ None - Select this option if there is no request message (HTTP GET example) ■ Binary - Select this option if the content-type of the message is unknown or not important. ■ Text - Select this option if the message can be restricted to text. ■ MFL - Select this option if the message is a binary document conforming to an MFL definition. Enter the MFL file name (qualified by its path), or click Browse to select a file. You can configure only one MFL file. Note: To support multiple MFL files, define the content as binary or text and use the MFL action in the message flow to convert to XML. ■ XML - Select this option if the message is an XML document. Enter the XML file name (qualified by its path), or click Browse to select a file. Optionally provide some type information by declaring (in the element or type field) the XML schema type of the XML document exchanged.
Response Message Type	<p>Select a message type for the response message:</p> <ul style="list-style-type: none"> ■ None - Select this option if there is no response message. ■ Binary - Select this option if the content-type of the message is unknown or not important. ■ Text - Select this option if the message can be restricted to text. ■ MFL - Select this option if the message is a binary document conforming to an MFL definition. Enter the MFL file name (qualified by its path), or click Browse to select a file. You can configure only one MFL file. Note: To support multiple MFL files, define the content as binary or text and use the MFL action in the message flow to convert to XML. ■ XML - Select this option if the message is an XML document. Enter the XML file name (qualified by its path), or click Browse to select a file. Optionally provide some type information by declaring (in the element or type field) the XML schema type of the XML document exchanged.

4.2.3 Business Service SOAP Binding Configuration Page

Use Business Service SOAP Binding Configuration page to configure the SOAP Binding for a business service based on a WSDL.

Select or deselect **Enforce WS-I Compliance** to specify whether or not the service is to conform to the Basic Profile defined by the Web Services Interoperability Organization. This option is available for or SOAP 1.1 services only

When a service is marked WS-I compliant, checks are performed against the messages sent to and from that service.

4.2.4 Business Service Transport Configuration Page

Use the Business Service Transport Configuration page to select, review, or change the service's transport protocol and to set, review, or change general transport configuration settings.

Outbound transport-level security applies to the connections between Oracle Service Bus proxy services and business services. For more information about transport-level security, see "Configuring Transport-Level Security" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV1557>.

Table 4-6 Business Service Transport Configuration Options

Option	Description
Protocol	<p>Select a transport protocol from the list. The protocols available differ, depending on the service type:</p> <ul style="list-style-type: none"> ■ WSDL Web Service: soa-direct, bpel-10g, dsp, http, jca, jms, sb, soa-direct, ws ■ Transport-Typed Service: ejb, flow, jejb ■ Messaging Service: email, file, ftp, http, jms, mq (if available), sftp, tuxedo ■ Any SOAP Service: dsp, http, jms, sb ■ Any XML Service: dsp, email, file, ftp, http, jms, mq (if available), sb, sftp, tuxedo
Load Balancing Algorithm	<p>Select one of these load-balancing algorithms:</p> <ul style="list-style-type: none"> ■ Round-robin - This algorithm dynamically orders the URLs that you enter in the Endpoint URI field for this business service. If the first one fails, it tries the next one, and so on until the retry count is exhausted. For every new message, there is a new order of URLs. ■ Random - This algorithm randomly orders the list of URLs that you enter in the Endpoint URI field for this business service. If the first one fails, it tries the next one, and so on until the retry count is exhausted. ■ Random-weighted - This algorithm randomly orders the list of URLs that you enter in the Endpoint URI field for this business service, but some are retried more than others based on the value you enter in the Weight field. ■ None - This algorithm orders the list of URLs that you enter in the Endpoint URI field for this business service from top to bottom.

Table 4–6 (Cont.) Business Service Transport Configuration Options

Option	Description
Endpoint URI	<p data-bbox="610 260 1446 394">Enter an endpoint URL in the format based on the transport protocol you selected in the Protocol field. Following are descriptions of the URI formats for each transport. Optional URI elements are shown in brackets []. For more information on transport URIs, see the respective transport chapters in the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i>.</p> <ul style="list-style-type: none"> <li data-bbox="610 407 1446 485">■ bpel-10g - protocol://host[:port] [/protocol-path]/domain/process[/version[/partnerlink/role]] <li data-bbox="610 499 1248 525">■ dsp - t3://dsp-ip-address:port/dsp-app-name <li data-bbox="610 539 1446 699">■ ejb - ejb:provider:jndiname <p data-bbox="659 579 1446 632">In the URI, <i>provider</i> is the name of the JNDI provider resource, and <i>JNDIname</i> is the JNDI name in the JNDI server for the EJB.</p> <p data-bbox="659 646 1446 699">If the JNDI provider is located on the same server, the JNDI provider need not be specified. The URI then would be <code>ejb::jndiname</code></p> <li data-bbox="610 714 1446 909">■ email - You can enter multiple endpoint URIs in any combination of the following formats: <pre data-bbox="659 779 1446 909">mailto:name@domain_name.com mailto:name@domain_name.com?smtp=smtp_server_resource mailto:name@domain_name.com?mailsession=jndi_mail_session</pre> <li data-bbox="610 924 1016 949">■ file - file:///root-dir/dir1 <li data-bbox="610 963 1127 989">■ ftp - ftp://hostname:port/directory <li data-bbox="610 1003 1446 1071">■ http - http://host:port/someService <p data-bbox="659 1041 1446 1071">The HTTP transport supports both HTTP and HTTPS endpoints.</p> <li data-bbox="610 1085 1127 1110">■ jca - jca://<resource_adapter_jndi> <li data-bbox="610 1125 1446 1230">■ jejb - jndi_provider:jndi_ejb_name <p data-bbox="659 1163 1446 1188">EJB 2.1 example: <code>jejb:myProvider:osb.jejb.myJejbBiz21</code></p> <p data-bbox="659 1203 1446 1230">EJB 3.0 example: <code>jejb:myProvider:myBiz31#osb.jejb.myJejbBiz</code></p> <li data-bbox="610 1245 1446 1587">■ jms - jms://host:port[,host:port]*/connection_factory/jndi_destination <p data-bbox="659 1310 1446 1383">To target a JMS destination to multiple servers, use the following URI format: <code>jms://host1:port,host2:port/connection_factory/jndi_destination</code></p> <p data-bbox="659 1398 1446 1451">You can also omit host and port from the URI to have the lookup performed on the local machine. For example:</p> <pre data-bbox="659 1470 1446 1495">jms:///connection_factory/jndi_destination</pre> <p data-bbox="659 1509 1446 1587">In a cluster: The host names in the JMS URI must exactly match the host names of the cluster servers as they are configured in Oracle WebLogic Server.</p>

Table 4–6 (Cont.) Business Service Transport Configuration Options

Option	Description
Endpoint URI (continued)	<ul style="list-style-type: none"> <li data-bbox="532 260 1369 659"> <p>■ mq - <code>mq://local-queue-name?conn=mq-connection-resource-ref</code></p> <p><code>local-queue-name</code> is the name of the MQ queue from which the business service reads messages.</p> <p><code>mq-connection-resource-ref</code> is the path (project/folder) and name of the MQ connection resource; for example, <code>default/my_MQconnection</code>.</p> <p>Note: The Endpoint URI cannot contain spaces, so do not create MQ Connection resources or projects/folders with spaces in the names.</p> <p>To make the MQ transport available in Oracle Service Bus, see "MQ Transport" in the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBDV1117.</p> <li data-bbox="532 674 1369 831"> <p>■ sb - <code>sb://<jndi_provider_name/>service_name</code></p> <p><code>jndi_provider_name</code> (optional) is the name of the Oracle Service Bus JNDI provider resource. When omitted, the default context is used.</p> <p><code>service_name</code> is a target service and corresponds to the remote proxy service URI.</p> <li data-bbox="532 846 1068 873"> <p>■ sftp - <code>sftp://hostname:port/directory</code></p> <li data-bbox="532 888 1369 1031"> <p>■ soa-direct - <code>[protocol://authority]/default/compositeName[!versionNumber[*label]]/serviceName</code></p> <p>The protocol and authority are optional when the SOA services are co-located on the same server as Oracle Service Bus.</p> <li data-bbox="532 1045 1369 1430"> <p>■ tuxedo - <code>tuxedo:resourcename/remotename</code></p> <p><code>tuxedo-queue:sendQSpace/sendQName[/[rcvQspace:] rvcQname]/[failureQname]</code></p> <p>In the URI, <code>resourcename</code> corresponds to a WTC Import name and <code>remotename</code> corresponds to the service name exported by the remote Tuxedo domain. The URI <code>resourcename</code> is required, and the <code>remotename</code> is optional.</p> <p>If more than one URI is specified, you must have unique resource names for the endpoints. If no remote name is specified, its value is the value of the resource name. If no remote name is entered or if remote and resource name are the same, only one URI is allowed. In this case resource name and remote name have the same value. This allows already defined WTC Imports to make use of WTC load-balancing and failover.</p> <li data-bbox="532 1444 1369 1856"> <p>■ ws - <code>http://host:port/someService</code></p> <p>Note: Oracle Service Bus no longer supports duplicate endpoint URIs within the same business service.</p> <p>Click Add to add one or more additional URIs. At run time, the URLs are selected based on the load balancing algorithm you selected in the Load Balancing Algorithm field.</p> <p>If you selected Random-weighted in the Load Balancing Algorithm field, you can also enter a weight in the Endpoint URI field. The default is 1.</p> <p>If you have multiple endpoint defined, and you selected None in the Load Balancing Algorithm field, the order of endpoints is significant. You can reorder the endpoints using the Up and Down buttons.</p> <p>Oracle Service Bus does not support duplicate endpoint URIs within the same business service.</p>

Table 4–6 (Cont.) Business Service Transport Configuration Options

Option	Description
Retry count	<p>In case of delivery failure when sending outbound requests, specify the number of times to retry individual URL endpoints; in other words, the number of failover attempts.</p> <p>For example, a business service has one configured URI (U1) and the number of retries is set to 3. If U1 fails on the first attempt, the system retries the U1 endpoint 3 more times.</p> <p>If a business service has 2 configured URIs (U1 and U2) and a retry count of 3, if the first attempt (for example, to U1) fails, the system tries (fails over to) the next URI (U2). If that also fails, the system makes two more attempts, once to U1 and once to U2.</p>
Retry Iteration Interval	<p>Specify the number of seconds the system pauses before iterating over all the endpoint URIs in the list again.</p> <p>For example, a business service has two configured URIs (U1 and U2) and a retry count of 2 with a retry iteration interval of 5 seconds. If the first attempt (to U1) fails, the system tries U2 right away. But if U2 also fails, the system waits for 5 seconds and retries U1 once more.</p>
Retry Application Errors	<p>Select Yes or No.</p> <p>In case of delivery failure when sending outbound requests, specify whether or not to retry endpoint URIs based on application errors (for example, a SOAP fault).</p>

4.2.5 Business Service Message Handling Configuration Page

Use the Business Service Message Handling Configuration page to specify how Oracle Service Bus is to encode outbound messages sent by business services and whether Oracle Service Bus should stream attachments in outbound response messages instead of buffering the attachment contents in memory.

Using this page, you can enable the business service to encode outbound messages in MTOM/XOP format. SOAP Message Transmission Optimization Mechanism (MTOM) is a method of sending binary data to and from Web services. MTOM uses XML-binary Optimized Packaging (XOP) to transfer the binary data.

Using this page, you can also enable the business service to store attachments in outbound response messages to a disk file and then process the data in a streaming fashion without buffering the attachment contents in memory. This enables the business service to process large attachments robustly and efficiently.

Table 4–7 Business Service Message Handling Options

Option	Description
XOP/MTOM Support	<p>Oracle Service Bus supports XOP/MTOM using the following transports:</p> <ul style="list-style-type: none"> ■ HTTP/S ■ Local ■ SB <p>Select the Enabled check box to enable the business service to encode outbound messages in MTOM/XOP format. Note that this option is disabled for imported business services that are based on previous release configurations.</p> <p>If XOP/MTOM Support is enabled, select how to handle binary data in the \$header and \$body message context variables from among the following options:</p> <ul style="list-style-type: none"> ■ Include Binary Data by Reference: (Default) In an outbound response message, replace xop:Include elements with ctx:binary-content elements when setting up the \$body message context variable. ■ Include Binary Data by Value: In an outbound response message, replace xop:Include elements with base64-encoded text versions of corresponding binary data when setting up the \$body message context variable. <p>Note that if XOP/MTOM Support is enabled for a business service, it is not required that every outbound message be in the MTOM format. Instead, this setting specifies that the business service is capable of handling an MTOM payload.</p> <p>Since Oracle Service Bus does not support a combination of MTOM and SwA, the system issues a runtime error when Oracle Service Bus attempts to dispatch an outbound request to a business service and the business service is both MTOM/XOP-enabled and the \$attachments message context variable is not null.</p>
Attachments	<p>Oracle Service Bus supports streaming MIME attachments using the HTTP/S transport.</p> <p>Select the Page Attachments to Disk check box to enable the business service to stream attachments in outbound response messages.</p> <p>Note that if you enable XOP/MTOM Support, the Attachments option is only available if you choose the Include Binary Data by Reference option under XOP/MTOM Support. Note also that payloads that contain attachments must conform to RFC 822. Specifically, lines containing Internet headers need to be terminated with CRLF (carriage return line feed).</p>

Table 4–7 (Cont.) Business Service Message Handling Options

Option	Description
Result Caching	<p>If you invoke business services whose results seldom change, result caching improves business service performance by returning cached results to the client instead of invoking an external service directly.</p> <p>To use result caching for business services, you must first enable it globally, as described in "Enabling Result Caching Globally" in the <i>Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBAG1294.</p> <p>Configure result caching in the Advanced Settings section using the following guidelines:</p> <ul style="list-style-type: none"> ■ Select the Supported option. When you do, the remaining fields are enabled. ■ Cache Token Expression – Oracle Service Bus uses a cache key to identify cached results for retrieval or population, and the cache token portion of the cache key provides the unique identifier. You can use an expression—the Cache Token Expression—to generate the cache token part of the cache key to uniquely identify a cached result for the business service. Click Expression to build the expression. <p>To generate the cache token from a value in the request, use an expression that gets the value from the message \$body, \$header, \$operation, or \$transportMetaData (\$outbound/ctx:transport/ctx:request or \$outbound/ctx:transport/ctx:response). For example, you can populate the cache-token from a customer ID in the \$body.</p> <p>In the Caching Expression builder, you can also select the namespace for the expression in the Namespace Definitions tab.</p> <p>The Cache Token Expression must resolve to a String or the value of simple content, such as an attribute or an element with no child elements. If the expression evaluates to null or causes an error, results are not cached.</p> <p>Note: You can also generate the cache token from the request (in the proxy service message flow that invokes the business service), without setting a Cache Token Expression in the business service configuration. To do this, include a value in \$outbound/ctx:transport/ctx:request/ctx:cache-token in the proxy service pipeline. Any value in that cache-token overrides the Cache Token Expression in the business service configuration.</p>

Table 4–7 (Cont.) Business Service Message Handling Options

Option	Description
Result Caching continued	<ul style="list-style-type: none"> <li data-bbox="651 260 1360 394"> <p>Expiration Time – Expiration Time, or time-to-live (TTL), determines when an entry in the business service’s result cache is flushed. A duration of zero (0) means no expiration. A negative duration means do not cache. Select one of the following:</p> <p>Use Default – The default is the expiry-delay value in <i>DOMAIN_HOME/config/osb/coherence/osb-coherence-config.xml</i>. The default is 5 minutes.</p> <p>Duration – Set a specific length of time.</p> <p>XQuery Expression – You can enter an XQuery expression that gets an Expiration Time value from the request or response. Select Request or Response.</p> <p>To generate the expiration time from a value in the request or response, use an expression that gets the value from the \$body, \$header, \$operation, or \$transportMetaData (\$outbound/ctx:transport/ctx:request or \$outbound/ctx:transport/ctx:response). For example, use a value you have set in the Cache-Control HTTP header.</p> <p>In the Caching Expression builder, you can also select the namespace for the expression using the Namespace Definitions tab.</p> <p>Expiration Time must resolve to an Integer (representing seconds), an XQuery dayTimeDuration (XSD type), or the integer value of simple content representing seconds, such as an attribute or an element with no child elements. If the expression evaluates to null or causes an error, results are not cached.</p> <p>Note: You can also generate the expiration from the request, without setting an Expiration Time in the business service configuration. To do this, include a value in \$outbound/ctx:transport/ctx:request/ctx:cache-ttl in the proxy service pipeline. Any value in the cache-ttl element overrides the Expiration Time in the business service configuration.</p> <p>For a more detailed description of result caching, see “Improving Performance by Caching Business Service Results” in the <i>Oracle Fusion Middleware Administrator’s Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBAG170.</p>

4.2.6 Business Service - Service Policy Configuration Page

Use the Business Service - Service Policy Configuration page to configure service policy settings for a business service.

Table 4–8 Business Service Service Policy Configuration Options

Option	Description
OWSM Policy Bindings	<p>To add Oracle Web Services Manager policies to services in the IDE, you must be connected to a running server that has Oracle Web Services Manager enabled.</p> <p>Note: When working with multiple servers in Eclipse, Eclipse chooses the first valid Oracle Service Bus server in the list of servers for retrieval of Oracle Web Services Manager policies.</p> <p>Use category filtering to help you find the policy you want. For business services, only client-side policies are displayed.</p> <p>After adding policies, you can provide overrides on the Security page.</p> <p>For information, see "Securing Oracle Service Bus with Oracle Web Services Manager" in the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBDV1681.</p>
WSDL-Based Policy	Select this option if the service policy is associated with the WSDL upon which the service is based.
Custom Policy Bindings	Select this option to add service-level policies, operation-level policies (in which case the policy applies to both the request and response messages), request policies, and response policies directly.

4.2.7 Business Service Security Configuration Page

Use the Security configuration page to provide overrides to security policies used in the business service.

Table 4–9 Business Service Security Configuration Options

Option	Description
Service Account	<p>Click Browse and select a service account from the list displayed. If no service account is specified, an anonymous subject is used.</p> <p>Note: This option is not available for services that use Oracle Web Services Manager policies, because user credentials are provided by override keys.</p> <p>For more information, see Section 4.19, "Service Accounts".</p>
Policy Overrides	<p>For OWSM policy bindings, provide any desired overrides that are allowed.</p> <p>For more information, see Table 4–8 and "Securing Oracle Service Bus with Oracle Web Services Manager" in the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBDV1681.</p>

4.3 Proxy Service Configuration

You configure new proxy services in the proxy service editor after you create a new proxy service. You can view and modify those settings in the Proxy Service editor: The proxy service configuration pages are:

- [Section 4.3.1, "Proxy Service General Configuration Page"](#)
- [Section 4.3.2, "Proxy Service Message Type Configuration Page"](#)

- [Section 4.3.3, "Proxy Service Operation Selection Configuration Page"](#)
- [Section 4.3.4, "Proxy Service Message Handling Configuration Page"](#)
- [Section 4.3.5, "Proxy Service SOAP Binding Configuration Page"](#)
- [Section 4.3.6, "Proxy Service Transport Configuration Page"](#)
- [Section 4.3.7, "Proxy Service - Service Policy Configuration Page"](#)
- [Section 4.3.8, "Proxy Service Security Configuration Page"](#)

4.3.1 Proxy Service General Configuration Page

Use the Proxy Service General Configuration page to set or modify general configuration properties for a proxy service.

4.3.1.1 Proxy Service Editor Options

The following table describes the options in the editor:

Table 4–10 Proxy Service Editor Options

Option	Description
Description	Enter a description for this proxy service.

Table 4–10 (Cont.) Proxy Service Editor Options

Option	Description
Service Type (editor only)	<p>This option shows the service type of the proxy service. You can change only some of the properties of some of the service types:</p> <ul style="list-style-type: none"> <p>■ WSDL Web Service - Select this option to create a proxy service based on a WSDL. Then enter the WSDL name, qualified by its path (for example, <code>myProject/myFolder/myWSDL</code>). Alternatively, click Browse to select a WSDL resource. You can enter (or click Browse to select) a different port or binding from the same WSDL. You can also specify a different WSDL, but by doing so, you are effectively creating a new service and you will have to configure it as if it were a new service.</p> <p>(port or binding) - Enter the name of a port (defined in the WSDL) to describe an actual transport address, or enter the name of a binding (defined in the WSDL) to map to a transport address. If you use Browse to select a WSDL, the Select a WSDL Definition dialog lists any defined ports and bindings.</p> <p>Note: If you are going to use the SOAP Body Type for operations, ensure that the WSDL does not have two operations with the same input message. The SOAP Body Type operation cannot be uniquely identified by inspecting the input message.</p> <p>■ Transport Typed Service - Select this option to create a JEB proxy service.</p> <p>■ Messaging Service - Select this option to create a service that can receive messages of one data type and respond with messages of a different data type. These exchanges can be either request/response or one-way.</p> <p>(HTTP GET is supported only in the Any XML Service and Messaging Service service types.)</p> <p>■ Any SOAP Service - Select this option to create a SOAP service that does not have an explicitly defined, concrete interface. You can change the SOAP version (SOAP 1.1 or SOAP 1.2)</p> <p>■ Any XML Service - Select this option to create an XML service that does not have an explicitly defined, concrete interface.</p> <p>(HTTP GET is supported only in the Any XML Service and Messaging Service service types.)</p>

4.3.2 Proxy Service Message Type Configuration Page

Use Proxy Service Message Type Configuration page to configure message types for a proxy service whose type is Messaging Service.

The binding definition for messaging services consists of configuring the content-types of the messages that are exchanged. The content-type for the response does not have to be the same as for the request; therefore, the response is configured separately (for example, the service could accept an MFL message and return an XML acknowledgment receipt).

Note: E-mail, File, FTP, or SFTP transport proxy services whose type is Messaging Service support one-way messaging *only*; the **Response Message Type** should be **None**. If you select an option other than **None**, the E-mail, File, FTP, or SFTP protocols will not be available on the Transport Configuration page.

Table 4–11 Proxy Service Message Type Options

Option	Description
Request Message Type	<p>Select a message type for the request message:</p> <ul style="list-style-type: none"> ■ None - Select this option if there is no request message. ■ Binary - Select this option if the content-type of the message is unknown or not important. ■ Text - Select this option if the message can be restricted to text. ■ MFL - Select this option if the message is a binary document conforming to an MFL definition. Enter the MFL file name (qualified by its path), or click Browse to select a file. You can configure only one MFL file. Note: To support multiple MFL files, define the content as binary or text and use the MFL action in the message flow to convert to XML. ■ XML - Select this option if the message is an XML document. Enter the XML file name (qualified by its path), or click Browse to select a file. Optionally provide some type information by declaring (in the element or type field) the XML schema type of the XML document exchanged.
Response Message Type	<p>Select a message type for the response message:</p> <ul style="list-style-type: none"> ■ None - Select this option if there is no response message. ■ Binary - Select this option if the content-type of the message is unknown or not important. ■ Text - Select this option if the message can be restricted to text. ■ MFL - Select this option if the message is a binary document conforming to an MFL definition. Enter the MFL file name (qualified by its path), or click Browse to select a file. You can configure only one MFL file. Note: To support multiple MFL files, define the content as binary or text and use the MFL action in the message flow to convert to XML. ■ XML - Select this option if the message is an XML document. Enter the XML file name (qualified by its path), or click Browse to select a file. Optionally provide some type information by declaring (in the element or type field) the XML schema type of the XML document exchanged.

4.3.3 Proxy Service Operation Selection Configuration Page

Use Proxy Service Operation Selection Configuration page to enforce WS-I compliance (for SOAP 1.1 services only) and select the selection algorithm to use to determine the operation called by this proxy service. This option is only available for SOAP or XML services defined from a WSDL.

The WSDL specification defines a default algorithm to compute which operation is called based on the type of the SOAP message received. However, there are cases (for example, performance issues, signature/encryption issues, or the default algorithm is not applicable) when you may need to select the operation based on other means.

Oracle Service Bus provides additional algorithms. Each of them follows the same pattern and are based on the evaluation of an expression to get a value that is then used to lookup the corresponding operation in a static table.

Oracle Service Bus is generally very forgiving if an inbound message is either missing data such that the operation cannot be determined, or has data that does not correspond to a valid operation. Both of these conditions result in `$operation` being empty. Rather than reject all such messages, Oracle Service Bus does not initialize the operation variable in the context but otherwise continues to process the message.

However, security requirements are enforced if the proxy service is WSDL-based and at least one of the following conditions is true:

- The WSDL has a WS-Security policy and the proxy is an active intermediary.
- The proxy has message-level custom authentication (either custom token or username/password).

If these conditions are met, then there is a runtime check to make sure the operation selection algorithm returns a valid operation name. If the operation selection returns null or an operation that is not in the WSDL, then the message is rejected and an error is raised.

Table 4–12 Proxy Service Operation Selection Options

Option	Description
Enforce WS-I Compliance	<p>Select or deselect this check box to specify whether or not the service is to conform to the Basic Profile defined by the Web Services Interoperability Organization. This option is available for or SOAP 1.1 services only</p> <p>When a service is marked WS-I compliant, checks are performed against the messages sent to and from that service.</p> <p>For proxies, checks are performed against request messages received by the proxy. For invoked services (i.e. services invoked by a proxy via service callout action or route node), checks are performed against the response messages received from those services. Note that it is the WS-I compliance property of the invoked service and not the proxy that determines whether or not checks are performed against messages received from the invoked service. If you specify WS-I compliance testing for an invoked service, the message flow generates a fault for response errors.</p>

Table 4–12 (Cont.) Proxy Service Operation Selection Options

Option	Description
Selection Algorithm	<p>Select one of the following and perform any required additional steps:</p> <ul style="list-style-type: none"> ■ Transport Header - Select this algorithm to define the transport header that contains the lookup value. Then: <ul style="list-style-type: none"> ■ In the Header Name field, enter the transport header that extracts the value used as a key to select the operation being invoked. ■ Under the Operation Mapping field, specify the value for each operation in the Value field. The value is used as the key of the operation. ■ SOAPAction Header - Select this algorithm to specify that operation mapping be done automatically from the WSDL associated with this proxy service. ■ WS-Addressing - Select this algorithm to specify that the lookup value is contained by the WS-Addressing <code>Action</code> tag located in the SOAP headers of the SOAP message. Then, under the Operation Mapping field, specify the value for each operation in the Value field. The value is used as the key of the operation. ■ SOAP Header - Select this algorithm to define an XPath expression to be evaluated against the SOAP headers. This allows you to get the lookup value. Then: <ul style="list-style-type: none"> ■ In the XPath Expression field, specify the XPath expression that extracts the value used as a key to select the operation being invoked. ■ In the Operation Mapping field, specify the value for each operation in the Value field. The value is used as the key of the operation. ■ SOAP Body Type - This is the default algorithm defined by the WSDL specification to compute which operation is called based on the type of the SOAP message received. <p>If the proxy service is configured for a Web Service security pass-through scenario with an encrypted body, you cannot select this algorithm. A similar caveat applies to pass-through encrypted SOAP headers.</p> <p>If you have a WSDL that has two operations with the same input message, do not select this algorithm for operations, because the operation cannot be uniquely identified by inspecting the input message.</p> ■ Payload Type - Available only for XML services based on a WSDL port or WSDL binding.
Header Name	<p>This option is available only when the Selection Algorithm option is set to Transport Header.</p> <p>Enter the transport header that extracts the value used as a key to select the operation being invoked.</p>
XPath Expression	<p>This option is available only when the Selection Algorithm option is set to SOAPHeader.</p> <p>Specify the XPath expression that extracts the value used as a key to select the operation being invoked.</p>
Operation Mapping	<p>This option is available only when the Selection Algorithm option is set to Transport Header, WS-Addressing, or SOAP Body Type.</p> <p>Specify the value for each operation in the Value field. The value is used as the key of the operation.</p>

4.3.4 Proxy Service Message Handling Configuration Page

Use the Proxy Service Message Handling Configuration page to specify different aspects of message handling, as described in [Table 4-13](#).

Table 4-13 Proxy Service Message Handling Options

Option	Description
Transaction Required	<p>Select this option to ensure Oracle Service Bus executes the proxy service message flow in the context of a transaction. If a global transaction already exists, the transport provider propagates it in the request (even if you do not select this option). If no global transaction exists, the message flow run time starts a transaction. If the message flow run time starts a transaction, the transaction context begins before the service configuration is validated or run (for example, security checking or WS-I validation), and before message flow execution, ensuring that all processing and execution occurs in the transaction context.</p> <p>If the message flow run time starts a transaction, quality of service is exactly-once. However, if Service Callouts or Publish actions have the outbound quality of service parameter set to best-effort (the default), Oracle Service Bus executes those actions outside of the transaction context. To have Oracle Service Bus execute those actions in the same request transaction context, set quality of service on those actions to exactly-once.</p> <p>The service maintains its messaging pattern (synchronous, asynchronous, one-way) regardless of the setting on this option.</p> <p>For transaction timeouts, the global transaction timeout value configured in the Oracle WebLogic Server Console applies.</p> <p>Exceptions in Transactions</p> <p>Oracle Service Bus invokes the system error handler for failed transactions. You cannot catch failed transaction exceptions in a user-configured error handler. For synchronous patterns, a transaction exception is returned through the response. For asynchronous patterns, where the transaction is designed to be committed in the request, the exception is sent back on the request thread.</p> <p>Note that in asynchronous patterns, an error in the response that occurs after transaction committal in the request does not affect the transaction.</p>
Same Transaction for Response	<p>If you select this option, Oracle Service Bus propagates the transaction context from the request thread to the response thread.</p> <p>If you select this option, the message pattern becomes synchronous automatically, regardless of the initial message pattern setting (such as asynchronous or one-way).</p> <p>You would not use this option, for example, if the business service in the request required a transaction committal before sending the response, such as in a one-way pattern.</p> <p>For transaction timeouts, the global transaction timeout value configured in the Oracle WebLogic Server Console applies.</p>

Table 4–13 (Cont.) Proxy Service Message Handling Options

Option	Description
Content Streaming	<p>Select this option to stream message content rather than store it in memory.</p> <p>Select the Enabled check box and choose the following:</p> <ul style="list-style-type: none">■ Whether to buffer the intermediate content in memory (Memory Buffer) or to a disk file (Disk Buffer)■ Whether to enable Compression <p>For more information, see "Streaming body Content" in the <i>Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBAG314</p>

Table 4–13 (Cont.) Proxy Service Message Handling Options

Option	Description
XOP/MTOM Support	<p data-bbox="716 260 1442 363">SOAP Message Transmission Optimization Mechanism (MTOM) is a method of sending binary data to and from Web services. MTOM uses XML-binary Optimized Packaging (XOP) to transfer the binary data.</p> <p data-bbox="716 380 1442 457">Use this option to decode and parse inbound messages in MTOM/XOP format and to send responses using the MTOM/XOP format, when appropriate.</p> <p data-bbox="716 474 1442 527">Oracle Service Bus supports XOP/MTOM using the following transports:</p> <ul data-bbox="716 537 854 642" style="list-style-type: none"> <li data-bbox="716 537 854 562">■ HTTP/S <li data-bbox="716 579 821 604">■ Local <li data-bbox="716 621 789 646">■ SB <p data-bbox="716 657 1442 789">Select the Enabled check box to enable the proxy service to decode and parse inbound messages in MTOM/XOP format and to send responses using the MTOM/XOP format, when appropriate. Note that this option is disabled for imported proxy services that are based on previous release configurations.</p> <p data-bbox="716 806 1442 884">If XOP/MTOM Support is enabled, select how to handle binary data in the \$header and \$body message context variables from among the following options:</p> <ul data-bbox="716 894 1442 1125" style="list-style-type: none"> <li data-bbox="716 894 1442 999">■ Include Binary Data by Reference: (Default) In an inbound request message, replace xop:Include elements with ctx:binary-content elements when setting up the \$header and \$body message context variables. <li data-bbox="716 1016 1442 1125">■ Include Binary Data by Value: In an inbound request message, replace xop:Include elements with base64-encoded text versions of corresponding binary data when setting up the \$header and \$body message context variables. <p data-bbox="716 1136 1442 1213">Use Include Binary Data by Reference when you need direct access to binary data, for example to pass data to a Java callout or Message Format Language (MFL) transform.</p> <p data-bbox="716 1230 1442 1255">Use Include Binary Data by Value in the following cases:</p> <ul data-bbox="716 1266 1442 1497" style="list-style-type: none"> <li data-bbox="716 1266 1442 1398">■ To bridge between MTOM and non-MTOM services. For example, consider an MTOM-enabled proxy service that receives a request that is then routed to a non-MTOM-enabled service. You could use this option to comply with existing standards for sending binary data in XML in base64-encoded form. <li data-bbox="716 1415 1442 1497">■ To validate the contents of the message against an XML schema that requires a base64binary element to be used in place of binary data <p data-bbox="716 1507 1442 1612">Note that if XOP/MTOM Support is enabled for a proxy service, it is not required that every inbound message be in the MTOM format. Instead, this setting specifies that when an MTOM-formatted message arrives, the proxy service should handle it accordingly.</p> <p data-bbox="716 1629 1442 1707">Note also that when proxy services not enabled for XOP/MTOM Support receive an MTOM-formatted message, the service rejects the message and issues a runtime error.</p>

Table 4–13 (Cont.) Proxy Service Message Handling Options

Option	Description
Attachments	<p>Use this option to have the proxy service store MIME attachment content to a disk file and then process the data in a streaming fashion without buffering the attachment contents in memory. This enables the proxy service to process large attachments robustly and efficiently.</p> <p>Oracle Service Bus supports streaming MIME attachments using the following transports:</p> <ul style="list-style-type: none"> ■ HTTP/S ■ Local (when chained through an HTTP proxy with streaming attachments enabled) <p>Select the Page Attachments to Disk check box to enable the proxy service to stream MIME attachments. When enabled for HTTP proxy services, the option applies to proxy service inbound request messages.</p> <p>Note that if you select XOP/MTOM Support, the Attachments option is only available if you choose the Include Binary Data by Reference option under XOP/MTOM Support. Note also that payloads that contain attachments must conform to RFC 822. Specifically, lines containing Internet headers need to be terminated with CRLF (carriage return line feed).</p>

4.3.5 Proxy Service SOAP Binding Configuration Page

This page is displayed only if the service you are creating has operations.

Use the Proxy Service SOAP Binding Configuration page to enforce WS-I compliance (for SOAP 1.1 services only) and select the selection algorithm to use to determine the operation called by this proxy service. This option is only available for SOAP or XML services defined from a WSDL.

The WSDL specification defines a default algorithm to compute which operation is called based on the type of the SOAP message received. However, there are cases (for example, performance issues, signature/encryption issues, or the default algorithm is not applicable) when you may need to select the operation based on other means.

Oracle Service Bus provides additional algorithms. Each of them follows the same pattern and are based on the evaluation of an expression to get a value that is then used to lookup the corresponding operation in a static table.

Oracle Service Bus is generally very forgiving if an inbound message is either missing data such that the operation cannot be determined, or has data that does not correspond to a valid operation. Both of these conditions result in `$operation` being empty. Rather than reject all such messages, Oracle Service Bus does not initialize the operation variable in the context but otherwise continues to process the message.

However, security requirements are enforced if the proxy service is WSDL-based and at least one of the following conditions is true:

- The WSDL has a WS-Security policy and the proxy is an active intermediary.
- The proxy has message-level custom authentication (either custom token or user name/password).

If these conditions are met, then there is a runtime check to make sure the operation selection algorithm returns a valid operation name. If the operation selection returns null or an operation that is not in the WSDL, then the message is rejected and an error is raised.

Table 4–14 Proxy Service SOAP Binding Options

Option	Description
Enforce WS-I Compliance	<p>For SOAP 1.1 services only:</p> <p>Select or deselect this check box if you want to specify whether or not the service is to conform to the Basic Profile defined by the Web Services Interoperability Organization.</p> <p>When a service is marked WS-I compliant, checks are performed against the messages sent to and from that service. For proxies, checks are performed against request messages received by the proxy. For invoked services (i.e. services invoked by a proxy via service callout action or route node), checks are performed against the response messages received from those services. Note that it is the WS-I compliance property of the invoked service and not the proxy that determines whether or not checks are performed against messages received from the invoked service. If you specify WS-I compliance testing for an invoked service, the message flow generates a fault for response errors.</p>

4.3.6 Proxy Service Transport Configuration Page

Use the Proxy Service Transport Configuration page to select a transport protocol for the proxy service and to set other general transport configuration settings.

Note: Inbound transport-level security applies to the client applications and Oracle Service Bus proxy services. Outbound transport-level security applies to the connections between Oracle Service Bus proxy services and business services.

Table 4–15 Proxy Service Transport Configuration Options

Option	Description
Protocol	<p>Select a transport protocol from the list. The protocols available differ, depending on the service type you are creating:</p> <ul style="list-style-type: none"> ▪ WSDL Web Service: http, jca, jms, local, sb, ws ▪ Transport Typed Service: jejb ▪ Messaging Service: email, file, ftp, http, jms, local, mq (if available), sftp, tuxedo ▪ Any SOAP Service: http, jms, local, sb ▪ Any XML Service: email, file, ftp, http, jms, local, mq (if available), sb, sftp, tuxedo

Table 4–15 (Cont.) Proxy Service Transport Configuration Options

Option	Description
Endpoint URI	<p data-bbox="548 260 1317 315">Enter an endpoint URI in the format based on the transport protocol you selected in the Protocol field, above: The formats are:</p> <ul style="list-style-type: none"> <li data-bbox="548 327 1105 354">■ email - mailfrom:mail-server-host:port <li data-bbox="548 367 951 394">■ file - file:///root-dir/dir1 <li data-bbox="548 407 1062 434">■ ftp - ftp://hostname:port/directory <li data-bbox="548 447 789 474">■ http - /someName <p data-bbox="594 487 1268 514">The HTTP transport supports both HTTP and HTTPS endpoints.</p> <ul style="list-style-type: none"> <li data-bbox="548 527 1062 554">■ jca - jca://<resource_adapter_jndi> <li data-bbox="548 567 1349 674">■ jejb - The URL format is jndi_name. The URI configured for a JEJB proxy service becomes the global JNDI name for locating the stateless session bean generated by the JEJB transport from the remote/business interface in the client JAR. <p data-bbox="594 686 1349 793">Note: For EJB 3.0, jndi_name is the mappedName attribute of the @javax.ejb.Stateless annotation in the generated bean. The lookup JNDI name for the generated EJB service is suffixed with #interface_class, which is the fully qualified name of the business interface.</p> <p data-bbox="594 806 1024 833">You can access the JEJB proxy service as:</p> <p data-bbox="594 846 1117 873">EJB 2.1 - protocol://host:port/jndi_name</p> <p data-bbox="594 886 1349 913">EJB 3.0 - protocol://host:port/jndi_name#interface_class</p> <ul style="list-style-type: none"> <li data-bbox="548 926 1227 980">■ jms - jms://host:port[,host:port]*/connection_factory/jndi_destination <p data-bbox="594 993 1365 1047">To target a target a JMS destination to multiple servers, use the following URI format:</p> <p data-bbox="594 1060 1357 1115">jms://host1:port,host2:port/QueueConnectionFactory/DestinationName</p> <p data-bbox="594 1127 1365 1234">Note that when you create a proxy service, you can configure a JMS endpoint URI even if the server at that endpoint is not available. However, in the case of JMS, when you activate the session, the endpoint must be available.</p> <p data-bbox="594 1247 1276 1302">You can also omit host and port from the URI to have the lookup performed on the local machine. For example:</p> <p data-bbox="594 1314 1198 1341">jms:///connection_factory/jndi_destination</p> <p data-bbox="594 1354 1357 1430">In a cluster: The host names in the JMS URI must exactly match the host names of the cluster servers as they are configured in Oracle WebLogic Server.</p> <ul style="list-style-type: none"> <li data-bbox="548 1442 651 1470">■ local <p data-bbox="594 1482 1110 1509">This transport does not require an endpoint URI.</p>

Table 4–15 (Cont.) Proxy Service Transport Configuration Options

Option	Description
Endpoint URI (continued)	<ul style="list-style-type: none"> <li data-bbox="626 260 1448 653"> <p>■ mq - mq://local-queue-name?conn=mq-connection-resource-ref</p> <p>local-queue-name is the name of the MQ queue from which the proxy service reads messages.</p> <p>mq-connection-resource-ref is the path (project/folder) and name of the MQ connection resource; for example, default/my_MQconnection.</p> <p>Note: The Endpoint URI cannot contain spaces, so do not create MQ Connection resources or projects/folders with spaces in the names.</p> <p>To make the MQ transport available in Oracle Service Bus, see "MQ Transport" in the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBDV1117.</p> <li data-bbox="626 674 1448 884"> <p>■ sb - service_name</p> <p>service_name is the unique identifier for the proxy service. By default, this name will be the proxy service name.</p> <p>service_name must only contain characters permitted in URIs (as described in RFC2396 at http://www.ietf.org/rfc/rfc2396.txt), except it cannot contain forward slash (/) or colon (:) characters.</p> <li data-bbox="626 905 1166 926"> <p>■ sftp - sftp://hostname:port/directory</p> <li data-bbox="626 947 1448 1293"> <p>■ tuxedo - servicename</p> <p>The URI servicename corresponds to a WTC Export that the remote Tuxedo domain identifies as a Tuxedo service.</p> <p>If more than one URI is specified, you must have unique resource names for the endpoints. If no remote name is specified, its value is the value of the resource name. If no remote name is entered or if remote and resource name are the same, only one URI is allowed. In this case resource name and remote name will have the same value. This allows users using already defined WTC Imports to make use of WTC load-balancing and failover.</p> <p>Note: If you configure two identical URIs, an error indicates that the service name already exists.</p> <li data-bbox="626 1314 1448 1404"> <p>■ ws - /contextPath</p> <p>contextPath must be unique for proxy services that use either HTTP or WS transport.</p>

Table 4–15 (Cont.) Proxy Service Transport Configuration Options

Option	Description
Get All Headers	<p>Select Yes to retrieve all the headers from the transport.</p> <p>Select No to retrieve a defined set of headers. If you select No, enter a set of headers in the Header field, then click Add. (This step does not apply to Local transport.)</p> <p>Note: Oracle Service Bus does not pass the HTTP Authorization header from the request to the pipeline because it opens a security vulnerability. You could inadvertently create a log action that writes the user name and unencrypted password to a log file. If your design pattern requires the HTTP Authorization header to be in the pipeline, do the following:</p> <ol style="list-style-type: none"> 1. In the startup command for Oracle Service Bus, set the following system property to true: <code>com.bea.wli.sb.transports.http.GetHttpAuthorizationHeaderAllowed</code> 2. In Eclipse, on the Transport Configuration page, select Yes for Get All Headers or select No and specify Authorization. 3. Restart Oracle Service Bus. <p>Oracle Service Bus will pass the Authorization header to the pipeline.</p>

4.3.7 Proxy Service - Service Policy Configuration Page

Use Proxy Service - Service Policy Configuration page to configure service policies for a proxy service.

Table 4–16 Proxy Service Service Policy Options

Option	Description
OWSM Policy Bindings	<p>To add Oracle Web Services Manager policies to services in the IDE, you must be connected to a running server that has Oracle Web Services Manager enabled.</p> <p>Note: When working with multiple servers in Eclipse, Eclipse chooses the first valid Oracle Service Bus server in the list of servers for retrieval of Oracle Web Services Manager policies.</p> <p>Use category filtering to help you find the policy you want. For proxy services, only service policies are displayed.</p> <p>After adding policies, you can provide overrides on the Security page.</p> <p>For information, see "Securing Oracle Service Bus with Oracle Web Services Manager" in the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBDV1681.</p>
WSDL-Based Policy	Select this option if the service policy is associated with the WSDL upon which the service is based.
Custom Policy Bindings	Select this option to add service-level policies, operation-level policies (in which case the policy applies to both the request and response messages), request policies, and response policies directly.

4.3.8 Proxy Service Security Configuration Page

Use the Proxy Service Message Level Security Configuration page to configure security for proxy services.

For WLS 9.2 policies, message-level custom tokens and message-level user name and password are supported on proxy services of the following binding types:

- WSDL-SOAP
- WSDL-XML
- Abstract SOAP
- Abstract XML
- Mixed - XML (in the request)
- Mixed - MFL (in the request)

The configuration for both custom user name/password and custom token is similar. In both cases, you specify XPath expressions that enable Oracle Service Bus to locate the necessary information. The root of these XPath expressions is as follows:

- Use `soap-env:Envelope/soap-env:Header` if the service binding is AnySOAP or WSDL-SOAP.
- Use `soap-env:Body` if the service binding is not SOAP based.

All XPath expressions must be in a valid XPath 2.0 format. The XPath expressions must use the XPath "declare namespace" syntax to declare any namespaces used, as follows:

```
declare namespace ns='http://webservices.mycompany.com/MyExampleService';)
```

Table 4–17 Proxy Service Message-Level Security Options

Option	Description
Service Key Provider	<p>The name of a service key provider to be used by the service.</p> <p>You can enter the path (project/folder) and name of a service key provider, or click Browse to select one.</p> <p>A service key provider is only required in certain cases:</p> <ul style="list-style-type: none"> ■ Outbound two-way TLS/SSL, where the proxy service routes messages to HTTPS services that require client-certificate authentication. ■ In some Web Service security scenarios, for example, if the proxy service requires messages to be encrypted. <p>To add a Web service security-enabled proxy service, you must create the proxy service from a WSDL (port or binding) with WS-Policy attachments.</p>
Policy Overrides	<p>For OWSM policy bindings, provide any desired overrides that are allowed.</p> <p>For more information, see Table 4–16 and "Securing Oracle Service Bus with Oracle Web Services Manager" in the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBDV1681.</p>
Custom Authentication Settings	<p>Select one of the following:</p> <ul style="list-style-type: none"> ■ None - if the service will not use custom authentication. ■ Custom User Name and Password - if the service will use a custom name and password, specified as XPath expressions ■ Custom Token - if the service will use a custom token
Custom User Name and Password - User Name XPath	<p>The user name, specified as an XPath expression.</p> <p>The XPath expression is evaluated against the message headers or payload, as appropriate, which allows Oracle Service Bus to obtain the user name and for custom authentication.</p>

Table 4–17 (Cont.) Proxy Service Message-Level Security Options

Option	Description
Custom User Name and Password - User Password XPath	<p>The password, specified as an XPath expression.</p> <p>The XPath expression is evaluated against the message headers or payload, as appropriate, which allows Oracle Service Bus to obtain the password values for custom authentication.</p>
Custom Token - Token Type	<p>Enter the type for the custom token type. Only the active token types configured for an Oracle WebLogic Server Identity Assertion provider can be used.</p>
Custom Token - Token XPath	<p>An XPath expression that specifies a path to the custom token. Oracle Service Bus evaluates the Token XPath expression against the message headers or payload, as appropriate, to obtain the token for custom authentication.</p> <p>To create or edit an expression, click <XPath> (or the expression fragment, if one is already defined) to display the XPath Expression Editor.</p>
Custom User Name and Password - Context Properties or Custom Token - Context Properties	<p>Optionally, specify one or more context properties to pass additional context information to the Authentication (Custom User Name and Password) or Identity Assertion (Custom Token) security provider.</p> <p>Context Properties provide a way (the <code>ContextHandler</code> interface) to pass additional information to the WebLogic Security Framework so that a security provider can obtain contextual information.</p> <p>Enter the Property Name as a literal string, and the Value Selector as a valid XPath expression. (XPath expressions can also be literal strings.)</p> <p>The XPath expressions are evaluated against the same message-part that is used for the custom token or custom user name/password. That is, the Value Selector XPath expressions for SOAP-based proxy services evaluate against the header and against the payload for non-SOAP-based proxy services.</p> <p>The XPath expression is evaluated at runtime to produce the property's value. A <code>ContextHandler</code> is essentially a name/value list and, as such, it requires that a security provider know what names to look for. Therefore, the XPath expressions are evaluated only if a security provider asks for the value of one of these user-defined properties.</p> <p>Click Add Property to add this context property. You can add multiple context properties.</p>

4.4 Oracle Service Bus Configurations and Projects

The following are provided for working with Oracle Service Bus configurations and projects.

- [Section 4.4.1, "Oracle Service Bus Configurations View"](#)
- [Section 4.4.2, "New Oracle Service Bus Configuration Project Wizard"](#)
- [Section 4.4.4, "New Oracle Service Bus Project"](#)

4.4.1 Oracle Service Bus Configurations View

This view displays a hierarchical list of all your Oracle Service Bus configurations and the projects they contain. See also [Section 4.4.2, "New Oracle Service Bus Configuration Project Wizard."](#)

You can do the following in the Configurations view:

- Edit the configuration:
 1. Right click the configuration you want to edit.
 2. Select **Properties** from the menu.
 3. Modify information as needed in the [Oracle Service Bus Configuration Page](#).
- Create a new configuration:
 1. Right-click anywhere in the view.
 2. Select **New Oracle Service Bus Configuration** from the menu.
 3. Enter information as needed in the [Oracle Service Bus Configuration Page](#).
- Move a project from one configuration to another by dragging it from the source configuration to the target configuration. You can also drag a project from the Project Explorer.
- Delete a configuration project:
 1. Right-click the configuration you want to delete.
 2. Select **Delete** from the menu.

4.4.2 New Oracle Service Bus Configuration Project Wizard

Use this wizard to create an Oracle Service Bus configuration project. For configuration options, see [Section 4.4.3, "Oracle Service Bus Configuration Page."](#)

4.4.3 Oracle Service Bus Configuration Page

An Oracle Service Bus configuration project is a grouping of Oracle Service Bus projects and resources destined for a server, a location for system resources (SMTP, UDDI, and such), and a container for validation; for example, a resource in a project associated with one Oracle Service Bus configuration cannot refer to a resource in a project associated to another Oracle Service Bus configuration.

Use this page to create a new Oracle Service Bus configuration project (in the New Oracle Service Bus Configuration Project wizard) or to configure an existing one (in the Properties for Oracle Service Bus Configuration editor).

The preserve, session, and customization file settings are used when publishing an Oracle Service Bus configuration to the server, and used as defaults when configuring the [Export Wizard - Export to Server - Select Resources Page](#).

Table 4–18 Oracle Service Bus Configuration Options

Option	Description
Configuration name	Enter a name for this Oracle Service Bus configuration or keep the default.
Preserve environment variable values	Select this check box when you are re-importing a resource but want to preserve environment variable values in the existing resource.
Preserve security and policy settings	Select this check box to preserve the security configuration (excluding access control policies) and the references to the WS-policies bound directly to the service (instead of bound to the WSDL).

Table 4–18 (Cont.) Oracle Service Bus Configuration Options

Option	Description
Preserve credentials (user name and password)	Select this check box to preserve PKI credentials in service key providers, user name and passwords in service accounts, and user name and password credentials in SMTP servers, JNDI providers, and UDDI registries.
Discard session if activation fails	Select this check box to discard the session if the activation fails (for example, due to conflicts).
Session Name	The session name.
Description	The session description.
Deployment customization file	Specify a <i>customization.xml</i> file or click Browse , locate the file, then click Open . For information on customization, see "Creating Customization Files" in the <i>Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBAG1360 .
Keystore file	Specify a <i>keystore.jks</i> file or click Browse , locate the file, then click Open . The key store settings are used when configuring a service key provider.
Password	Enter the password that you use to secure access to the key store.
Server	The name of the server associated with this Oracle Service Bus configuration. This setting is automatically configured unless there is more than one server from which to choose. When multiple servers are associated with the same Oracle Service Bus configuration, use the list to select the server you want to associate with this Oracle Service Bus configuration. The server setting is only used for transport specific configuration, when the transport benefits from being connected to a server (for example, when configuring the dispatch policy setting in the HTTP transport).

4.4.4 New Oracle Service Bus Project

Use this page to create a new Oracle Service Bus project.

Table 4–19 New Oracle Service Bus Project Options

Option	Description
Project name	Enter a unique name for the project.
Oracle Service Bus Configuration	Select an existing Oracle Service Bus configuration or click New to open the New Oracle Service Bus Configuration Project Wizard , where you can create a new configuration.

4.5 Custom Resources

You can define custom resources for use by Oracle Service Bus using the New Custom Resource wizard and the Custom Resource editor, as described in the following topics:

- [Section 4.5.1, "New Custom Resource Wizard"](#)
- [Section 4.5.1, "New Custom Resource Wizard"](#)
- [Section 4.5.3, "New Custom Resource - Resource Type Page"](#)

4.5.1 New Custom Resource Wizard

Use this wizard to create a new custom resource.

4.5.2 New Custom Resource Editor

Use this editor to modify the configuration of a custom resource. The General page identifies the type of custom resource. The Custom page provides options for editing the configuration for that type of custom resource.

4.5.3 New Custom Resource - Resource Type Page

Use this page to select the type of custom resource to create. Select the type of custom resource to create from the list of available resource type.

4.5.4 Custom MQ Resource Configuration Page

Use this page to configure a custom MQ connection resource. For information on MQ connection resources and MQ transports, see the following:

- "MQ Connections" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBAG909>
- "MQ Transport" in the *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBDV1117>.

Table 4–20 MQ Resource Options

Option	Description
Connection Type	Select one of the following modes for connecting to the MQ queue manager: <ul style="list-style-type: none"> ■ <code>tcp mode</code>—Use TCP/IP to connect to a queue manager that does not reside on the same machine as Oracle Service Bus. ■ <code>binding mode</code>—Use the bindings mode to connect to a queue manager that is located on the same machine as Oracle Service Bus.
MQ Host Name	For <code>tcp mode</code> connections only: Enter the host name of the MQ queue manager.
MQ Port Number	For <code>tcp mode</code> connections only: Enter the port number of the MQ queue manager listener.
MQ Queue Manager Name	Enter the name of the MQ queue manager to which to connect.
Queue Manager CCSID	For <code>tcp mode</code> connections only: The coded character set identifier (CCSID) to be used when establishing a connection. The CCSID is used mainly for internationalization support. To learn more, see IBM's WebSphere MQ Fundamentals at http://www.redbooks.ibm.com/redbooks/SG247128/wwhelp/wwhimpl/java/html/wwhelp.htm .
MQ Queue Manager Channel Name	For <code>tcp mode</code> connections only: Enter the queue manager server connection channel name.

Table 4–20 (Cont.) MQ Resource Options

Option	Description
SSL Required	For <code>tcp</code> mode connections: Select the check box to use SSL for sending messages. Only server-side SSL is enabled when the 2-way SSL Required option is <i>not</i> selected.
Cipher Suite	This option is available only when the SSL Required check box is selected. Select the Cipher Suite algorithm to be used by SSL. The Cipher Suite algorithm is used to encrypt and decrypt message communications between the WebSphere MQ server and the WebSphere MQ client. Thus a Cipher Suite algorithm must be specified when using SSL to communicate with a WebSphere MQ server.
2-way SSL Required	This option is available only when the SSL Required check box is selected. Select the check box to enable both client-side and server-side SSL authentication.
Reference to the Service Key Provider	If you selected 2-way SSL Required , you must provide a reference to the service key provider for obtaining the appropriate key store and trust store information. Enter the path (project/folder) and name of a service key provider, or click Browse to select one from the Select Service Key Provider page.
Reference to the Static Service Account	For <code>tcp</code> mode connections only: Required for user name and password authentication. Enter the path (project/folder) and name of a static service account, or click Browse to select service accounts from a browser.
WebSphere MQ Version	Select the WebSphere MQ version: <ul style="list-style-type: none"> ■ 5.3 ■ 6.0
MQ Connection Pool Size	Enter the size of the MQ connection pool.
MQ Connection Timeout	Enter the time interval in seconds after which unused connections are destroyed. The default is 1800 seconds.
MQ Connection Max Wait	Enter the Max Wait in seconds for the amount of time to wait for a connection to become available. If a connection is not made within that time interval, Oracle Service Bus throws an exception. The default is 3 seconds.

4.6 Export Wizard

The Export wizard contains the following pages:

- [Section 4.6.1, "Export Wizard - Oracle Service Bus Configuration JAR Export Page"](#)
- [Section 4.6.2, "Export Wizard - Export to Server - Select Resources Page"](#)
- [Section 4.6.3, "Export Wizard - Export to Server - Review Resources Page"](#)

4.6.1 Export Wizard - Oracle Service Bus Configuration JAR Export Page

Use this page to export Oracle Service Bus resources to a configuration JAR file. For more information, see "Exporting Resources" in the *Oracle Fusion Middleware*

Administrator's Guide for Oracle Service Bus at
<http://www.oracle.com/pls/as1111/lookup?id=OSBAG1320>.

Table 4–21 Configuration JAR Export Options

Option	Description
Oracle Service Bus Configuration	Select an existing Oracle Service Bus configuration and resources to export.
Export Level	Select whether to export projects or resources. Notes A System project cannot be exported at the project level. Exporting projects might cause resource deletion when you import the full project JAR file.
Include Dependencies	If you selected to export resources, select or clear this check box. Use the Include Dependencies option to export any other resources that this resource references
Export Destination	Enter the fully qualified name of a JAR file to export, or click Browse to select it.

4.6.2 Export Wizard - Export to Server - Select Resources Page

Use this page to select the projects or resources you want to export. Clear the check boxes next to any resources that you do not want to include in this export.

Table 4–22 Resource Export Options

Option	Description
Resource	The name of the project and resource.
Operation	Create or update. The operation that will be performed on the resource.
Include Dependencies	Select this check box if you are exporting resources (not projects) and want to ensure that all the associated resources are exported.
Preserve environment variable values	Select this check box when you want to preserve (protect against overwriting) the environment variables values in the resource you are exporting.
Preserve security and policy settings	Select this check box to preserve the security configuration (excluding access control policies) and the references to the WS-policies bound directly to the service (instead of bound to the WSDL).
Preserve credentials (user name and password)	Select this check box to preserve PKI credentials in service key providers, user name and passwords in service accounts, and user name and password credentials in SMTP servers, JNDI providers, and UDDI registries.
Activate session after publish	Select this check box to create and activate a session in the Oracle Service Bus console.
Discard session if activation fails	Select this check box to discard the session if the activation fails (for example, due to conflicts).
Description	The session description.

Table 4–22 (Cont.) Resource Export Options

Option	Description
Customization File	Specify a customization file or click Browse , locate the file, then click Open . For information on customization, see "Creating Customization Files" in the <i>Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/as1111/lookup?id=OSBAG1360 .

4.6.3 Export Wizard - Export to Server - Review Resources Page

Use this page to select the projects or resources you want to export.

Table 4–23 Resource Export Options

Option	Description
Oracle Service Bus Configuration	Select an existing Oracle Service Bus configuration and resources to export.
Export Level	Select whether to export projects or resources.
Server	From the list, select an Oracle Service Bus destination server.
Session	The session name.

4.7 Import Wizard

The Import wizard has the following pages:

- [Section 4.7.1, "Import Wizard - Config JAR Import - Load Resources Page"](#)
- [Section 4.7.2, "Import Wizard - Config JAR Import - Review Resources Page"](#)
- [Section 4.7.3, "Import Wizard - Config ZIP Import - Load Resources Page"](#)
- [Section 4.7.4, "Import Wizard - Config ZIP Import - Review Resources Page"](#)
- [Section 4.7.5, "Import Wizard - URL Import - Load Resources Page"](#)
- [Section 4.7.6, "Import Wizard - URL Import - Review Resources Page"](#)

4.7.1 Import Wizard - Config JAR Import - Load Resources Page

Use this page to select a configuration JAR to import and an import destination. For more information, see "Importing Resources" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBAG1312>.

Table 4–24 Configuration JAR Import

Option	Description
Oracle Service Bus Configuration	Select an existing Oracle Service Bus configuration or click New to open the New Oracle Service Bus Configuration Project Wizard , where you can create a new configuration.
Jar	Select a full project JAR file or a resource JAR file that has been previously exported from another Oracle Service Bus domain.

4.7.2 Import Wizard - Config JAR Import - Review Resources Page

Use this page to select the objects on which you want to operate (create, update, or delete). Resources are only scheduled for deletion when the JAR being imported is a full project JAR and there are project resources in the importing system that are not present in the imported JAR file. To prevent resources from being deleted, deselect them.

Table 4–25 Configuration JAR Import Options

Option	Description
Resources	The name of the project and resource.
Operation	Expand the Project folder to display the operations (create, update, delete) that will be performed on the resources. To prevent resources from being deleted, deselect them. Resources are only scheduled for deletion when the JAR being imported is a full project JAR and there are project resources in the importing system that are not present in the imported JAR file.
Include Dependencies	Select this check box if you are importing a resource JAR file and want to ensure that all its associated resources are imported.
Passphrase	Enter the password that was used to encrypt the data.
Preserve environment variable values	Select this check box when you are re-importing a resource but want to preserve environment variable values in the existing resource.
Preserve security and policy settings	Select this check box to preserve the security configuration (excluding access control policies) and the references to the WS-Policies bound directly to the service (instead of bound to the WSDL).
Preserve credentials (user name and password)	Select this check box to preserve PKI credentials in service key providers, user name and passwords in service accounts, and user name and password credentials in SMTP servers, JNDI providers, and UDDI registries.

4.7.3 Import Wizard - Config ZIP Import - Load Resources Page

Use this page to select a ZIP file with resources to import and an import destination.

Table 4–26 Configuration ZIP Import

Option	Description
Oracle Service Bus Configuration	Select an existing Oracle Service Bus configuration and an import destination (project or folder).
File Name	Enter the fully qualified name of a ZIP file to import, or click Browse to select it.

4.7.4 Import Wizard - Config ZIP Import - Review Resources Page

Use this page to select the objects you want to import. Clear the check boxes next to any resources that you do not want to include in this import.

Table 4–27 Configuration ZIP Import Options

Option	Description
Resource	The name of the project and resource.

Table 4–27 (Cont.) Configuration ZIP Import Options

Option	Description
Operation	Create or update. The operation that will be performed on the resource.
File Name	The file name of the resource, including the file extension.

4.7.5 Import Wizard - URL Import - Load Resources Page

Use this page to import resources such as WSDLs or XML schemas that are available on the Web. You can import them, along with all their dependents, by specifying the URL of the root resource.

Table 4–28 URL Import Options

Option	Description
Oracle Service Bus Configuration	Select an existing Oracle Service Bus configuration and an import destination (project or folder).
URL	Enter the URL where the file is located. To specify a local resource, you can use the <code>file</code> protocol with a file name fully qualified by its path, for example, <code>file:///c:/alsbresources/ForeachAction.jar</code> .
Resource Name	Enter a name for the resource.
Resource Type	Select the type of resource from the list.

4.7.6 Import Wizard - URL Import - Review Resources Page

Use this page to select the objects you want to import. Clear the check boxes next to any resources that you do not want to include in this import.

Table 4–29 URL Import Options

Option	Description
Resource	The name of the project and resource.
Operation	Create or update. The operation that will be performed on the resource.
URL	Enter the URL where the file is located. To specify a local resource, you can use the <code>file</code> protocol with a file name fully qualified by its path, for example, <code>file:///c:/alsbresources/ForeachAction.jar</code> .

4.8 JNDI Providers

The following are provided for working with JNDI providers:

- [Section 4.8.1, "JNDI Provider Editor"](#)
- [Section 4.8.2, "New JNDI Provider Resource Wizard"](#)

4.8.1 JNDI Provider Editor

Use this editor to modify an existing JNDI provider configuration. For descriptions of the fields, see the [Section 4.8.2, "New JNDI Provider Resource Wizard."](#)

4.8.2 New JNDI Provider Resource Wizard

Use this page to configure a new JNDI provider resource.

Table 4–30 *New JNDI Provider Resource Options*

Option	Description
Description	Enter a description for the JNDI provider.
JNDI Cache	Keep the default Enabled option or select Disabled . When enabled, the JNDI context and JNDI objects are cached locally which improves performance when doing an object lookup. Oracle recommends that you keep the JNDI cache enabled.
Provider URL	Enter the URL for the JNDI provider in the format: <code>protocol://host:port</code> You can use any protocol, for example: <code>http, https, t3, t3s, iiop, iiops</code>
JNDI Request Timeout	The JNDI request timeout in milliseconds. The default of zero (0) means no timeout.
User Name	If access to the target JNDI provider requires a user name and password, enter a user name in the User Name field, and the associated password in the Password and Confirm Password fields. These fields are optional, and required only if the JNDI tree is secured.
Password	Enter the associated password.
Confirm Password	Enter the same password you entered for the Password field.

4.9 Proxy Servers

Use this page to configure and edit a proxy server resource after you create a new proxy server.

Table 4–31 *Proxy Server Configuration Options*

Option	Description
Description	Enter a description of the proxy server resource.
User Name	Enter the user name used for proxy authentication.
Password	Enter the password associated with the user name.
Confirm Password	Enter the same password you entered for the Password field.

Table 4–31 (Cont.) Proxy Server Configuration Options

Option	Description
Host-Port Parameters	<p>The list of proxy server hosts. You can configure multiple proxy servers for each proxy server resource. This enables Oracle Service Bus to perform load balancing and offer fault tolerance features for the proxy server resource.</p> <p>The following options are available:</p> <ul style="list-style-type: none"> ■ Click Add and specify the following information to configure a new proxy server for the resource: ■ Server Host: The host name or IP address of the proxy server. The Server Host name for the Oracle Service Bus proxy server must be identical to the server host name of the actual proxy server. ■ Clear Text Port: The proxy server clear-text port number. ■ SSL Port: The proxy server SSL port number. ■ Select a proxy server in the list and edit the Host-Port Parameters. ■ Select a proxy server in the list and click Delete to remove it from the resource configuration.

4.10 Message Flow Design Palette

The Message Flow Design Palette lists all the nodes and actions you can use to construct a message flow for a proxy service. To insert a node or an action into a message flow, drag the item from the palette to the [Message Flow Editor](#). When you drag an item to the palette, one or more target icons indicate that you can drop the dragged item in that position on the palette. When you drag the item onto a target, the target icon is highlighted to show that you can drop the item there.

You can also add nodes and actions to message flows by right-clicking an object in the Message Flow Editor and choosing a node or action from the **Insert**, **Insert Into**, or **Insert After** submenus. You can also add an error handler by selecting the **Add Error Handler** command. These submenus and commands are context sensitive. That is, they appear only when it is legal to add the item into the selected context in the message flow.

The Message Flow Design Palette is organized into the following categories:

- [Section 4.10.1, "Message Flow Nodes"](#)
- [Section 4.10.2, "Message Flow Route Actions - Communication Actions"](#)
- [Section 4.10.3, "Message Flow Route Actions - Flow Control Actions"](#)
- [Section 4.10.4, "Message Flow Stage Actions - Communication Actions"](#)
- [Section 4.10.5, "Message Flow Stage Actions - Flow Control Actions"](#)
- [Section 4.10.6, "Message Flow Stage Actions - Message Processing Actions"](#)
- [Section 4.10.7, "Message Flow Stage Actions - Reporting Actions"](#)

4.10.1 Message Flow Nodes

The following table describes the nodes you can add to a message flow:

Table 4–32 Message Flow Nodes

Node	Description
Conditional branch	A branch node allows processing to proceed down exactly one of several possible paths. For configuration properties, see Section 4.11.3, "Conditional Branch Node Properties."
Operational branch	An operational branch node determines what branch to follow based on specified operations. For configuration properties, see Section 4.11.17, "Operational Branch Node Properties."
Pipeline pair	A pipeline pair node consists of a request pipeline and a response pipeline. For configuration properties, see Section 4.11.18, "Pipeline Pair Node Properties."
Route	Route node actions define the handling of messages as they flow through the route node. For configuration properties, see Section 4.11.27, "Route Node Properties."
Stage	A stage node is a container of actions. For configuration properties, see Section 4.11.33, "Stage Node Properties."
Error handler	An error handler provides the logic for resending errors in the message flow. For configuration properties, see Section 4.11.7, "Error Handler Node Properties."

4.10.2 Message Flow Route Actions - Communication Actions

The following table describes the communication actions you can add to a route in a message flow:

Table 4–33 Message Flow Route Node Communication Actions

Action	Description
Dynamic routing	A dynamic routing action assigns a route for a message based on routing information available in an XQuery resource. For configuration properties, see Section 4.11.6, "Dynamic Routing Action Properties."
Routing	A routing action identifies a target service for the message and configures how the message is routed to that service: For configuration properties, see Section 4.11.28, "Routing Action Properties."
Routing table	A routing table action assigns a set of routes wrapped in a switch-style condition table. Different routes are selected based upon the results of a single XQuery expression. For configuration properties, see Section 4.11.30, "Routing Table Action Properties."

4.10.3 Message Flow Route Actions - Flow Control Actions

The following table describes the flow control action you can add to a route in a message flow:

Table 4–34 Message Flow Route Node Flow Control Actions

Action	Description
If-then	An if-then action performs an action or a set of actions conditionally, based on the Boolean result of an XQuery expression. For configuration properties, see Section 4.11.9, "If-Then Action Properties."

4.10.4 Message Flow Stage Actions - Communication Actions

The following table describes the communication actions you can add to a stage in a message flow:

Table 4–35 Message Flow Stage Node Communication Actions

Action	Description
Dynamic Publish	A dynamic publish action publishes a message to a service identified by an XQuery expression For configuration properties, see Section 4.11.5, "Dynamic Publish Action Properties."
Publish	A publish action publishes a message to a statically specified service. For configuration properties, see Section 4.11.19, "Publish Action Properties."
Publish Table	A publish table action publishes a message to zero or more statically specified services. Switch-style condition logic is used to determine at run time which services will be used for the publish. For configuration properties, see Section 4.11.20, "Publish Table Action Properties."
Routing Options	A routing options action modifies any or all of the following properties in the outbound request: URI, Quality of Service, Mode, Retry parameters, Message Priority. For configuration properties, see Section 4.11.29, "Routing Options Action Properties."
Service Callout	A service callout action configures a synchronous (blocking) callout to an Oracle Service Bus-registered proxy or business service. For configuration properties, see Section 4.11.31, "Service Callout Action Properties."
Transport Headers	A transport header action sets the transport header values in messages For configuration properties, see Section 4.11.34, "Transport Headers Action Properties."

4.10.5 Message Flow Stage Actions - Flow Control Actions

The following table describes the flow control actions you can add to a stage in a message flow:

Table 4–36 Message Flow Stage Node Flow Control Actions

Action	Description
For-Each	A for-each action iterates over a sequence of values and executes a block of actions For configuration properties, see Section 4.11.8, "For-Each Action Properties."
If-Then	An if-then action performs an action or set of actions conditionally, based on the Boolean result of an XQuery expression. For configuration properties, see Section 4.11.9.1, "If Action and Else-If Action Properties."
Raise error	A raise-error action raises an exception with a specified error code (a string) and description. For configuration properties, see Section 4.11.21, "Raise Error Action Properties."
Reply	A reply action specifies that an immediate reply be sent to the invoker. For configuration properties, see Section 4.11.24, "Reply Action Properties."
Skip	A skip action specifies that at run time, the execution of the current stage is skipped and the processing proceeds to the next stage in the message flow. For configuration properties, see Section 4.11.32, "Skip Action Properties."
Resume	A resume action resumes message flow after an error is handled by an error handler. For configuration properties, see Section 4.11.26, "Resume Action Properties."

4.10.6 Message Flow Stage Actions - Message Processing Actions

The following table describes the message processing actions you can add to a stage in a message flow:

Table 4–37 Message Flow Stage Node Message Processing Actions

Action	Description
Assign	An assign action assigns the result of an XQuery expression to a context variable. For configuration properties, see Section 4.11.2, "Assign Action Properties."
Delete	A delete action deletes a context variable or a set of nodes specified by an XPath expression. For configuration properties, see Section 4.11.4, "Delete Action Properties."
Insert	An insert action inserts the result of an XQuery expression at an identified place relative to nodes selected by an XPath expression. For configuration properties, see Section 4.11.10, "Insert Action Properties."
Java callout	A Java callout action invokes a Java method from the pipeline. For configuration properties, see Section 4.11.11, "Java Callout Action Properties."

Table 4–37 (Cont.) Message Flow Stage Node Message Processing Actions

Action	Description
MFL transform	A MFL transform action converts non-XML to XML or XML to non-XML in the pipeline. For configuration properties, see Section 4.11.16, "MFL Transform Action Properties."
Rename	A rename action renames elements selected by an XPath expression without modifying the contents of the element. For configuration properties, see Section 4.11.22, "Rename Action Properties."
Replace	A replace action replaces a node or the contents of a node specified by an XPath expression. For configuration properties, see Section 4.11.23, "Replace Action Properties."
Validate	A validate action validates elements selected by an XPath expression against an XML schema element or a WSDL resource. For configuration properties, see Section 4.11.35, "Validate Action Properties."

4.10.7 Message Flow Stage Actions - Reporting Actions

The following table describes the reporting actions you can add to a stage in a message flow:

Table 4–38 Message Flow Stage Node Reporting Actions

Action	Description
Alert	An alert action ends an alert notification based on pipeline message context. For configuration properties, see Section 4.11.1, "Alert Action Properties."
Log	A log action constructs a message to be logged. For configuration properties, see Section 4.11.12, "Log Action Properties."
Report	A report action enables message reporting for a proxy service. For configuration properties, see Section 4.11.25, "Report Action Properties."

4.11 Message Flow Editor

Use this editor to construct a proxy service message flow. To insert a node or an action into a message flow, drag the item from the [Message Flow Design Palette](#) to the editor. When you drag an item to the editor, one or more target icons indicate that you can drop the dragged item in that position on the palette. When you drag the item onto a target, the target icon is highlighted to show that you can drop the item there.

You can also add nodes and actions to message flows by right-clicking an object in the Message Flow Editor and choosing a node or action from the **Insert**, **Insert Into**, or **Insert After** submenus. You can also add an error handler by selecting the **Add Error Handler** command. These submenus and commands are context sensitive. That is, they appear only when it is legal to add the item into the selected context in the message flow.

When you select a node or action in the editor, a configuration page for that item is displayed in the Properties view. You can add or modify configuration properties in those pages. The message flow node and action properties configuration pages are:

- [Section 4.11.1, "Alert Action Properties"](#)
- [Section 4.11.13, "Message Flow Properties - Comment"](#)
- [Section 4.11.2, "Assign Action Properties"](#)
- [Section 4.11.3, "Conditional Branch Node Properties"](#)
- [Section 4.11.3, "Conditional Branch Node Properties"](#)
- [Section 4.11.4, "Delete Action Properties"](#)
- [Section 4.11.5, "Dynamic Publish Action Properties"](#)
- [Section 4.11.6, "Dynamic Routing Action Properties"](#)
- [Section 4.11.7, "Error Handler Node Properties"](#)
- [Section 4.11.7, "Error Handler Node Properties"](#)
- [Section 4.11.8, "For-Each Action Properties"](#)
- [Section 4.11.9.1, "If Action and Else-If Action Properties"](#)
- [Section 4.11.9, "If-Then Action Properties"](#)
- [Section 4.11.10, "Insert Action Properties"](#)
- [Section 4.11.11, "Java Callout Action Properties"](#)
- [Section 4.11.12, "Log Action Properties"](#)
- [Section 4.11.16, "MFL Transform Action Properties"](#)
- [Section 4.11.14, "Message Flow Properties - Namespaces"](#)
- [Section 4.11.17, "Operational Branch Node Properties"](#)
- [Section 4.11.18, "Pipeline Pair Node Properties"](#)
- [Section 4.11.19, "Publish Action Properties"](#)
- [Section 4.11.20, "Publish Table Action Properties"](#)
- [Section 4.11.21, "Raise Error Action Properties"](#)
- [Section 4.11.22, "Rename Action Properties"](#)
- [Section 4.11.23, "Replace Action Properties"](#)
- [Section 4.11.24, "Reply Action Properties"](#)
- [Section 4.11.25, "Report Action Properties"](#)
- [Section 4.11.26, "Resume Action Properties"](#)
- [Section 4.11.27, "Route Node Properties"](#)
- [Section 4.11.28, "Routing Action Properties"](#)
- [Section 4.11.29, "Routing Options Action Properties"](#)
- [Section 4.11.30, "Routing Table Action Properties"](#)
- [Section 4.11.31, "Service Callout Action Properties"](#)
- [Section 4.11.32, "Skip Action Properties"](#)
- [Section 4.11.33, "Stage Node Properties"](#)

- [Section 4.11.34, "Transport Headers Action Properties"](#)
- [Section 4.11.35, "Validate Action Properties"](#)
- [Section 4.11.15, "Message Flow Properties - Variables"](#)

4.11.1 Alert Action Properties

In a message flow, use the alert action to generate alerts based on message context in a pipeline, to send to an alert destination. Unlike SLA alerts, notifications generated by the alert action are primarily intended for business purposes, or to report errors, and not for monitoring system health. Alert destinations should be configured and chosen with this in mind. To learn more about alert destinations, see [Section 4.1.1, "Alert Destination Editor."](#)

If pipeline alerting is not enabled for the service or at the domain level, the configured alert action is bypassed during message processing.

In the [Message Flow Editor](#), click an alert action to display its properties in the Properties view. Use these properties pages to configure the selected alert action. The pages are:

- Alert
- Comment
- Namespaces
- Variables

The Alert page has the following options:

Table 4–39 Alert Action Options

Option	Description
Expression	An XQuery expression that specifies the message context to be added to the alert message. To create or edit an expression, click <Expression> (or the <i>expression fragment</i> , if one is already defined) to display the XQuery/XSLT Expression Editor .
Summary	A short description of the alert. This will be the subject line in the case of an e-mail notification and can contain no more than 80 characters. If no description is provided, a predefined subject line that reads, "ALSB Alert," will be used instead.
Severity	The severity level for this alert. Select a level from the list.
Destination	The destination for the alert. To specify a destination, click Browse to select an appropriate resource.

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.2 Assign Action Properties

In a message flow, use an assign action to assign the result of an XQuery expression to a context variable.

In the [Message Flow Editor](#), click an assign action to display its properties in the Properties view. Use these properties pages to configure the selected assign action. The pages are:

- Assign
- Comment
- Namespaces
- Variables

The Assign page has the following options:

Table 4–40 Assign Action Options

Option	Description
Expression	An expression that creates the data that is assigned to variable named in the Variable field, described below. To create or edit an expression, click <Expression> (or the expression_fragment , if one is already defined) to display the XQuery/XSLT Expression Editor .
Variable	The variable to which the value created in the XQuery expression described above is assigned.

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.3 Conditional Branch Node Properties

In a message flow, use a conditional branch node to specify that message processing is to proceed along exactly one of several possible paths, based on a result returned by an XPath condition.

Conditional branching is driven by a lookup table with each branch tagged with a simple, but unique, string value. A variable in the message context is designated as the lookup variable for that node, and at run time, its value is used to determine which branch to follow. If no branch matches the value of the lookup variable, the default branch is followed. You should design the proxy service in such a way that the value of the lookup variable is set before reaching the branch node.

In the [Message Flow Editor](#), click a conditional branch node to display its properties in the Properties view. Use these properties pages to configure the selected conditional branch node. The pages are:

- Flow
- Conditional Branch

The Flow page has the following options:

Table 4–41 Flow Options

Option	Description
Name	Enter a name for the conditional branch node.
Description	Enter a description for the conditional branch node.

The Conditional Branch page has the following options:

Table 4–42 Conditional Branch Options

Option	Description
XPath	The XPath expression that defines the condition to be evaluated for determining the branch to follow. To create or edit the XPath expression, click <XPath> (or the XPath fragment , if one is already defined) to display the XPath Expression Editor .
Variable	Enter a context variable.

A conditional branch node also always contains one or more branches, which are configured as part of the node.

Click a **Branch** icon in a conditional branch node to display the Branch properties page in the Properties view. Use these properties pages to configure the selected branch node in a message flow. This page has the following options.

Table 4–43 Branch Options

Option	Description
Label	A label for the branch. This label appears as the label for the branch in the Message Flow Editor.
Operator	Select an operator from the list to be used with the value in the next field, for creating the condition on which the branch is based.
Value	Enter a value to be used with the operator in the previous field, for creating the condition

4.11.4 Delete Action Properties

In a message flow, use a delete action to delete a context variable or a set of nodes specified by an XPath expression.

In the [Message Flow Editor](#), click a delete action to display its properties in the Properties view. Use these properties pages to configure the selected delete action. The pages are:

- Delete
- Comment
- Namespaces
- Variables

The Delete page has the following options. Select **Variable** to delete a variable, or select **XPath** to delete an XPath expression.

Table 4–44 Delete Action Options

Option	Description
Variable	Select this radio button to delete a context variable. Enter the name of the context variable to delete in the adjacent text field.

Table 4–44 (Cont.) Delete Action Options

Option	Description
XPath	<p>Select this radio button to delete all nodes selected by an XPath expression. Then:</p> <ul style="list-style-type: none"> ■ To create or edit the XPath expression, click <XPath> (or the <i>xPath_fragment</i>, if one is already defined) to display the XPath Expression Editor. ■ After saving the expression, in the In Variable field, enter the context variable containing the nodes specified in the XPath expression.

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.5 Dynamic Publish Action Properties

In a message flow, use a dynamic publish action to publish a message to a service specified by an XQuery expression.

In the [Message Flow Editor](#), click a dynamic publish action to display its properties in the Properties view. Use these properties pages to configure the selected dynamic publish action. The pages are:

- Dynamic Publish
- Comment
- Namespaces
- Variables

The Dynamic Publish page has the following options:

Table 4–45 Dynamic Publish Action Options

Option	Description
Service	<p>An XQuery expression that defines the service to which a message is to be published.</p> <p>To create or edit the expression, click <Expression> (or the <i>expression_fragment</i>, if one is already defined) to display the XQuery/XSLT Expression Editor.</p> <p>In the editor, enter an Xquery expression or select an XQuery resource that provides a result similar to:</p> <pre><ctx:route isProxy="false"> <ctx:service>project/folder/businessservicename</ctx:service> <ctx:operation>foo</ctx:operation> </ctx:route></pre> <p>Note: The element <code>operation</code> is optional.</p>

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.6 Dynamic Routing Action Properties

In a message flow, use a dynamic routing action to assign a route for a message based on routing information available in an XQuery resource.

This is a terminal action, which means you cannot add another action after this one. However, this action can contain request and response actions.

In the [Message Flow Editor](#), click a dynamic routing action to display its properties in the Properties view. Use these properties pages to configure the selected dynamic routing action. The pages are:

- Dynamic Routing
- Comment
- Namespaces
- Variables

The Dynamic Routing page has the following option:

Table 4–46 Dynamic Routing Action Properties

Option	Description
Service	<p>An XQuery expression that defines the route to be taken by a message.</p> <p>To create or edit an expression, click <Expression> (or the <i>expression fragment</i>, if one is already defined) to display the XQuery/XSLT Expression Editor.</p> <p>In the XQuery/XSLT Expression editor, enter an Xquery expression, the result of which is similar to:</p> <pre><ctx:route> <ctx:service isProxy='true'>{\$service}</ctx:service> <ctx:operation>{\$operation}</ctx:operation> </ctx:route></pre> <p>Note: If a proxy service is being invoked, <code>isProxy</code> attribute should be set to true.</p> <ul style="list-style-type: none"> ■ The service name is the fully qualified service name. ■ The operation element is optional

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.7 Error Handler Node Properties

Use an error handler to specify what should happen if an error occurs in a specific location in the message flow.

All configuration for an error handler is in one or more stages contained by the error handler. See [Section 4.11.33, "Stage Node Properties."](#)

4.11.8 For-Each Action Properties

In a message flow, use the for-each action to iterate over a sequence of values and execute a block of actions.

In the [Message Flow Editor](#), click a for-each action to display its properties in the Properties view. Use these properties pages to configure the selected for-each action. The pages are:

- For Each
- Comment
- Namespaces
- Variables

The For Each page has the following options:

Table 4–47 For-Each Action Properties

Option	Description
For Each Variable	Enter the name of the variable on whose contents the for-each action(s) will be executed.
XPath	An XPath expression that specifies where in the structure of the containing context variable the variable specified in the For Each Variable field is located. To create or edit the XPath expression, click <XPath> (or the XPath fragment , if one is already defined) to display the XPath Expression Editor .
In Variable	The context variable containing the variable on whose contents the for-each action(s) will be executed.
Index Variable	A variable containing the current number of iterations in the loop.
Count Variable	A variable containing the total number of iterations.

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.9 If-Then Action Properties

In a message flow, use an if-then action to perform an action or a set of actions conditionally, based on the Boolean result of an XQuery expression.

In the [Message Flow Editor](#), click an if-then action to display its properties in the Properties view. Use this page to review and configure general properties of the if-then action. The pages are:

- Comment
- Namespaces
- Variables

An if-then action always contains an if condition plus zero or more else-if condition, where you define the conditions for the if-then action. An if-then action also always contains an else condition, which defines the default path when no other condition is met.

Click an **If:condition** icon or an **Else-If:condition** icon in an if-then action to display their properties in the Properties view. Use these properties pages to configure the selected condition. This **If** condition properties page has the following option.

4.11.9.1 If Action and Else-If Action Properties

If actions and else-if actions are always contained by an if-then action. The message follows the path associated with the first if action or else-if action that returns true.

Table 4–48 If and Else-If Action Options

Option	Description
Condition	<p>A condition in an if action or an if-else action that contains one of the tests for following alternative paths in the flow.</p> <p>To add or edit a condition</p> <ol style="list-style-type: none"> 1. Click <Condition> (or <i>condition fragment</i>, if one is already defined) to display the Condition Editor. Define a condition to be evaluated in the if-then action. 2. When you finish editing the condition, add one or more action that will be executed when the condition evaluates to true. To add an action, drag it from the Design Palette to associate with the condition. <p>In the route node, you can select only the routing, dynamic routing, or routing table actions. However, these actions can contain request and response actions.</p> <p>To add an Else-If condition:</p> <p>Click add condition. Then add one or more actions to this else-if condition, as described above for the If condition. Continue with as many else-if conditions as your logic requires.</p> <p>Condition actions can be nested.</p>

4.11.9.2 Else Action Properties

The else action does not have associate properties. When all if and else-if conditions return false, the message proceeds down the path associated with the else action.

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.10 Insert Action Properties

In a message flow, use an insert action to insert the result of an XQuery expression at an identified place relative to nodes selected by an XPath expression.

In the [Message Flow Editor](#), click an insert action to display its properties in the Properties view. Use these properties pages to configure the selected insert action. The pages are:

- Insert
- Comment
- Namespaces
- Variables

The Insert page has the following options:

Table 4–49 Insert Action Options

Option	Description
Expression	The XQuery expression used to create the data that will be inserted at a specified location in a named variable. To create or edit an expression, click <Expression> (or the expression_fragment , if one is already defined) to display the XQuery/XSLT Expression Editor .
Location	The location where the insert is performed, relative to the result of the XPath expression: Options are: <ul style="list-style-type: none"> ■ Before—as sibling before each element or attribute selected by the XPath expression ■ After—as sibling after each element or attribute selected by the XPath expression ■ As first child of—as first child of each element identified by the XPath expression. An error occurs if the result of the XPath returns attributes. ■ As last child of—as last child of each element identified by the XPath expression. An error occurs if the XPath returns attributes.
XPath	Valid configurations include those in which: <ul style="list-style-type: none"> ■ XQuery and XPath expressions both return elements. ■ The XQuery and XPath expressions both return attributes—in which case, the XQuery expression must return attributes. To create or edit the XPath expression, click <XPath> (or the XPath_fragment , if one is already defined) to display the XPath Expression Editor .
Variable	The context variable whose contents are evaluated by the XPath variable defined above. Enter the name of the variable in the text field.

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.11 Java Callout Action Properties

In a message flow, use a Java callout action to invoke a Java method or an EJB business service from within the message flow.

In the [Message Flow Editor](#), click a Java Callout action to display its properties in the Properties view. Use these properties pages to configure the selected Java callout action. The pages are:

- Java Callout
- Comment
- Namespaces
- Variables

The Java Callout page has the following options:

Table 4–50 Java Callout Action Options

Option	Description
Method	<p>An external Java method to be called from the message flow.</p> <p>Click Browse to select a class and a static method from an archived resource. Once you have selected the class and method, a table appears on the Java Callout Properties page:</p> <p>The Name column lists all the method’s arguments.</p> <p>The Action column provides an <Expression> or expression_ fragment link to the XQuery/XSLT Expression Editor, where you can create an expression to retrieve a value for the argument.</p> <p>Data types for the arguments must be one of the following:</p> <ul style="list-style-type: none"> ■ <code>java.lang.String</code> ■ Primitive types, and their corresponding class types (e.g., <code>int</code> vs. <code>java.lang.Integer</code>) ■ <code>java.lang.BigDecimal</code>, and <code>java.lang.BigInteger</code> (these types are used in financial calculations where round-off errors or overflows are not tolerable) ■ only <code>org.apache.xmlbeans.XmlObject</code> and no typed XML beans. ■ <code>byte[]</code> ■ <code>java.lang.String[]</code> (INPUT ONLY) ■ <code>XmlObject []</code> (INPUT ONLY) <p>If the type of the input value you enter does not match the declared input argument type, Oracle Service Bus tries to automatically typecast input values to the declared type of the input argument. For example a string value of "123" will be converted to integer 123 if the declared type of the input argument is java primitive <code>int</code>.</p>
Result Type	<p>The variable to which the result is assigned. The label for the field indicates the data type of the result.</p> <p>If the result is a byte array (the only possible array returned), the binary-content XML element is returned.</p>
Service Account	<p>An optional Service Account, which can be specified if there is a security context for this Java method.</p> <p>Click Browse to select a service account.</p> <p>In the case of fixed and mapped service accounts, the <code>userid/password</code> from the service account is authenticated in the local system and the security context propagated to the Java callout. In the case of <code>passthru</code>, the security context is propagated to the Java callout. This context is the message level context if defined (with WS-Security). Otherwise, it is the transport level context.</p>

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.12 Log Action Properties

In a message flow, use the log action to construct a message to be logged and to define a set of attributes with which it will be logged.

In the [Message Flow Editor](#), click a log action to display its properties in the Properties view. Use these properties pages to configure the selected log action. The pages are:

- Log
- Comment
- Namespaces
- Variables

The Log page has the following options:

Table 4–51 Log Action Options

Option	Description
Expression	The message context to be logged through XQuery expressions on context variables. To create or edit an expression, click <Expression> (or the <i>expression fragment</i> , if one is already defined) to display the XQuery/XSLT Expression Editor .
Annotation	Notes for this log action. These notes are logged along with the result of the previously defined expression.
Severity	The severity of the log message. Options are: <ul style="list-style-type: none"> ■ Debug - While your application is under development, you might find it useful to create and use messages that provide verbose descriptions of low-level activity within the application. ■ Info -Used for reporting normal operations; a low-level informational message. ■ Warning - A suspicious operation or configuration has occurred but it might not affect normal operation. ■ Error - A user error has occurred. The system or application can handle the error with no interruption and limited degradation of service.

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.13 Message Flow Properties - Comment

In the [Message Flow Editor](#), click a node or an action to display its properties in the Properties view. For most nodes and actions, one of the properties pages displayed (as a tab) is the Comment page. Use this page to add an optional comment about the item selected in the message flow.

4.11.14 Message Flow Properties - Namespaces

In the [Message Flow Editor](#), click a node or an action to display its properties in the Properties view. For most nodes and actions, one of the properties pages displayed (as a tab) is the Namespaces page. Use this page to see a list of defined namespaces or to create a new namespace.

4.11.15 Message Flow Properties - Variables

In the [Message Flow Editor](#), click a node or an action to display its properties in the Properties view. For many nodes and actions, one of the properties pages displayed (as a tab) is the Variables page. Use this page to see a list of all the predefined and user-defined context variables. Click **Add** to define a new variable.

4.11.16 MFL Transform Action Properties

In a message flow, use a MFL (Message Format Language) transform action to convert message content from XML to non-XML, or vice versa, in the message pipeline. An MFL is a specialized XML document used to describe the layout of binary data. It is an Oracle proprietary language used to define rules to transform formatted binary data into XML data, or vice versa.

In the [Message Flow Editor](#), click a MFL transform action to display its properties in the Properties view. Use these properties pages to configure the selected MFL transform action. The pages are:

- MFL Transform
- Comment
- Namespaces
- Variables

The MFL Transform page has the following options:

Table 4–52 MFL Transform Action Options

Option	Description
Apply Transform	The type of transform to be applied. Select XML to Non-XML or Non-XML to XML , according to your requirement.
on <Expression>	The variable on which the MFL transformation action is to be performed. Click <Expression> (or the <i>expression fragment</i> , if one is already defined) to display the XQuery/XSLT Expression Editor , where you can specify the variable. This input must be text or binary when transforming to XML, and must be XML when transforming to non-XML. Binary content in the message context is represented by the binary-content XML element. This XML should be the result of the Xquery expression when the input needs to be binary.
Resource	Select this option to perform an MFL transform action using a static MFL resource. Click Browse to select the resource.
Resource From	Select this option to specify an MFL resource that will perform the transform action. Click <Expression> (or the <i>expression fragment</i> , if one is already defined) to display the XQuery/XSLT Expression Editor , where you can specify the MFL resource that will perform the action, in the format project/folder/MFLresourcename.
Variable	The name of the variable to which the result of this transform action is to be assigned. The result will be a binary-content XML element.

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.17 Operational Branch Node Properties

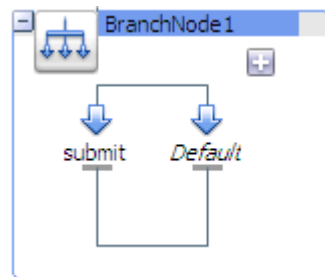
In a message flow, use an operational branch node to configure branching based on operations defined in a WSDL.

When message flows define WSDL-based proxy services, operation-specific processing is required. Instead of configuring a branching node based on operations manually, Oracle Service Bus provides a branching node that automatically branches based on operations. In other words, when you create an operational branch node in a message flow, you can quickly build your branching logic based on the operations defined in the WSDL, because the Oracle Service Bus plug-ins present those operations in the operational branch node configuration page.

A branch node allows processing to proceed along exactly one of several possible paths. Branching is driven by an XPath-based switch table. Each branch in the table specifies a condition (for example, `<500`) that is evaluated in order down the message flow against a single XPath expression (for example, `./ns:PurchaseOrder/ns:totalCost on $body`). Whichever condition is satisfied first determines which branch is followed. If no branch condition is satisfied, then the default branch is followed. A branch node may have several descendants in the message flow: one for each branch, including the default branch.

When you add an operational branch node to a message flow in the [Message Flow Editor](#), the node contains an initial conditional branches based on the first operation defined in the WSDL, plus a default branch, as shown in [Figure 4-1](#).

Figure 4-1 Branch Node With "Submit" Operation From WSDL and "Default" Branch



Click the node itself (the **Operational Branch Node** icon or the bounding box connected to the icon) to display the Flow properties page in the Properties view. You can provide a name and a description for the node on that page.

Click an operational branch icon to display the properties for that operational branch on the Operational Branch page. That page contains a list of all the operations defined in the WSDL. You can select a different operation for that branch. You can also click the plus sign in the operational branch node, in the Message Flow Editor, to add another operation branch.

After you have added all the branches, add nodes and stages to them to define the processing for each branch.

4.11.18 Pipeline Pair Node Properties

In a message flow, use a pipeline pair node to define request and response processing.

Message flows can include zero or more pipeline pair nodes: request and response pipelines for the proxy service (or for the operations on the service), and error handler pipelines that can be defined for stages, pipelines, and proxy services. Pipelines can include one or more stages, which in turn include actions.

A pipeline pair always contains a request pipeline and a response pipeline. Add stages and actions to those pipelines, as needed.

In the [Message Flow Editor](#), click a pipeline pair node to display its properties in the Properties view. Use this properties page to configure the selected pipeline pair node. The options are:

Table 4–53 Pipeline Pair Node Options

Option	Description
Name	Enter a name for the pipeline pair node.
Description	Enter a description for the pipeline pair node.

4.11.19 Publish Action Properties

In a message flow, use a publish action to identify a statically specified target service for a message and to configure how the message is packaged and sent to that service.

In the [Message Flow Editor](#), click a publish action to display its properties in the Properties view. Use these properties pages to configure the selected publish action. The pages are:

- Publish
- Comment
- Namespaces
- Variables

The Publish page has the following options:

Table 4–54 Publish Action Options

Option	Description
Service	The target service for the publish action. Click Browse to select a proxy service or business service from a list.
Invoking	The operation to be invoked on the target service. This option appears only if the selected service defines any operations. To configure how the message is packaged and sent to the service, in the Request Actions field, click Add an Action . Then select an action to associate with the service. You can add more than one action.
Use inbound operation for outbound	Select this option to make the outbound operation the same as the inbound operation. This option appears only if the selected service defines any operations.

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

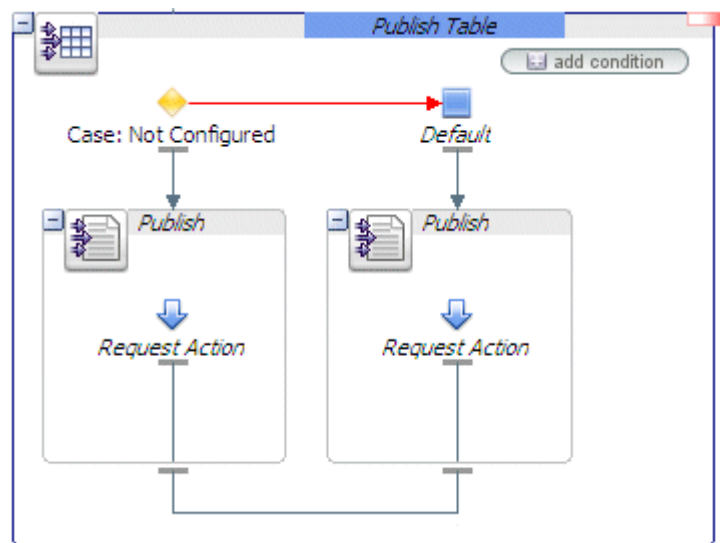
Use the Variables page to see a list of defined context variables or to create a new one.

4.11.20 Publish Table Action Properties

In a message flow, use a publish table action to publish a message to zero or more statically specified services. Switch-style condition logic is used to determine at run time which services will be used for the publish. A publish table action contains one or more case actions to define conditions for following alternative paths (by default, via publish actions), as well as a default path to a default publish action. To fully define a publish table action, you must configure the publish table action, its case actions, and its publish actions (and any other actions you may add).

When you add a publish table action to a message flow in the [Message Flow Editor](#), the publish table action contains a case action with a path to a publish action plus a default case action with a path to a default publish action, as shown in [Figure 4–2](#). Click **add condition** to add another condition to the publish table.

Figure 4–2 Publish Table Action



4.11.20.1 Publish Table Properties

In the Message Flow Editor, click the publish table action itself (the **Publish Table** icon or the bounding box connected to the icon) to display the Publish Tables properties page in the Properties view. Use these properties pages to configure the selected publish table action. The pages are:

- Publish Table
- Comment
- Namespaces
- Variables

The Publish Table properties page has the following option:

Table 4–55 Publish Table Options

Option	Description
Expression	An XQuery expression, which at run time returns the value upon which the routing decision will be made. To create or edit an XQuery expression, click <Expression> (or the expression_fragment , if one is already defined) to display the XQuery/XSLT Expression Editor .

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.20.2 Case Action Properties

In the Message Flow Editor, click a case action to display its properties in the Properties view. Use this properties page to configure the selected case action, as described below:

Table 4–56 Case Action Options

Option	Description
Operator and Value	Select a comparison operator from the list. Then enter a value against which the value returned from the XQuery expression defined for the publish table action will be evaluate

4.11.20.3 Publish Action Properties

See [Section 4.11.19, "Publish Action Properties."](#)

4.11.21 Raise Error Action Properties

In a message flow, use the raise error action to raise an exception with a specified error code (a string) and description.

In the [Message Flow Editor](#), click a raise error action to display its properties in the Properties view. Use these properties pages to configure the selected raise error action. The pages are:

- Raise Error
- Comment
- Namespaces
- Variables

The Raise Error page has the following options:

Table 4–57 Raise Error Action Options

Option	Description
Code	The error code to be raised.
Message	Description of the error code

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.21.1 Transactions

If a service is transactional, a triggered Raise Error action aborts the transaction in the request (asynchronous) or in either the request or response (synchronous). For example, you may introspect messages and determine conditions under which a Raise Error action should occur even if no SOAP fault occurs, and Raise Error causes the transaction to be aborted.

4.11.22 Rename Action Properties

In a message flow, use the rename action to rename elements selected by an XPath expression without modifying the contents of the element.

In the [Message Flow Editor](#), click a rename action to display its properties in the Properties view. Use these properties pages to configure the selected rename action. The pages are:

- Rename
- Comment
- Namespaces
- Variables

The Rename page has the following options:

Table 4–58 *Rename Action Options*

Option	Description
XPath	An XPath expression used to specify the data (in the named variable) that will be renamed. To create or edit the XPath expression, click <XPath> (or the <i>XPath fragment</i> , if one is already defined) to display the XPath Expression Editor .
In Variable	The context variable that holds the element you want to rename. Enter the name of the variable in this field.
Localname	A local name to use to rename the selected elements. Enter the local name in this field.
Namespace	A namespace to use when renaming the selected elements. Enter the namespace in this field.

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.23 Replace Action Properties

In a message flow, use a replace action to replace a node or the contents of a node specified by an XPath expression. The node or its contents are replaced with the value returned by an XQuery expression.

A replace action can be used to replace simple values, elements and even attributes. An XQuery expression that returns nothing is equivalent to deleting the identified

nodes or making them empty, depending upon whether the action is replacing entire nodes or just node contents.

In the [Message Flow Editor](#), click a replace action to display its properties in the Properties view. Use these properties pages to configure the selected replace action. The pages are:

- Replace
- Comment
- Namespaces
- Variables

The Replace page has the following options:

Table 4–59 Replace Action Options

Option	Description
XPath	The XPath expression used to specify the data (in the named variable) that will be replaced. To create or edit the XPath expression, click <XPath> (or the XPath fragment , if one is already defined) to display the XPath Expression Editor .
Variable	Enter a context variable.
Expression	The XQuery expression used to create the data that replaces the data specified by the XPath in the named variable. To create or edit an expression, click <Expression> (or the expression fragment , if one is already defined) to display the XQuery/XSLT Expression Editor .
Replace entire node or Replace node contents	When you finish editing the XQuery expression, select one of the options: <ul style="list-style-type: none"> ■ Replace entire node—to specify that the nodes selected by the XPath expression you defined are replaced along with all of its contents ■ Replace node contents—to specify that the node is not replaced; only the contents are replaced. <p>Note: Selecting the Replace node contents option and leaving the XPath field blank is more efficient than selecting the Replace entire node option and setting the XPath to <code>./*</code></p>

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.24 Reply Action Properties

In a message flow, use the reply action to specify that an immediate reply be sent to the invoker.

The reply action can be used in the request, response or error pipeline. You can configure it to result in a reply with success or failure. In the case of reply with failure where the inbound transport is HTTP, the reply action specifies that an immediate reply is sent to the invoker.

In the [Message Flow Editor](#), click a reply action to display its properties in the Properties view. Use these properties pages to configure the selected reply action. The pages are:

- Assign
- Comment
- Namespaces
- Variables

The Reply page has the following options:

Table 4–60 Reply Action Options

Option	Description
With Success or With Failure	Select With Success to reply that the message was successful, or select With Failure to reply that the message has a fault. Reply With Failure will cause a transaction, if started by Oracle Service Bus, to be aborted.

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.25 Report Action Properties

In a message flow, use the report action to enable message reporting for a proxy service.

In the [Message Flow Editor](#), click a report action to display its properties in the Properties view. Use these properties pages to configure the selected report action. The pages are:

- Report
- Comment
- Namespaces
- Variables

The Report page has the following option:

Table 4–61 Report Action Options

Option	Description
Expression	The XQuery expression used to create the data that will be reported. To create or edit an expression, click <Expression> (or the expression fragment , if one is already defined) to display the XQuery/XSLT Expression Editor .

Table 4–61 (Cont.) Report Action Options

Option	Description
Search Keys	<p>When you finish editing the XQuery expression, click Add a Key to add one or more key value pairs to be used to extract key identifiers from any message context variable or message payload. (The rest of the message is ignored.) The keys are a convenient way to identify a message.</p> <p>In the Key Name: Name field enter a name for the key. In the Key Value column, click <XPath> to create the XPath expression in the XPath Expression Editor. In the In variable field, enter the name of the variable on which the expression will be executed.</p>

For example, consider a report action configured on an error handler in a stage. The action reports the contents of the fault context variable in the event of an error. The report action is configured as follows:

- Key name = `errorCode`
- Key value = `./ctx:errorCode` in variable `fault`

Each time this action is executed at run time, a message is reported via the Reporting Data Stream. The following table shows the results after the report action is executed twice.

Table 4–62 Report Action Execution Results

Report Index	DB TimeStamp	Inbound Service	Error Code
errorCode=BEA-382505	04/26/07 9:45 AM	MortgageBroker/ProxySvc/loanGateway3	BEA-382505
errorCode=BEA-382505	04/26/07 9:45 AM	same	BEA-382505

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.26 Resume Action Properties

In a message flow, use the resume action to resume message flow after an error is handled by an error handler. This action has no parameters and can only be used in error pipelines.

In the [Message Flow Editor](#), click a resume action to display its properties in the Properties view. Use these properties pages to configure the selected resume action. The pages are:

- Comment
- Namespaces
- Variables

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.27 Route Node Properties

In a message flow, use the route node to handle request and response dispatching of messages to and from business services. No other nodes can follow a route node.

In the [Message Flow Editor](#), click a route node to display its properties in the Properties view. Use these properties pages to configure the selected route node. The pages are:

- Route
- Namespaces
- Variables

The Route page has the following option:

Table 4–63 *Route Node Options*

Option	Description
Name	Enter a name for the route node.
Description	Enter a description for the route node.

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.28 Routing Action Properties

In a message flow, use a routing action to identify a target service for the message and configure how the message is routed to that service.

This is a terminal action, which means you cannot add another action after this one. However, this action can contain request and response actions.

In the [Message Flow Editor](#), click a routing action to display its properties in the Properties view. Use these properties pages to configure the selected routing action. The pages are:

- Routing
- Comment
- Namespaces
- Variables

The Routing page has the following option:

Table 4–64 *Routing Action Options*

Option	Description
Service	The target service for the routing action. Click Browse to select a proxy service or business service from a list.

Table 4–64 (Cont.) Routing Action Options

Option	Description
Invoking	<p>The operation to be invoked on the target service.</p> <p>This option appears only if the selected service defines any operations.</p> <p>To configure how the message is packaged and sent to the service, in the Request Actions field, click Add an Action. Then select an action to associate with the service. You can add more than one action.</p>
Use inbound operation for outbound	<p>Select this option to make the outbound operation the same as the inbound operation.</p> <p>This option appears only if the selected service defines any operations.</p>

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.29 Routing Options Action Properties

In a message flow, use a routing options action to modify any or all of the following properties in the outbound request: URI, Quality of Service, Mode, Retry parameters, Message Priority.

In the [Message Flow Editor](#), click a routing options action to display its properties in the Properties view. Use these properties pages to configure the selected routing options action. The pages are:

- Routing Options
- Comment
- Namespaces
- Variables

The Routing Options page has the following option:

Table 4–65 Routing Options Action Properties

Option	Description
URI	<p>Select this option to set the URI for the outbound message:</p> <p>To create or edit an expression, click <Expression> (or the expression fragment, if one is already defined) to display the XQuery/XSLT Expression Editor.</p> <p>Enter an expression that returns a URI. This overrides the URI for the invoked service.</p>
QoS	<p>Select this option to set the Quality of Service element:</p> <p>Select a QoS option from the list. This overrides the default.</p>
Mode	<p>Select this option to set the mode to Request or Request-Response.</p> <p>Select the mode from the list.</p> <p>Note: This is normally already automatically set, based on the interface of the service invoked. However, in some cases like Any Soap or Any XML services, this is not so.</p>

Table 4–65 (Cont.) Routing Options Action Properties

Option	Description
Retry Interval	Select this option to set the retry interval. Specify the number of seconds between retries. This overrides the default configured with the invoked service.
Retry Count	Select this option to set the retry count. Specify the number of retries the system must attempt before discontinuing the action. This overrides the default configured with the invoked service.
Priority	Select this option to set the message priority: To create or edit an expression, click <Expression> (or the <i>expression_fragment</i> , if one is already defined) to display the XQuery/XSLT Expression Editor . Enter an expression that returns a positive integer.

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

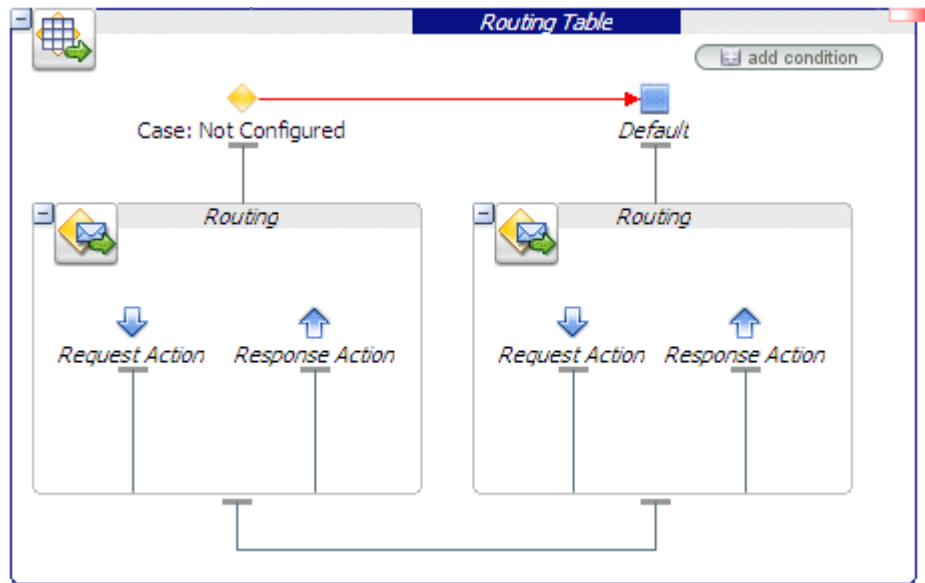
4.11.30 Routing Table Action Properties

In a message flow, use a routing table to select different routes based upon the results of a single XQuery expression. A routing table action contains a set of routes wrapped in a switch-style condition table.

This is a terminal action, which means you cannot add another action after this one. However, this action can contain request and response actions.

When you add a routing table action to a message flow in the [Message Flow Editor](#), the routing table action contains a case action with a path to a routing action plus a default case action with a path to a default routing action, as shown in [Figure 4–3](#). Click **add condition** to add another condition to the routing table.

Figure 4–3 Routing Table Action



4.11.30.1 Routing Table Properties

In the Message Flow Editor, click the publish table action itself (the **Publish Table** icon or the bounding box connected to the icon) to display the Publish Tables properties page in the Properties view. Use these properties pages to configure the selected publish table action. The pages are:

- Routing Table
- Comment
- Namespaces
- Variables

The Routing Table properties page has the following option:

Table 4–66 Routing Table Options

Option	Description
Expression	An XQuery expression, which at run time returns the value upon which the routing decision will be made. To create or edit an XQuery expression, click <Expression> (or the expression_fragment , if one is already defined) to display the XQuery/XSLT Expression Editor .

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.30.2 Case Action Properties

In the Message Flow Editor, click a case action to display its properties in the Properties view. Use this properties page to configure the selected case action, as described below:

Table 4–67 Case Action Options

Option	Description
Operator and Value	Select a comparison operator from the list. Then enter a value against which the value returned from the XQuery expression defined for the routing table action will be evaluate

4.11.30.3 Routing Action Properties

See [Section 4.11.28, "Routing Action Properties."](#)

4.11.31 Service Callout Action Properties

In a message flow, use a service callout action to configure a synchronous (blocking) callout to an Oracle Service Bus-registered proxy or business service.

In the [Message Flow Editor](#), click a service callout action to display its properties in the Properties view. Use these properties pages to configure the selected service callout action. The pages are:

- Service Callout
- Comment
- Namespaces
- Variables

The Service Callout page has the following options:

Table 4–68 Service Callout Action Options

Option	Description
Service	The target service for the service callout action. Click Browse to select a proxy service or business service from a list.
Invoking	The operation to be invoked on the target service. This option appears only if the selected service is WSDL-based and has operations that can be invoked on the service.
Configure Soap Body or Configure Payload Document	Specify how you want to configure the request and response messages by selecting one of the following options: <ul style="list-style-type: none"> ■ Select Configure SOAP Body to configure the SOAP Body. Selecting this option allows you to use \$body directly. This option supports SOAP-RPC encoded, which is not supported when configuring payload parameters or document. ■ Select Configure Payload Parameters or Configure Payload Document to configure the payload.

Subsequent configuration options depend on the kind of service you selected and on the kind of configuration options you chose. [Table 4–69](#) shows the options available for each service type.

Table 4–69 SOAP Body, Payload Parameters, and Payload Document Options

Selected Service Type	"Configure SOAP Body" Options	"Configure Payload Parameters" Options or "Configure Payload Document" Options
SOAP RPC	See "SOAP Request Body and SOAP Response Body" in the following table. (Optional) See "SOAP Request Header and SOAP Response Header" in the following table.	See "Request Parameters and Response Parameters" (optional) (Optional) See "SOAP Request Header and SOAP Response Header" in the following table.
SOAP Document and Any SOAP	See "SOAP Request Body and SOAP Response Body" in the following table. (Optional) See "SOAP Request Header and SOAP Response Header" in the following table.	See "Request Document and Response Document" in the following table. (Optional) See "SOAP Request Header and SOAP Response Header" in the following table.
XML, Any XML, and Messaging	See "SOAP Request Body and SOAP Response Body" in the following table.	See "Request Document and Response Document" in the following table.

The following table provides instructions for each of the options listed in the previous table.

Table 4–70 SOAP Body, Payload Parameters, and Payload Document Option Descriptions

For These Options...	Follow These Steps...
SOAP Request Body and SOAP Response Body	To configure these options: <ul style="list-style-type: none"> ■ In the SOAP Request Body field, enter the name of a variable to hold the XML of the SOAP Body element for the callout request. ■ In the SOAP Response Body field, enter the name of a variable to which the XML of the SOAP Body element on the response will be bound.
SOAP Request Header and SOAP Response Header	To configure these options: <ul style="list-style-type: none"> ■ In the SOAP Request Header field, enter the name of a variable to hold the XML of the SOAP Header element for the callout request You must wrap the input document for the SOAP Request Header with <code><soap-env:Header> . . . </soap-env:Header></code>. ■ In the SOAP Response Header field, enter the name of a variable to which the XML of the SOAP Headers on the response, if any, will be bound.

Table 4–70 (Cont.) SOAP Body, Payload Parameters, and Payload Document Option Descriptions

For These Options...	Follow These Steps...
Request Parameters and Response Parameters	<p>To configure options:</p> <ul style="list-style-type: none"> ■ In the Request Parameters fields, enter names for the variables that will be evaluated at run time to provide values for the request parameters. <p>You must provide only the core payload documents in the input variable—the SOAP package is created for you by Oracle Service Bus. In other words, do not wrap the input document with <code><soap-env:Body> . . . </soap-env:Body></code>.</p> <p>For example, when creating a body input variable that is used for this request parameter, you would define that variable's contents using the XPath statement <code>body/*</code> (to remove the wrapper <code>soap-env:Body</code>), not <code>\$body</code> (which results in keeping the <code>soap-env:Body</code> wrapper).</p> ■ In the Response Parameters fields, enter the names of the variables to which the responses will be assigned at run time.
Request Document and Response Document	<p>To configure these options:</p> <ul style="list-style-type: none"> ■ In the Request Document Variable field, enter the name of a variable to assign a request document to. <p>For <i>SOAP Document-type</i> services, the variable is evaluated at runtime to form the body of the SOAP message sent to the service. For <i>Any XML</i> services, the variable is evaluated at runtime to form the body of the XML message sent to the service.</p> <p>For SOAP Document-type services and for Any XML services, you provide only the core payload documents in the input variable—the SOAP package is created for you by Oracle Service Bus. In other words, do not wrap the input document with <code><soap-env:Body> . . . </soap-env:Body></code>.</p> <p>For example, when creating a body input variable that is used for this request parameter, you would define that variable's contents using the XPath statement <code>body/*</code> (to remove the wrapper <code>soap-env:Body</code>), not <code>\$body</code> (which results in keeping the <code>soap-env:Body</code> wrapper).</p> <p>For <i>Messaging</i> services, the variable is evaluated to form the body of the message, based on the type of data expected by the service. The following restrictions apply to variables used with Messaging services:</p> <ul style="list-style-type: none"> ■ For services that expect binary data, the variables must have a <code>ctx:binary-content</code> element. ■ For services that expect MFL data, the variable must have the XML equivalent. ■ For services that expect text data, the variable is a string. ■ In the Response Document Variable field, enter the name of the variable to which a response document will be assigned at run time.

Optionally, add one or more transport header actions. For more information about transport header actions, see [Section 4.11.34, "Transport Headers Action Properties."](#)

Note: In addition to the transport headers you specify, headers are added by the Oracle Service Bus binding layer.

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

4.11.32 Skip Action Properties

In a message flow, use the skip action to specify that at run time, the execution of the current stage is skipped and the processing proceeds to the next stage in the message flow. This action has no parameters and can be used in the request, response or error pipelines.

In the [Message Flow Editor](#), click a skip action to display its properties in the Properties view. Use these properties pages to configure the selected skip action. The pages are:

- Comment
- Namespaces
- Variables

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.33 Stage Node Properties

In a message flow, use a stage node as a container for actions in a message flow. You can string multiple stages together, to compartmentalize processing logic.

In the [Message Flow Editor](#), click a stage node to display its properties in the Properties view. Use these properties pages to configure the selected stage node. The pages are:

- Stage
- Namespaces
- Variables

The Stage page has the following options:

Table 4–71 Start Node Options

Option	Description
Name	Enter a name for the stage node.
Description	Enter a description of the stage node.

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.34 Transport Headers Action Properties

In a message flow, use a transport header action to set header values in messages.

In the [Message Flow Editor](#), click a transport headers action to display its properties in the Properties view. Use these properties pages to configure the selected transport headers action. The pages are:

- Transport Headers
- Comment
- Namespaces
- Variables

The Transport Header page has the following options:

Table 4–72 Transport Header Action Options

Option	Description
Direction	<p>From the Set Transport Headers for list, select one of the following, to specify to the run time which of the message context locations are to be modified:</p> <ul style="list-style-type: none"> ■ Outbound Request - Select this option to set header values for outbound requests (the messages sent out by a proxy service in route, publish, or service callout actions). This header element is located in the message context as follows: <code>\$outbound/ctx:transport/ctx:request/tp:headers</code> ■ Inbound Response - Select this option to set header values for inbound responses (the response messages a proxy service sends back to clients). This header element is located in the message context as follows: <code>\$inbound/ctx:transport/ctx:response/tp:headers</code>
Pass All Headers	Select this option to pass all headers through from the inbound message to the outbound message or vice versa. Every header in the source set of headers will be copied to the target header set, overwriting any existing values in the target header set.
Headers	<p>Click Add Header to add a header to the Headers table. Then configure each header as described in the following rows.</p> <p>You can add as many headers as necessary to this table. You do not have to order the headers in the table, because the run time declares namespaces and places header elements in their proper order when generating the corresponding XML.</p>
Headers: Name:	<p>Specify a header by doing either of the following:</p> <ul style="list-style-type: none"> ■ From the list in the Name column, select a header name. The list contains all of the predefined header names for the target transport (for example, Content-Type for HTTP transports, JMSCorrelationID for JMS transports, etc.). ■ Enter a header name in the Other field. If that header name is not one of the predefined headers for this service’s transport, it becomes a user-header, as defined by the transport specification.

Table 4–72 (Cont.) Transport Header Action Options

Option	Description
Headers: Action	<p>Select one of the options in this column to specify how to set the header's value:</p> <ul style="list-style-type: none"> <li data-bbox="651 327 1360 646"> <p>■ Set Header to Expression</p> <p>Selecting this option allows you to use an XQuery or XSLT expression to set the value of the header. The expression can be simple (for example, "text/xml") or a complex XQuery or XSLT expression.</p> <p>Because the Oracle Service Bus transport layer defines the XML representation of all headers as string values, the result of any expression is converted to a string before the header value is set. Expressions that return nothing result in the header value being set to the empty string. You cannot delete a header using an expression.</p> <p>Caution: Not all of the header settings you can specify in this action are honored at run time.</p> <li data-bbox="651 722 1360 1331"> <p>■ Delete Header</p> <p>Specifies that the header is removed from the request or response metadata.</p> <p>Copy Header from Inbound Request (if you are setting transport headers for the Outbound Request)</p> <p>or</p> <p>Copy Header from Outbound Response (if you are setting transport headers for the Inbound Response)</p> <p>Specifies that this header is copied directly from the corresponding header of the same name from the inbound message to the outbound message and vice versa. For example, if you want to set the SOAPAction header for an outbound request, selecting Copy Header from Inbound Request causes the run time to copy the value from the SOAPAction request header of <code>\$inbound</code>. In the case of inbound response headers, the source of the header to copy is the response headers of <code>\$outbound</code>.</p> <p>If the Copy Header option is selected for a header that does not exist in the source, this option is ignored and no action is performed on the target for this header.</p>

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.11.35 Validate Action Properties

In a message flow, use a validate action to validate elements selected by an XPath expression against an XML schema element or a WSDL resource. You can validate global elements only; Oracle Service Bus does not support validation against local elements.

In the [Message Flow Editor](#), click a validate action to display its properties in the Properties view. Use these properties pages to configure the selected validate action. The pages are:

- Validate

- Comment
- Namespaces
- Variables

The Validate page has the following options:

Table 4–73 Validate Action Options

Option	Description
XPath	An XPath expression that specifies the elements to be validated. To create or edit the XPath expression, click <XPath> (or the XPath_fragment , if one is already defined) to display the XPath Expression Editor .
In Variable	The name of the variable to hold the element to be validated. Enter the name of the variable, or, if Content Assist is on, press Ctrl-Space to display a list of declared variables.
Against Resource	An XML schema element or a WSDL resource against which the elements selected by the XPath expression (in the XPath field described above) are validated.
Save Variable or Raise Error	To save the result of this validation (a boolean result), select Save Variable and enter the name of the variable in which you want to save the result. Alternatively, to raise an error if the element fails validation against the WSDL or XML schema element, select Raise Error .

Use the Comment page to add a comment, if desired:

Use the Namespaces page to see a list of defined namespaces or to create a new one.

Use the Variables page to see a list of defined context variables or to create a new one.

4.12 Modify JAR Dependencies Dialog

In Oracle Service Bus, JAR resources can contain classes that depend on other classes in different JARs. Use the Modify JAR Dependencies page to add or remove referenced JAR resources. Use the **Up** and **Down** buttons to reorder the hierarchy (order of precedence) of referenced JARs. For more information, see "JARs" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* at <http://www.oracle.com/pls/as1111/lookup?id=OSBAG898>.

Table 4–74 Modify JAR Dependencies Properties

Option	Description
Available JARs	The project name and the name of the folder, if applicable, in which the JAR resources reside. This column shows all available JARs in the current workspace.
JAR references	The name and path of the referenced JAR resources. This column shows all the JARs currently configured as dependencies.

4.13 SMTP Servers

The following editor and wizard are provided for working with SMTP servers:

- [Section 4.13.1, "Edit SMTP Server Page"](#)
- [Section 4.13.2, "New SMTP Server Resource Wizard"](#)

4.13.1 Edit SMTP Server Page

Use this page to view the details of an SMTP server and edit the configuration, if required. For descriptions of the fields, see the [Section 4.13.2, "New SMTP Server Resource Wizard."](#)

If you rename an SMTP resource, the new name is automatically updated in the any e-mail business services that reference the SMTP resource in the endpoint URI.

4.13.2 New SMTP Server Resource Wizard

Use this page to configure a new SMTP server resource.

Table 4–75 New SMTP Server Options

Option	Description
Description	Enter a description for this SMTP server resource.
Server URL	Enter the URL that points to the SMTP server. This is a required field.
Port Number	Enter a port number for the SMTP server (the default port is 25). This is a required field.
User Name	If access to the target SMTP server requires a user name and password, enter a user name in the User Name field, and the associated password in the Password and Confirm Password fields. These fields are optional, and required only if the SMTP server is secured.
Password	Enter the associated password.
Confirm Password	Enter the same password you entered for the Password field.

4.14 UDDI Registry Configuration Page

Use this page to configure a UDDI registry resource. This page is displayed in the New UDDI Registry Resource wizard and in the UDDI Configuration editor.

For more information, see "UDDI" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* at

<http://www.oracle.com/pls/as1111/lookup?id=OSBAG392>.

Table 4–76 UDDI Registry Options

Option	Description
Description	Enter a description of the registry.
Inquiry URL	The URL of the Inquiry API endpoint used for locating and importing services. Enter an inquiry URL in the format: <code>http://host:port/APPLICATION_SERVER_CONTEXT/uddi/inquiry</code>
Publish URL	The URL of the Publish API endpoint used for publishing services. Enter a publish URL in the format: <code>http://host:port/APPLICATION_SERVER_CONTEXT/uddi/publishing</code>
Security URL	The URL of the Security API endpoint used for getting an authentication token so that you can publish to the registry. Enter a security URL in the format: <code>http://host:port/APPLICATION_SERVER_CONTEXT/uddi/security</code>

Table 4–76 (Cont.) UDDI Registry Options

Option	Description
Subscription URL	The URL of the Subscription API endpoint used for subscribing to registry changes, creating a registry subscription, and listening for changes to imported services. Enter a subscription URL in the format: <code>http://host:port/APPLICATION_SERVER_CONTEXT/uddi/subscription</code> By default, the value for APPLICATION_SERVER_CONTEXT in the installer is registry.
User Name	Enter the user name to log into the registry console.
Password (Confirm Password)	Enter the password to log into the registry console.
Validate	Click Validate to validate that a connection can be made to the configured registry.

4.15 Outline view - Oracle Service Bus

This view displays a hierarchical view, or outline, of a structured file that is currently open in the editor area. The contents of this view are editor-specific.

In Oracle Service Bus, when the proxy service editor is open and the Message Flow Editor is selected, the Outline view displays a hierarchical view of the nodes and actions in the message flow. When the Message Flow Editor is displayed, you can switch between the hierarchical view and a thumbnail view of the service.

In thumbnail view, when the entire message flow does not fit in the editor, a blue mask appears in the outline view to show what portion of the flow is visible in the editor. To display a different portion of the flow, you can drag the mask to the portion of the flow you want to display in the editor.

4.16 Resource Management

The following are provided for managing resources:

- [Section 4.16.1, "Preferences dialog - Oracle Service Bus - Type Associations Page"](#)
- [Section 4.16.2, "References View"](#)
- [Section 4.16.3, "Select Clone Target Dialog"](#)
- [Section 4.16.3, "Select Clone Target Dialog"](#)

4.16.1 Preferences dialog - Oracle Service Bus - Type Associations Page

Use this page to associate one or more file extensions with an Oracle Service Bus resource type.

Table 4–77 Type Association Properties

Option	Description
Resource Types	This pane lists all the available types of Oracle Service Bus resources. Select a type to display its assigned file extension associated with the type.

4.16.2 References View

Use this page to view resource dependencies. This view shows your current selection whether it is in the Project Explorer or the active editor.

Table 4–78 Reference Dependency Properties

Option	Description
Referenced By	A list of the resources outside of this project, folder, or resource that are dependent on resources inside this project or folder, or this resource.
References	A list of the resources outside of this project, folder, or resource on which resources inside this project or folder, or this resource depends.

4.16.3 Select Clone Target Dialog

Use this dialog to specify where to clone a project, as a new project or as a folder under a different project.

Table 4–79 Clone Project Options

Option	Description
Name	Enter a name for the new (cloned) project.
As project	Select As project to clone the project as another project, that is, as a peer to the other projects.
As folder in location	Select As folder in location to convert a project (and its contents) into a folder under a project. When you select this option, you must also select the new location.

4.16.4 Select a Resource Dialog

Use this dialog to select a resource, appropriate to the context, that has been created in or imported into the Oracle Service Bus plug-ins. When you click **Browse** from an editor or a wizard, this dialog displays the available resources that are appropriate for the context. The dialog has different names, depending on the type of resource displayed, for example, **Select a WSDL**, **Select a MFL**, **Select an XML Schema**, etc. In some cases, the dialog displays a hierarchical list of items contained in each resource. For example, the **Select an XML Schema** dialog lists elements and types defined in the schemas. The **Select a WSDL** dialog displays ports and bindings defined in the WSDLs.

In some cases, for example, the **Select a WSDL** dialog, you can click the **Consume** button to open the **Service Consumption** dialog, where you can consume services from the following resource types: Oracle Enterprise Repository, file system, UDDI, URI, workspace: consume a service residing in the current workspace.

4.17 New Service Key Provider Resource

Use this page to configure a service key provider resource after you create one.

Table 4–80 New Service Key Provider Options

Option	Description
Description	Enter a description for the service key provider.

Table 4–80 (Cont.) New Service Key Provider Options

Option	Description
Encryption Key	<p>To enter an Encryption Key:</p> <ol style="list-style-type: none"> 1. Next to Encryption Key, select the Key check box. 2. Enter a key alias that maps to an X.509 certificate and that supports encryption, or click Browse to display the key aliases from the key store that your realm's PKI credential mapper is using. 3. Enter the password that you use to secure access to the key store. (You set the password when you create the key store.) <p>When you associate this service key provider with a proxy service, Oracle Service Bus embeds the X.509 certificate into the proxy service's WSDL. The proxy service then uses this certificate to encrypt the messages that it sends to its endpoint. The proxy service uses the private key in the PKI credential to decrypt the messages that the endpoint returns.</p>
Digital Signature Key	<p>To enter a Digital Signature Key:</p> <ol style="list-style-type: none"> 1. Next to Digital Signature Key, select the Key check box. 2. Enter a key alias, or click Browse to display the key aliases from the key store that your realm's PKI credential mapper is using. 3. Enter the password that you use to secure access to the key store. (You set the password when you create the key store.)
SSL Client Authentication Key	<p>To enter an SSL Client Authentication Key:</p> <ol style="list-style-type: none"> 1. Next to SSL Client Authentication Key, select the Key check box. 2. Enter a key alias, or click Browse to display the key aliases from the key store that your realm's PKI credential mapper is using. 3. Enter the password that you use to secure access to the key store. (You set the password when you create the key store.)

4.18 New WS-Policy

Web Services Policy Framework (WS-Policy) is an extensible XML-based framework that extends the configuration of a Web Service with domain specific security assertions and specifies the security requirements, expectations, and capabilities of the Web Service. In Oracle Service Bus, one of the primary uses of WS-Policy is configuring message-level security in proxy services and business services.

Use this page to select a name and location for a new WS-Policy resource.

Table 4–81 New WS-Policy Options

Option	Description
Enter or select the parent folder	Enter or select the name of the project or folder in which to locate a new WS-Policy resource.
File Name	Enter a name for this WS-Policy.

4.19 Service Accounts

The following are provided for working with service accounts:

- [Section 4.19.1, "New Service Account Resource"](#)

- [Section 4.19.2, "Service Account Editor - General Configuration Page"](#)
- [Section 4.19.3, "Service Account Editor - Static User Configuration Page"](#)
- [Section 4.19.4, "Service Account Editor - User Mappings Configuration Page"](#)

4.19.1 New Service Account Resource

Use the Service Account Resource editor to configure service account resource after you create it.

Table 4–82 New Service Account Options

Option	Description
Description	Enter a description for the service account.
Resource Type	<ul style="list-style-type: none"> ■ To create a service account that provides the user names and passwords that it receives from incoming client requests, select Pass Through and click Finish. ■ To create a service account that provides a user name and password that you save with the service account configuration, select Static. Continue as described in Section 4.19.3, "Service Account Editor - Static User Configuration Page." ■ To create a service account that maps the user name from one or more authenticated clients to user names and passwords that you specify, select Mapping. Continue as described in Section 4.19.4, "Service Account Editor - User Mappings Configuration Page."

4.19.2 Service Account Editor - General Configuration Page

Use this page to configure a service account resource, as described in the following table.

Table 4–83 Service Account General Options

Option	Description
Description	Enter a description for the service account.
Resource Type	Leave the resource type as is, or select a different resource type: <ul style="list-style-type: none"> ■ A Pass Through type provides the user names and passwords that it receives from incoming client requests. ■ A Static type provides a user name and password that you save with the service account configuration. Configure this type on the Service Account Editor - Static User Configuration Page. ■ A Mapping type maps the user name from one or more authenticated clients to user names and passwords that you specify. Configure this type on the Service Account Editor - User Mappings Configuration Page.

4.19.3 Service Account Editor - Static User Configuration Page

Use this to page to save a user name and password with the service account configuration. The service account encodes this user name and password in the outbound request.

Table 4–84 Service Account Static User Options

Option	Description
User Name	Enter a user name for this service account.
Password	Enter the associated password.
Confirm Password	Enter the same password you entered for the Password field.

4.19.4 Service Account Editor - User Mappings Configuration Page

Use this to page to create a service account that maps the user name from one or more clients to user names and passwords that you specify.

Table 4–85 Service Account User Mappings Options

Option	Description
Remote Users	In the Remote User Name and Remote Password fields, enter the user name and password that you want to send in outbound requests. (Optional) Add additional remote users in the Remote Users table.
Local User Mappings	To map <i>authorized</i> clients to remote user names and passwords, in the Local User Name field, enter the name that identifies a client that has been authenticated on its inbound request. If you have not already added this user in the Security Configuration module of the Oracle Service Bus Console, do so before you use this mapping in a runtime environment. Oracle Service Bus lets you create a mapping for a non-existent local user, but the mapping will never match an authenticated user and will never be used. From the Remote User Name list, select the user name that you want to send in outbound requests for the authenticated user you specified in the Local User Name field.
Map Anonymous Requests	To map <i>anonymous</i> clients to remote user names, select the Map Anonymous Requests check box. From the Select Remote User list, select the user name that you want to send in outbound requests for all anonymous users.

4.20 Expression Editors

The following editors help you to write expressions for use in services.

- [Section 4.20.1, "XQuery/XSLT Expression Editor"](#)
- [Section 4.20.2, "XPath Expression Editor"](#)
- [Section 4.20.3, "Condition Editor"](#)

4.20.1 XQuery/XSLT Expression Editor

Use the XQuery/XSLT Expression editor to create variable structures, define user namespaces, use predefined message context variables to build inline XQuery expressions, build inline XQuery expressions manually, and select XQuery or XSLT resources for execution as inline XQueries.

The XQuery/XSLT Expression editor contains two panels, each containing three tabs.

Use the panel on the left to write or construct the expression. The panel contains these pages (tabs):

- [Section 4.20.5, "Expression Page"](#)
- [Section 4.20.6, "XQuery Resource Page"](#)
- [Section 4.20.7, "XSLT Resource Page"](#)
- [Section 4.20.8, "Dynamic XQuery Page"](#)

Use the tabs in the panel on the right to manage the variables, namespaces, and XQuery functions you can use to construct an XQuery or XSLT. The panel contains these pages (tabs):

- [Section 4.20.9, "Variable Structures Page"](#)
- [Section 4.20.10, "Namespace Definitions Page"](#)
- [Section 4.20.11, "XQuery Functions Page"](#)

4.20.2 XPath Expression Editor

Use the XPath Expression Editor to create an XPath expression, which is then inserted into the location from which you launched the editor.

The XPath Expression Editor contains these pages:

- [Section 4.20.5, "Expression Page"](#)
- [Section 4.20.9, "Variable Structures Page"](#)
- [Section 4.20.10, "Namespace Definitions Page"](#)
- [Section 4.20.11, "XQuery Functions Page"](#)

You can write an expression directly in the **Expression** text field, or you can drag variables from the Variable Structure page and drag XQuery functions from the XQuery Functions page to construct a valid XPath expression.

When you are finished constructing the expression, you can click **Test** to test the expression on a running server. Or click **OK** to insert the expression without testing it.

4.20.3 Condition Editor

Use the Condition Editor to create an inline XQuery condition.

Use the panel on the left to write or construct the condition. The panel contains these tabs, each of which is described below:

- [Section 4.20.5, "Expression Page"](#)
- [Section 4.20.4, "Condition Builder Page"](#)

Use the tabs in the panel on the right to manage the variables, namespaces, and XQuery functions you can use to construct a condition. The panel contains the following tabs. They are the same as the tabs in the right-hand panel of the XQuery/XSLT Expression Editor, and the following links display the help for those tabs in the XQuery/XSLT Expression Editor.

- [Section 4.20.9, "Variable Structures Page"](#)
- [Section 4.20.10, "Namespace Definitions Page"](#)
- [Section 4.20.11, "XQuery Functions Page"](#)

4.20.4 Condition Builder Page

Use the Condition Builder page to build an inline condition. This page appears only in the [Condition Editor](#).

Build an expression in the Condition Builder as follows:

1. Select **Comparison Expression** to build a comparison expression or select **Unary Expression** to build a unary expression.
2. If you chose **Comparison Expression**, do the following:
 - a. In the **Operand** field, enter the name of the operand you want to compare to a value. Alternatively, drag an item from the **Variable Structures** tab on the right to the **Operand** field.
 - b. In the **Value** field, enter the value against which to compare the operand. Alternatively, drag item from the **Variable Structures** tab on the right to the **Value** field.
 - c. Select a comparison operator from the **Operator** list.
 - d. Go to step 4.
3. If you chose **Unary Expression**, do the following:
 - a. Select or deselect the **Not** check box to indicate whether to evaluate the expression as true or not true, that is, to specify whether the expression should be enclosed by not().
 - b. Enter an expression in the text field, or construct it by dragging items from the **Variable Structures** tab on the right.
4. Do either of the following:
 - Click **Add** to add a new statement to the expression in the **Condition Expression** field.
 - Select a statement in the **Condition Expression** field, then click **Update** to modify the statement.
5. Click **Test** to test the expression in a running server, or click **OK** to insert the condition in the message flow.

4.20.5 Expression Page

This page appears on the [XQuery/XSLT Expression Editor](#), the [XPath Expression Editor](#), and the [Condition Editor](#).

The Expression page contains a text field where you can build expressions by typing directly into the field or by dragging items from the [Variable Structures Page](#) or the [XQuery Functions Page](#) on the right side of the editor.

Click **Test** to test the expression in an Oracle Service Bus domain on a running server. Oracle WebLogic Server is packaged with Oracle Service Bus, so an installation of Oracle Service Bus includes an embedded server you can use for testing purposes.

4.20.6 XQuery Resource Page

This page appears in the [XQuery/XSLT Expression Editor](#) only.

In the message flow of a proxy service, you can assign XQuery expressions to message context variables, assign if-then-else actions based on the Boolean result of an XQuery expression, insert the result of an XQuery expression at an identified place relative to

an XPath expression, specify the message context that you want to log through XQuery expressions on context variables, and so on.

Use the XQuery Resource page to configure an XQuery transformation to be executed in the message flow of a proxy service.

Table 4–86 XQuery Resource Options

Field	Description
XQuery	The XQuery resource to be executed. Click Browse to find and open an XQuery resource that has been registered in Oracle Service Bus.
Bind Variables	When you select a resource, each input parameter of the transformation is displayed. Each label corresponds to the name of a parameter, and each text box is for defining an XQuery expression to be mapped to the parameter. You must define a mapping for each parameter. Enter the expression directly, or drag variables and structures from the right panel.

4.20.7 XSLT Resource Page

This page appears in the [XQuery/XSLT Expression Editor](#) only.

Use the **XSLT Resource** tab to configure an XSLT transformation to be executed in the message flow of a proxy service.

Table 4–87 XSLT Resource Options

Field	Description
XSLT	The XSLT resource to execute. Click Browse to find and open a resource registered with Oracle Service Bus.
Input Document	An XQuery expression for the input document to the transformation, for example <code>\$body</code> .
Bind Variables	<p>A label and a corresponding text box is displayed for each input parameter of the transformation. Each label corresponds to the name of a parameter, and each text box is for defining an XQuery expression to be bound to the parameter. You must define a binding for each parameter. For example, if an XSL transformation has two input parameters named one and two, the Variable Name field has two labels one and two, with a text box associated with each into which the XQuery expression is entered.</p> <p>Enter the expression directly, or drag variables and structures from the right panel.</p> <p>The following XQuery expressions are examples of valid input to this field:</p> <pre>\$body/*[1] \$body/po:PurchaseOrder</pre> <p>Note: The following variable name is not a valid entry for this field and results in an exception:</p> <pre>body</pre>

4.20.8 Dynamic XQuery Page

This page appears in the [XQuery/XSLT Expression Editor](#) only.

You can specify a dynamic XQuery expression that evaluates at runtime to the name of a pre-registered XQuery resource. Oracle Service Bus executes this XQuery resource,

with optional variable bindings, against the message context to produce the required transformation.

Use the XQuery Resource page to configure a dynamic XQuery transformation to be executed in the message flow of a proxy service.

Table 4–88 *Dynamic XQuery Options*

Field	Description
Expression	The XQuery expression that will evaluate at runtime to the name of a pre-registered XQuery resource. The following shows the syntax for the XQuery resource (representing the full name of the resource): <code>Project/folder1/folder2/XQueryResourceName</code>
Select XQuery Template	The resource to serve as a template for the shape of the query (the number and names of the variables). Click Browse to select an existing registered resource. After selecting a template, the variables appear in the Bind Variables area. Note that the template is not persisted with the configuration. Instead, the template serves as a quick start to help you specify the variables for the query.
Add Custom Variable	An input parameter of the transformation. Type a variable name in the Add Custom Variable field, and click Add .
Bind Variables	When you add a custom variable, it appears in the Bind Variables area. Similarly, when you select an XQuery template, each input parameter of the transformation is displayed. Each label corresponds to the name of a parameter, and each text box is for defining an XQuery expression to be mapped to the parameter. You must define a mapping for each parameter. Enter the expression directly, or drag variables and structures from the right panel.

4.20.9 Variable Structures Page

This page appears in the [XQuery/XSLT Expression Editor](#), the [XPath Expression Editor](#), and the [Condition Editor](#).

The Variable Structures page displays variables and their contents as trees. It includes the built-in message context variables `attachments`, `body`, `header`, `outbound`, and `inbound`, as well as any user-defined variables. It includes `fault` if the context of the expression is appropriate (that is, in an error handler). The `outbound` variable is always listed; even though it is not valid in every context. If `outbound` is used in invalid contexts, it will be reported when validating.

Each variable structure mapping entry has a label and maps a variable or variable path to one or more structures. The scope of these mappings is a stage or a route node.

You can drag items from this page to the editor's [Expression Page](#) to insert them into the current expression. They are inserted as XPath expressions.

To define a new variable and add it to the variable structure tree, click **Add** to open the [Add Variable Structure Dialog](#).

4.20.10 Namespace Definitions Page

This page appears in the [XQuery/XSLT Expression Editor](#), the [XPath Expression Editor](#), and the [Condition Editor](#).

The Namespace Definitions page lists default Oracle Service Bus namespaces, variable namespaces, and user-defined namespaces. Click **Add** to define a new namespace,

which is then added to the list of user-defined namespaces. To modify a user-defined namespace, select it in the list, then click **Edit**.

4.20.11 XQuery Functions Page

This page appears in the [XQuery/XSLT Expression Editor](#), the [XPath Expression Editor](#), and the [Condition Editor](#).

The XQuery Functions page lists a set of standard XQuery functions, organized alphabetically and by type. You can drag functions from this page to the editor's [Section 4.20.5, "Expression Page"](#) to insert them into the current expression. When you insert a function into an expression, placeholders are used for parameter values you must supply.

4.20.12 Add Variable Structure Dialog

Use this dialog to define a variable and add it to the tree of variable structures in the [Variable Structures Page](#) of the [Expression Editors](#).

You create variable structures in this dialog to define the structure of a variable for design purposes. For example, it is easier to browse the XPath variable in the structure view rather than viewing the XML schema of the XPath variable. Variable structures do not create variables. Variables are created at runtime as the target of the assign action in the stage.

You can declare your own variable structures based on:

- XML types, including
 - Schema elements
 - WSDL elements
 - Schema types
 - WSDL types
- MFLs
- Service interfaces
- Simple types (string or any XML)

You can use this feature directly for all user-defined variables, as well as `$inbound`, `$outbound`, and `$fault`. However, you cannot use it directly to access XML attachments in `$attachments`, headers in `$header`, or documents and RPC parameters in `$body`, with one exception— you can use it directly to access documents and parameters in `$body` for request messages received by a WSDL proxy service.

When you create a variable structure based on XML types, MFLs, or service interfaces, you must enter the following information into the fields at the top of the page:

Table 4–89 Variable Structure Options

Option	Description
Structure Label	A display name for the variable you want to create. This display name enables you to give a meaningful name to the structure so you can recognize it at design time but it has no impact at run time.
Structure Path	The path of the variable structure at run time

4.21 New XSL Transformation

Transformation maps describe the mapping between two data types. eXtensible Stylesheet Language Transformation (XSLT) maps describe XML-to-XML mappings.

Use this page to select a name and location for a new XSL transformation. This wizard creates a skeleton for the XSLT. Add details in the XSLT editor.

Table 4–90 *New XSL Transformation Options*

Option	Description
Enter or select the parent folder	Enter or select the name of the project or folder in which to locate a new XSL transformation.
File Name	Enter a name for this new XSL transformation.

4.22 Split-Join User Interface Reference

The following sections describe the fields and user interface components involved with creating and configuring Split-Joins in Oracle Service Bus.

- [Section 4.22.1, "Split-Join Design Palette"](#)
- [Section 4.22.2, "Global / Start Node Properties"](#)
- [Section 4.22.3, "Variable Properties"](#)
- [Section 4.22.4, "Error Handler Properties"](#)
- [Section 4.22.5, "Invoke Service Properties"](#)
- [Section 4.22.6, "Reply Properties"](#)
- [Section 4.22.7, "For Each Properties"](#)
- [Section 4.22.8, "If Properties"](#)
- [Section 4.22.9, "If and Else If Properties"](#)
- [Section 4.22.10, "Parallel Properties"](#)
- [Section 4.22.11, "Raise Error Properties"](#)
- [Section 4.22.12, "Repeat Until Properties"](#)
- [Section 4.22.13, "Re-Raise Error Properties"](#)
- [Section 4.22.14, "Scope Properties"](#)
- [Section 4.22.15, "While Properties"](#)
- [Section 4.22.16, "Wait Properties"](#)
- [Section 4.22.17, "Assign Properties"](#)
- [Section 4.22.18, "Copy Properties"](#)
- [Section 4.22.19, "Delete Properties"](#)
- [Section 4.22.20, "Insert Properties"](#)
- [Section 4.22.21, "Java Callout Properties"](#)
- [Section 4.22.22, "Log Properties"](#)
- [Section 4.22.23, "Replace Properties"](#)
- [Section 4.22.24, "Receive Properties"](#)

- [Section 4.22.25, "Counter Variable Dialog"](#)
- [Section 4.22.26, "Create/Edit Variable Dialog"](#)
- [Section 4.22.27, "Create Message Variable Dialog"](#)
- [Section 4.22.28, "Service Browser"](#)
- [Section 4.22.29, "SOAP Fault Variable Dialog"](#)
- [Section 4.22.30, "WSDL Browser"](#)
- [Section 4.22.31, "Split-Join Wizard - New Split-Join"](#)
- [Section 4.22.32, "Split-Join Wizard - Specify Operations"](#)

4.22.1 Split-Join Design Palette

The Split-Join Design Palette lists all the operations you can use to construct a Split-Join. To insert a control into a Split-Join, drag the icon from the palette to the Split-Join editor. When you drag an item to the editor, one or more target icons indicate that you can drop the dragged item in that position on the editor. When you drag the item into a target icon, it is highlighted to show that you can drop the item there.

4.22.1.1 Operations

The Split-Join Design palette is organized into the following categories:

- Communication
- Flow Control
- Assign Operations

The following tables describe the operations you can add to a Split-Join:

Table 4–91 Split-Join Communication Operations

Operation	Description
Invoke Service	Invoke Service invokes a WSDL-based, non-transport-typed Business Service, a WSDL-based Proxy Service, or a Split-Join. For configuration properties, see Section 4.22.5, "Invoke Service Properties."
Reply	Reply sends a response or fault back to the Oracle Service Bus Message Flow. For configuration properties, see Section 4.22.6, "Reply Properties."

Table 4–92 Split-Join Flow Control Operations

Operation	Description
For Each	For Each executes logic configured within its Scope a specified number of times. For configuration properties, see Section 4.22.7, "For Each Properties."
If	If provides conditional behavior within a Split-Join. For configuration properties, see Section 4.22.8, "If Properties."

Table 4–92 (Cont.) Split-Join Flow Control Operations

Operation	Description
Parallel	Parallel creates a fixed number of configured parallel branches. For configuration properties, see Section 4.22.10, "Parallel Properties."
Raise Error	Raise Error generates an error that causes the Split-Join to stop normal processing. If the error is not handled using an Error Handler, the Split-Join will terminate and a Fault will be sent to the Oracle Service Bus Message Flow. For configuration properties, see Section 4.22.11, "Raise Error Properties."
Repeat Until	Repeat Until lets you repeat operations until a condition evaluates to true within a Split-Join. The condition is evaluated after each loop finishes. For configuration properties, see Section 4.22.12, "Repeat Until Properties."
Re-Raise Error	Re-Raise Error lets you re-raise an error caught by an Error Handler Catch or CatchAll. For configuration properties, see Section 4.22.13, "Re-Raise Error Properties."
Scope	Scope creates a context which influences the behavior of its enclosed operations. For configuration properties, see Section 4.22.14, "Scope Properties."
While	While lets you repeat operations until a condition evaluates to false within a Split-Join. The condition is evaluated before each loop commences. For configuration properties, see Section 4.22.15, "While Properties."

Table 4–93 Split-Join Assign Operations

Operation	Description
Assign	Lets you assigns the result of an XQuery expression to a Variable. For configuration properties, see Section 4.22.17.1, "Assign Operation Properties."
Copy	Lets you copy the information specified by an XPath expression from a source document to a destination document. For configuration properties, see Section 4.22.18, "Copy Properties."
Delete	Lets you delete a set of nodes specified by an XPath Expression. For configuration properties, see Section 4.22.19, "Delete Properties."
Insert	Lets you insert the result of an XQuery expression at an identified place relative to nodes selected by an XPath Expression. For configuration properties, see Section 4.22.20, "Insert Properties."

Table 4–93 (Cont.) Split-Join Assign Operations

Operation	Description
Java Callout	Lets you invoke a static Java method from a Split-Join for custom actions such to be handled in Java such as validation, transformation, and logging. For configuration properties, see Section 4.22.21, "Java Callout Properties."
Log	Lets you log data at a specified severity so that administrators can take appropriate action. For configuration properties, see Section 4.22.22, "Log Properties."
Replace	Lets you replace a node or the contents of a node specified by an XPath expression.s For configuration properties, see Section 4.22.23, "Replace Properties."

Operations have a General properties tab for changing the node's label and providing comments.

4.22.2 Global / Start Node Properties

The start node in a Split-Join specifies its global properties. Among these properties, global variables and associated External Services can be reviewed and configured by expanding the left-side arrow. The global Error Handler can be reviewed and configured by expanding the right-side arrow.

Use the Properties view to review and configure the Global Properties of the selected Split-Join.

This page has two tabs:

- Imports
- General

The Imports tab has the following options:

Table 4–94 Split-Join Start Node Options - Imports Tab

Option	Description
WSDL Imports	Displays WSDL Imports used by the Split-Join. Select a WSDL in the list and right-click to delete it.
Schema Imports	Displays Schema Imports used by the Split-Join. Select a Schema in the list and right-click to delete it.

The General tab has the following options:

Table 4–95 Split-Join Start Node Options - General Tab

Option	Description
Label	Enter a label for the file defining the Process Node.
Documentation	Enter a description and/or comments.

4.22.3 Variable Properties

Use the Properties view to review and configure Variables in the selected Split-Join.

Table 4–96 Split-Join Variable Options

Option	Description
(tree)	Depicts the hierarchical structure of the Variable's type.
Edit	Click to display the Create/Edit Variable Dialog , where you can modify the Variable's name and type.

4.22.4 Error Handler Properties

The Error Handler receives and handles all of the errors that are raised in a Split-Join.

An Error Handler lets you add Catch and CatchAll operations.

Use the Properties view to review and configure the selected Error Handler in a Split-Join.

The Catch tab has the following options.

Table 4–97 Split-Join Error Handler Options - Catch Tab

Option	Description
SOAP Fault Variable Name	Defines a variable to contain SOAP (1.1 or 1.2) faults. If the Error Handler is executed due to a SOAP fault received from invoked external services, this variable is populated with the received SOAP fault.
Fault Name – Define Fault	Lets you define a custom local fault. Click Define Fault to enter a Namespace and Fault Name.
Fault Name – Predefined	Lets you select an existing WSDL, Application, or Standard fault. Click Pick Fault to select an existing fault.

The catchAll tab has the following options.

Table 4–98 Split-Join Error Handler Options - CatchAll Tab

Option	Description
SOAP Fault Variable Name	Defines a variable to contain SOAP (1.1 or 1.2) faults. If the Error Handler is executed due to a SOAP fault received from invoked external services, this variable is populated with the received SOAP fault.

4.22.5 Invoke Service Properties

The Invoke Service invokes a WSDL-based, non-transport-typed Business Service, a WSDL-based Proxy Service, or another Split-Join.

Use the Properties view to review and configure the selected Invoke Service in a Split-Join.

The Operation tab has the following options:

Table 4–99 Split-Join Invoke Service Options - Operation Tab

Option	Description
Operation	The operation to be invoked by the Service. Click Browse to select the operation you want to invoke. When you select an operation, a dashed blue line appears pointing to the external service in the Split-Join editor.

Table 4–99 (Cont.) Split-Join Invoke Service Options - Operation Tab

Option	Description
Service Location	The location of the invoked Service. Click the location path to open the service file.
Qos	The quality of service option that controls transaction support. Select one of the following: <ul style="list-style-type: none"> ▪ Best Effort – The operation does not execute in the context of an existing transaction. ▪ Exactly Once – The operation executes in the context of an existing transaction.

The Input Variable tab has the following options:

Table 4–100 Split-Join Invoke Service Options - Input Variable Tab

Option	Description
Message Variable	A list of message type variables with the type matching the operation's input message type. Select Create Message Variable to define a new message variable. Note: If message type variables with the type matching the operation's input message type do not exist, you must define a new message type variable with the required type.
Message Type Namespace	The namespace of the operation's input message type.
Message Type	The operation's input message type.

The Output Variable tab has the following options:

Table 4–101 Split-Join Invoke Service Options - Output Variable Tab

Option	Description
Message Variable	A list of message type variables with the type matching the operation's output message type. Select Create Message Variable to define a new message variable. Note: If message type variables with the type matching the operation's output message type do not exist, you <i>must</i> define a new message type variable with the required type.
Message Type Namespace	The namespace of the operation's output message type.
Message Type	The operation's output message type.

4.22.5.1 Invoking Another Split-Join

A Split-Join can invoke another Split-Join in the same Oracle Service Bus configuration. This functionality provides more flexibility in service design, letting you split up complex Split-Join functionality into multiple Split-Joins, allowing for componentization and re-use of Split-Join functionality. Performance is maintained, because there is no marshalling and unmarshalling of data between the Split-Joins.

You must ensure that you do not create circular Split-Join references. Oracle Service Bus does not check for circular references.

4.22.6 Reply Properties

Reply sends a response or fault back to the Oracle Service Bus message flow.

Use the Properties view to review and configure the selected Reply in a Split-Join.

The Operation tab has the following options:

Table 4–102 *Split-Join Reply Options - Operation Tab*

Option	Description
Operation	The operation to be invoked by the Reply.

The Variable tab has the following Select options:

Table 4–103 *Split-Join Reply Options - Variable Tab*

Response Options	Description
Message Variable	A list of the message variables whose type matches the operation's output message type. Select Create Message Variable to define a new message variable. Note: If message type variables with the type matching the operation's output message-type do not exist, you must define a new message type variable with the required type.
Message Type Namespace	The namespace of the operation's output message type.
Message Type	The operation's output message type.
Pick WSDL Fault/SOAP Fault	Determine whether the fault reply is a fault message defined in the operation of the WSDL or an explicit SOAP fault message.
WSDL Fault Name	Select a fault name from the list of faults defined in the operation of the WSDL.
Message Variable	A list of the Message Variables whose type matches the operation's output message type. Select Create Message Variable to define a new message variable. Note: If message type variables with the type matching the operation's output message-type do not exist, you must define a new message type variable with the required type.
Message Type Namespace	The namespace of the operation's output message type.
Message Type	The operation's output message type.
SOAP Fault	Select SOAP Fault See Section 4.22.29, "SOAP Fault Variable Dialog."
Propagate SOAP Fault	Propagate the SOAP fault in the SOAP fault variable defined in the Error Handler. See Section 4.22.4, "Error Handler Properties."

Reply automatically includes an implicit Exit operation to end that instance of the flow without triggering a fault. The Exit operation is not visible in the development environment.

4.22.7 For Each Properties

For Each executes logic configured within its Scope a specified number of times.

Use the Properties view to review and configure the selected For Each in a Split-Join. The Counter Variables tab has the following options:

Table 4–104 Split-Join For Each Options - Counter Variables Tab

Option	Description
Parallel	Select one of the following options: <ul style="list-style-type: none"> ■ If you select yes, each iteration of For Each is executed in parallel. ■ If no, each iteration of For Each is executed sequentially.
Counter Variable Name	Defines an implicit variable within the Scope of the For Each. Each iteration of the For Each contains an isolated instance of this variable. In turn, every instance is set to an iteration number; for example, the first iteration has its value set to Start Counter Value , the second iteration to Start Counter Value+1 , etc.
Start Counter Value	The value of the Counter Variable for the first iteration of For Each. Determined as the result of an XPath expression. The result must be "1" or more. ("0" is not a valid Start Counter Value.) The browse button launches the expression builder. The expression should generate an integer for the initial Start Count Value.
Final Counter Value	The value of the Counter Variable for the final iteration of For Each. Determined as the result of an XPath expression. The result must be "1" or more. ("0" is not a valid Final Counter Value.) The browse button launches the expression builder. The expression should generate an integer for the Final Count Value.

The Completion Condition tab has the following options:

Table 4–105 Split-Join For Each Options - Completion Condition

Option	Description
Number of Finished Branches	An optional expression that determines when to stop creating branches. Depending on the context in the Split-Join, the expression prevents some of the children from executing or forces early termination of some children. The browse button launches the expression builder.
Successful Branches Only?	Select this option if you want only successfully completed branches to be counted when determining if the completion condition has been met.

4.22.8 If Properties

If nodes contain If, Else If, and Else operations. See [Section 4.22.9, "If and Else If Properties."](#)

Use the Properties view to review and configure the selected If in a Split-Join.

4.22.9 If and Else If Properties

The If and Else If operations provide conditional behavior within a Split-Join.

Use the Properties view to review and configure the selected If or Else If in a Split-Join.

If and Else If have the following options:

Table 4–106 Split-Join If and Else If Options

Option	Description
Condition	Define an XPath expression that evaluates to true or false. If the condition evaluates to true, the associated If/Else If branch is executed. Click the browse icon to launch the expression builder.

4.22.10 Parallel Properties

Parallel lets you create a static Split-Join that handles fixed number of message requests. Parallels contain one or more Scope branches.

Use the Properties view to review and configure the selected Parallel in a Split-Join.

4.22.11 Raise Error Properties

Raise Error establishes an error condition under which the execution of the process will halt.

Use the Properties view to review and configure the selected Raise Error in a Split-Join.

The Raise Error tab has the following options:

Table 4–107 Split-Join Raise Error Options

Option	Description
Define Fault	Lets you specify a custom local fault not contained in the WSDL. Click the Fault link to specify the fault Namespace and Fault Name.
Pick Fault From WSDL	Lets you select an existing fault specified in the WSDL. Click the Pick Fault link to select the WSDL fault.

4.22.12 Repeat Until Properties

Repeat Until lets you repeat operations until a condition evaluates to true within a Split-Join. The condition is evaluated after each loop finishes.

Use the Properties view to review and configure the selected Repeat Until in a Split-Join.

The Condition tab has the following options:

Table 4–108 Split-Join Repeat Until Options

Option	Description
Condition	Define an XPath expression that evaluates to true or false. The operation(s) in the Repeat Until are executed until the condition evaluates to true. Click the browse icon to launch the expression builder.

4.22.13 Re-Raise Error Properties

Use Re-Raise Error within an Error Handler to re-raise an error caught by a Catch or a CatchAll. In the Properties view you can rename and enter comments about the Re-Raise Error node.

Use the Properties view to review and configure the selected Re-Raise Error in a Split-Join.

4.22.14 Scope Properties

The Scope creates a context which influences the behavior of its enclosed operations. Local variables and the Error Handler defined within the Scope are restricted to this context.

Use the Properties view to review and configure the selected Scope in a Split-Join.

Related Topics

[Section 4.22.26.1, "Scope and Variables"](#)

4.22.15 While Properties

While lets you repeat operations until a condition evaluates to false within a Split-Join. The condition is evaluated before each loop commences.

Use the Properties view to review and configure the selected While in a Split-Join.

The Condition tab has the following options:

Table 4–109 Split-Join While Options

Option	Description
Condition	Define an XPath expression that evaluates to true or false. If the condition evaluates to true, the associated operations is executed. Click the browse icon to launch the expression builder.

4.22.16 Wait Properties

Wait lets you insert a pause in the Split-Join flow for a short duration to wait for other dependent jobs to complete. After the short duration, the Wait continues the Split-Join execution.

Use the Properties view to review and configure the selected Wait in a Split-Join.

The Wait tab has the following options:

Table 4–110 Split-Join Wait Options

Option	Description
Duration	Define an XPath expression that evaluates to a duration type of xsd:duration in the following format: nYnMnDTnHnMnS (number of years, months, days, hours, minutes, and seconds, with a date/time separator, represented by "T".) Be sure the expression resolves to a correct duration type. The Wait operation does not validate the results of the expression. When the duration is reached, the Wait resumes the Split-Join execution. Click the browse icon to launch the expression builder.

4.22.17 Assign Properties

Assign lets you perform data manipulation, including initializing and updating a Variable. You can perform the following operations in an Assign node: Assign, Copy, Delete, Insert, Java Callout, Log, and Replace.

Use the Properties view to review and configure the selected Assign in a Split-Join.

4.22.17.1 Assign Operation Properties

The Assign tab has the following options:

Table 4–111 *Split-Join Assign Operation Options*

Option	Description
Expression	<p>An XQuery expression used to create the data that will be assigned to the Variable.</p> <p>Clicking the expression launches the XQuery editor.</p> <p>When Oracle Service Bus binds variables in an inline XQuery, it assumes the type <code>xs:string</code>. This can cause parser errors in operations with constants that are incompatible with <code>xs:string</code>. To ensure compatible types, use an explicit XQuery cast. For example, the following inline XQuery will fail. Although the <code>\$itemsTotal</code> is of type <code>xs:double</code>, it is bound as an <code>xs:string</code>, which is incompatible in the test against 10000.</p> <pre>if (\$itemsTotal < 10000) then . . .</pre> <p>To make this inline XQuery work, explicitly cast the <code>\$itemsTotal</code> to an <code>xs:double</code>:</p> <pre>if ((\$itemsTotal cast as xs:double) < 10000) then . . .</pre> <p>Also, when creating an Assign action to a String result or variable, ensure that your expression returns a String value. Assigning a non-String value to a String result or String global variable does not cause a <code>MismatchedAssignmentFailure</code> exception, as specified by the WS-BPEL specification.</p>
Variable	<p>A variable to which the result of the XQuery expression is assigned.</p> <p>Only previously defined Variables, the Counter Variable, and SOAP Fault Variables (for Error Handlers) are available.</p>
Qos	<p>The quality of service option that controls transaction support. Select one of the following:</p> <ul style="list-style-type: none"> ■ Best Effort – The operation does not execute in the context of an existing transaction. ■ Exactly Once – The operation executes in the context of an existing transaction.

Oracle Service Bus's Assign functionality in Split-Joins conforms to the WS-BPEL specification for resolution of XPath/XQuery expressions to simple type variables. Supported simple types for binding XPath/XQuery expressions to variables in Split-Joins are String, Boolean, and Float. The Assign converts the value you provide the type with which the variable is defined.

For example:

- If you assign `<foo><bar>4</bar></foo>` to a response variable defined as a String (`$response.result`), Oracle Service Bus returns `<bar>4</bar>` as a String in the `<result>` through a simple copy of the child element and value.

- If you map `<foo><bar>4</bar></foo>` to a String variable (such as `myStr`), then assign `$myStr` to `$response.result`, Oracle Service Bus returns `<result>4</result>`, because it first converts the value in `$myStr` to a String before it makes the assignment to the `$response.result` String variable.

4.22.18 Copy Properties

The Copy operation copies the information specified by an XPath expression from a source document to a destination document.

Use the Properties view to configure a Copy Operation in an Assign. Configure Select From and Select To using the following guidance.

The Copy tab has the following options:

Table 4–112 Split-Join Copy Options

Option	Description
Keep Source Element	<p>Lets you determine which element name (source or destination) is used when values are copied from a source to a destination.</p> <p>If you do not select this option, the existing element name in the destination is used to hold the copied value.</p> <p>If you select this option, the name of the source element is used in the destination to hold the copied value.</p> <p>For example, if you are copying a zip code value from <code><zipCode>80303</zipCode></code> in the source, selecting Keep Source Element uses the <code><zipCode></code> element in the destination. If you do not select the option, the zip code value is copied to the existing destination element, such as <code><pinCode>80303</pinCode></code>.</p>
Choose Type	<p>Allows the user to select the desired type: Variable, Expression, Literal, or XML Fragment. Literals and XML fragments are available only in the Select From menu.</p>
Choose Type - Variable	<p>Select an XPath on a Variable. Use this option when simple node selection on a Variable is required.</p> <p>When a node is selected by expanding the Variable tree, an Xpath expression is automatically generated.</p>
Choose Type - Expression	<p>Create an Xpath expression to select a node from a Variable. Use this option when more complex Xpaths (potentially with predicates) are required to select a node on a Variable.</p> <p>Click the Expression link to use the XPath Expression Builder.</p> <p>Note: The entered Xpath expression should only copy to one XML node/element, or there will be run-time errors.</p>
Choose Type - Literal (Select From only)	<p>A Literal string entered by the user.</p> <p>Click the Literal link to enter the string.</p>
Choose Type - XML Fragment (Select From only)	<p>An XML fragment entered by the user.</p> <p>Click the XML Fragment link to enter the fragment.</p>

4.22.19 Delete Properties

The Delete operation deletes a set of nodes specified by an XPath Expression.

Use the Properties view to configure a Delete Operation in an Assign.

Note: Unlike a Delete operation in a proxy service message flow, a Delete operation in a Split-Join does not allow deleting a variable directly.

The Delete tab has the following options:

Table 4–113 *Split-Join Delete Options*

Option	Description
XPath	An XPath Expression that selects the nodes to be deleted. Click the XPath link to launch the expression editor.
In Variable	Variable on which the XPath expression is executed to select the nodes to be deleted.
Qos	The quality of service option that controls transaction support. Select one of the following: <ul style="list-style-type: none"> ■ Best Effort – The operation does not execute in the context of an existing transaction. ■ Exactly Once – The operation executes in the context of an existing transaction.

4.22.20 Insert Properties

The Insert operation inserts the result of an XQuery expression at an identified place relative to nodes selected by an XPath Expression.

Use the Properties view to configure an Insert Action in an Assign.

The Insert tab has the following options:

Table 4–114 *Split-Join Insert Options*

Option	Description
Expression	An XQuery expression used to create the data that will be inserted at a specified location in a Variable. Click the Expression link to launch the expression editor.
Location	The location used to control where the Insert operation is performed relative to the result of the XPath Expression. Options include: <ul style="list-style-type: none"> ■ before: Immediately before the element specified by the result of the Xpath Expression. ■ after: Immediately after the element specified by the result of the Xpath Expression. ■ as first child of: The first child element of the element specified by the result of the Xpath Expression. ■ as last child of: The last child element belonging to the element specified by the result of the Xpath Expression.
XPath	An XPath that determines the nodes to be selected. Click the Xpath link to launch the expression editor.
Variable	A Variable to be evaluated by the XPath.

Table 4–114 (Cont.) Split-Join Insert Options

Option	Description
Qos	<p>The quality of service option that controls transaction support. Select one of the following:</p> <ul style="list-style-type: none"> ▪ Best Effort – The operation does not execute in the context of an existing transaction. ▪ Exactly Once – The operation executes in the context of an existing transaction.

4.22.21 Java Callout Properties

A Java Callout operation lets you invoke a static Java method from a Split-Join for custom actions such to be handled in Java such as validation, transformation, and logging.

Use the Properties view to configure a Java Callout operation in an Assign.

The Java Callout tab has the following options:

Table 4–115 Split-Join Java Callout Options

Option	Description
Method	<p>Package your Java class in a JAR file in your Oracle Service Bus project. Click Browse to first select the JAR, then the method you want to invoke.</p> <p>Using the following guidelines for the Java callout method:</p> <ul style="list-style-type: none"> ▪ The method must be static. ▪ Only the following Java types are supported for input parameters: <ul style="list-style-type: none"> ▪ boolean, byte, char, double, float, int, long, short and arrays of these types ▪ java.lang.[Boolean Byte Character Double Float Integer Long Short String] and arrays of these types ▪ java.math.[BigInteger BigDecimal] and arrays of these types ▪ org.apache.xmlbeans.XmlObject and arrays of this type ▪ Only the following Java types are supported for method return: <ul style="list-style-type: none"> ▪ All types supported for input parameters except their array equivalent ▪ void
Expression	<p>An XQuery expression used to map data to the input parameters of the static Java method. Click the Expression link to launch the expression editor.</p>
Result Value	<p>Select the variable to contain the result value for the Java method.</p>
Service Account	<p>You can use Service Account to put the appropriate subject on the thread when executing the Java callout.</p> <p>Click Browse to select a service account.</p>
Qos	<p>The quality of service option that controls transaction support. Select one of the following:</p> <ul style="list-style-type: none"> ▪ Best Effort – The operation does not execute in the context of an existing transaction. ▪ Exactly Once – The operation executes in the context of an existing transaction.

Oracle Service Bus provides the following errors for Java callouts:

- 2031350 – Received more than one element when only one is expected during conversion to Java
- 2031351 – Found simple type instead of XmlObject when converting to Java
- 2031352 – Error converting simple type to its corresponding Java type
- 2031353 – Received exception during invocation of the Java method
- 2031354 – Errors setting the security context

4.22.22 Log Properties

The Log operation lets you log Split-Join data at a specified severity to the server log file. Administrators can use log information to take appropriate action based on the severity of the data logged.

Use the Properties view to configure a Log operation in an Assign.

The Log tab has the following options:

Table 4–116 *Split-Join Log Options*

Option	Description
Expression	An XQuery expression used to select the data to be logged. Click the Expression link to launch the expression editor.
Annotation	Optionally specify a note for the log. The annotation is logged along with the data selected by the expression.
Severity	Select one of the following the severity levels for the log: <ul style="list-style-type: none"> ■ Debug ■ Info ■ Warning ■ Error

4.22.23 Replace Properties

The Replace operation lets you replace a node or the contents of a node specified by an XPath expression.

Use the Properties view to configure a Replace operation in an Assign.

The Replace tab has the following options:

Table 4–117 *Split-Join Replace Options*

Option	Description
XPath	An XPath Expression used to specify the data (in the Variable) that will be replaced. Click the XPath link to launch the expression editor.
Variable	The Variable that contains the data to be replaced.
Expression	An XQuery expression used to create the data that replaces the data specified by the XPath in the named Variable. Click the Expression link to launch the expression editor.
Replace entire node	Specifies that the nodes selected by the expression are replaced along with all of its contents.

Table 4–117 (Cont.) Split-Join Replace Options

Option	Description
Replace node contents	Specifies that the node is not replaced. Only the contents are replaced.
Qos	The quality of service option that controls transaction support. Select one of the following: <ul style="list-style-type: none"> ▪ Best Effort – The operation does not execute in the context of an existing transaction. ▪ Exactly Once – The operation executes in the context of an existing transaction.

4.22.24 Receive Properties

Use the Properties view to review and configure the selected Receive in a Split-Join.

The Operation tab has the following options:

Table 4–118 Split-Join Receive Options - Operation Tab

Option	Description
Operation	The operation to be invoked by the Receive. Browse to select an operation from the WSDL Browser .

The Variable tab has the following options:

Table 4–119 Split-Join Receive Options - Variable Tab

Option	Description
Message Variable	A list of the message variables whose type matches the operation's output message type Select Create New Variable to define a new message variable. Note: If message type variables with the type matching the operation's output message-type do not exist, you <i>must</i> define a new message type variable with the required type.
Message Type Namespace	The namespace of the operation's output message type.
Message Type	The operation's output message type.

4.22.25 Counter Variable Dialog

Use this dialog to create a Counter Variable.

Table 4–120 Split-Join Counter Variable Options

Option	Description
Counter Variable Name	The name of the Counter Variable

4.22.26 Create/Edit Variable Dialog

Use this dialog to review and/or configure Variables in the Split-Join.

4.22.26.1 Scope and Variables

Although variables are visible in the scope in which they are defined and in all scopes nested within that scope, a variable declared in an outer scope is hidden when you declare a variable with an identical name in an inner scope. For example, if you define variable myVar in an outer scope (So) and then define variable myVar again in an inner scope (Si) which is contained by scope So, then you can only access the myVar you defined in the inner scope Si. This myVar overrides the myVar you defined in scope So.

Table 4–121 Split-Join Variable Options

Option	Description
Name	Enter a name for the Variable that is unique within the Scope.
Select Variable Type	Select one of the following Variable Types: <ul style="list-style-type: none"> ■ Built-in Types ■ Schema Types ■ Message Types
Select Variable Type - Built-in Types	When selected, only Built-in Type Variables are displayed.
Select Variable Type- Schema Types	When selected, all Schema Types in the current Oracle Service Bus Configuration are displayed. If "Show only applicable schema types/elements for this Split-Join" is selected, only Schema Types directly applicable to the current Split-Join are displayed. This is checked by default.
Select Variable Type - Message Types	When selected, only message types are displayed. If "Show only applicable schema types/elements for this Split-Join" is selected, only Message Types directly applicable to the current Split-Join are displayed. This is checked by default.
Type	The Variable type.
Namespace	The namespace of the Variable's type.

4.22.27 Create Message Variable Dialog

Use this dialog to create a new message variable.

Table 4–122 Split-Join New Message Variable Options

Option	Description
Name	Enter a unique name for the Variable.
Type	One of the following Variable types: WSDL message, XSD element, or XSD type (simple, complex, built-in).
Namespace	The namespace of the Variable's type.

4.22.28 Service Browser

Use this dialog to browse for and select an operation from a business service, a proxy service, or a Split-Join in the tree.

4.22.29 SOAP Fault Variable Dialog

Use this dialog to create a SOAP fault variable.

Table 4–123 Split-Join SOAP Fault Variable Options

Option	Description
SOAP Fault Variable Name	The name of the SOAP fault variable.

4.22.30 WSDL Browser

Use this dialog to browse for and select an operation from a WSDL in the depicted tree.

4.22.31 Split-Join Wizard - New Split-Join

Use this page to locate and name the new Split-Join

Table 4–124 New Split-Join Options

Option	Description
Enter or select the parent folder	Enter the name of the folder to contain this Split-Join, or select a folder from the list.
File Name	Enter a name for the file defining the Split-Join.

4.22.32 Split-Join Wizard - Specify Operations

Use this page to select an operation for the new Split-Join.

Table 4–125 New Split-Join Options - Selecting an Operation

Option	Description
Select Operation	Select the operation from those available in the tree.
Consume	Click this button to import WSDLs from outside your current Oracle Service Bus configuration that can subsequently be used to select an operation.

Part II

XQuery Mapper

This part contains the XQuery Mapper IDE help, which includes the following chapters:

- [Chapter 5, "Introduction"](#)
- [Chapter 6, "Transforming Data Using XQuery Mapper"](#)
- [Chapter 7, "Examples: Data Transformation Using XQuery Mapper"](#)
- [Chapter 8, "Upgrading XQuery Code"](#)

Introduction

Oracle XQuery Mapper is a graphical mapping tool that enables you to transform data between XML, non-XML, and Java data types, allowing you to integrate heterogeneous applications rapidly. For example, you can package data transformations in Oracle WebLogic Integration (WLI) as controls and reuse the controls in multiple business processes and applications. You can also use .xq files created in XQuery Mapper as resources in Oracle Service Bus.

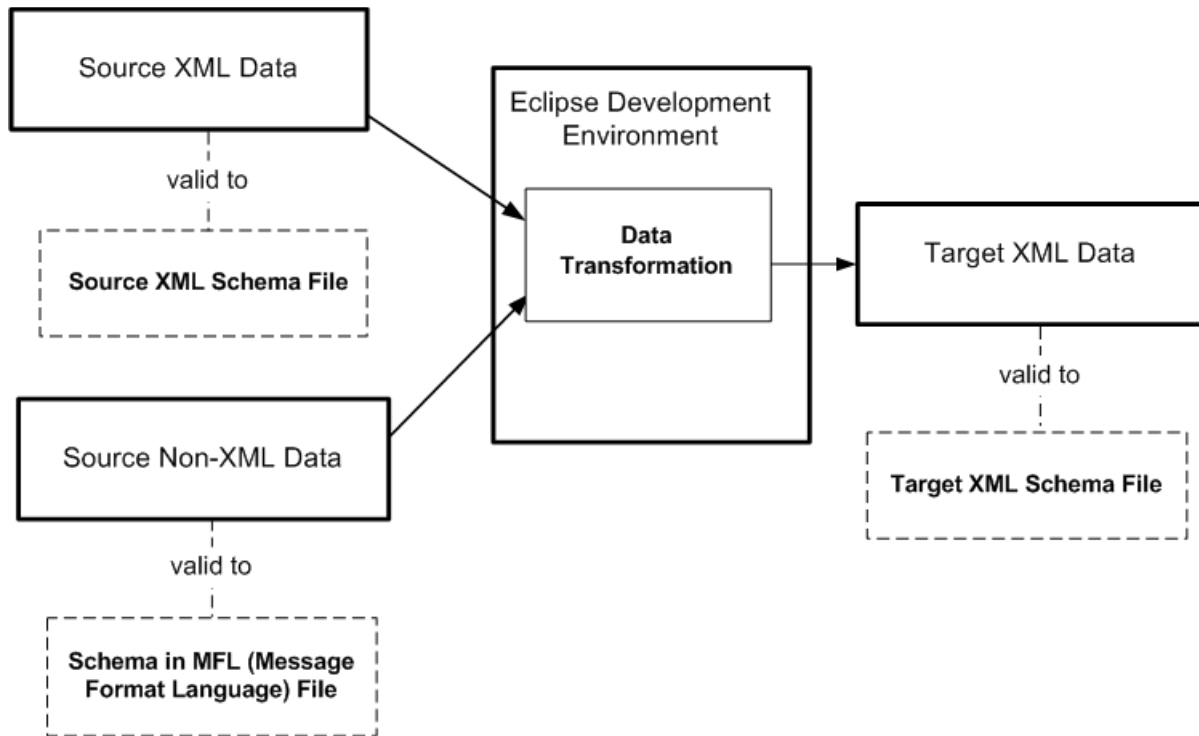
The output of XQuery Mapper is a query in the XQuery language, which is defined by the World Wide Web Consortium (W3C). For more information about W3C and the XQuery language, see <http://www.w3.org/XML/Query/>.

5.1 Overview of XQuery Mapper

You can use XQuery Mapper to transform data between XML, non-XML, and Java data types. For example, XML data that is valid against one schema can be converted to XML that is valid against a different schema. The data can be based on XML schemas, Web Service Definition Language (WSDL) file, and Message Format Language (MFL) files.

When you select the **Simple** source type, you can transform standard schema types, such as `boolean`, `byte`, `double`, `float`, `int`, `long`, `short`, `String`, and `Date`, to any other required target data format.

A data transformation can have multiple input types, but only one target type. For example, data can be transformed from two sources to one target, as shown in the following figure.

Figure 5–1 Data Transformation from Multiple Sources to One Target

5.2 Support for XQuery 2002 and 2004

WLI supports data transformation for the following versions of XQuery:

- XQuery 2004: Graphical design view (XQuery Mapper), source view, and test view.
- XQuery 2002: Source view and test view.

Note: For XQuery 2002-compliant XQuery files, the source view does not show compilation errors.

When you open an XQuery 2002-compliant XQuery file, it opens automatically in the **XQuery 2002 Transformation Editor**, which has **Source** and **Test** views, but no **Design** view.

5.2.1 Restrictions Applicable to the XQuery Test View

Table 5–1 describes restrictions in XQuery test view.

Table 5–1 Restrictions Applicable to the XQuery Test View

Restriction	Applicable to XQ2002?	Applicable to XQ2004?
If an XQuery calls a Java user function, the Java method must be static.	Yes	Yes

Table 5-1 (Cont.) Restrictions Applicable to the XQuery Test View

Restriction	Applicable to XQ2002?	Applicable to XQ2004?
If an input Java type argument to an XQuery is an abstract class or an interface, the test view can not process it. An error message is displayed in the results view.	Yes	No
The input Java type and its member variables (except those of type primitive, <code>String</code> , <code>java.sql.Date</code> and <code>java.util.Date</code>) must follow the standard Java Bean guidelines.	Yes	No

Transforming Data Using XQuery Mapper

You can use the graphical interface of XQuery Mapper to create data transformations, by mapping elements in source schemas to elements in a target schema. XQuery Mapper generates an XQuery, which is saved as an **.xq** file.

The procedure to transform data using XQuery Mapper is described in the following sections:

- [Section 6.1, "Launching XQuery Mapper"](#)
- [Section 6.2, "Importing the XQuery Mapper Sample Project"](#)
- [Section 6.3, "Creating an XQuery Mapper Project"](#)
- [Section 6.4, "Importing and Creating Schema Files"](#)
- [Section 6.5, "Selecting Source and Target Data Types"](#)
- [Section 6.6, "Creating Data Transformations"](#)
- [Section 6.7, "Editing Data Transformations"](#)
- [Section 6.8, "Restricting Output of Optional Elements"](#)
- [Section 6.9, "Testing Data Transformations"](#)
- [Section 6.10, "Graphical Features in Design View"](#)
- [Section 6.11, "XML Global Elements, Global Types, Local Elements, and Attributes"](#)

6.1 Launching XQuery Mapper

In Oracle Enterprise Pack for Eclipse, open the XQuery transformation perspective by choosing **Window > Open Perspective > XQuery Transformation** from the Eclipse menu.

The XQuery transformation perspective launches automatically when you open an XQuery file. If, however, XQuery Mapper is open and no XQuery file is open, you must launch the XQuery transformation perspective manually.

6.2 Importing the XQuery Mapper Sample Project

The XQuery Mapper sample project includes sample schema and XML files, which you can use to create XQuery transformations as described in [Chapter 7, "Examples: Data Transformation Using XQuery Mapper."](#)

Use the following procedure to import the sample project.

1. From the Eclipse menu bar, choose **File > Import**.
2. In the Import window, select **General > Existing Projects into Workspace**, then click **Next**.
3. In the Select root directory field, click **Browse**, then select the following folder: *OSB_ORACLE_HOME/eclipse/plugins/com.bea.alsb.common.mapper_version/samples*, and click **OK**.
4. In the Import window, select **Copy projects into workspace**, and click **Finish**.
5. Enter a name for your sample project and click **Finish**.

The samples project is created and displayed in the Project Explorer view. The project contains the following folders:

- **schemas** folder: Contains the XML schema (.xsd) files of the sample project.
- **XML** folder: Contains test XML files required by some of the samples.
- **XQueryTransformations** folder: This is the folder in which you will create the XQuery files for the samples.

To learn more about creating projects and importing the files you need for those projects, see the following topics:

- [Section 6.3, "Creating an XQuery Mapper Project"](#)
- [Section 6.4, "Importing and Creating Schema Files"](#)
- [Chapter 7, "Examples: Data Transformation Using XQuery Mapper"](#)

Note: You can import project-specific XML schemas, Web Service Definition Language (WSDL) files, and Message Format Language (MFL) files from any location. Before you import the files, it is recommended that you create a folder structure that meets your business needs. For more information, see [Section 6.4, "Importing and Creating Schema Files."](#)

6.3 Creating an XQuery Mapper Project

To create an XQuery Mapper project:

1. Switch to the XQuery transformation perspective by choosing **Window > Open Perspective > XQuery Transformation**.
2. Choose **File > New > Project**.
The **New Project** wizard is displayed.
3. Choose **General > Project** and click **Next**.
4. Enter a name for the project.
5. Ensure that the **Use default location** check box is selected.
6. Click **Finish**.

6.4 Importing and Creating Schema Files

Schema files can be created in and imported from any location. The following schema types are supported:

- **XSD (XML Schema Definition):** XML schemas describe and constrain data in XML files. Multiple namespaces are supported in XQuery Mapper. For example, you can transform data from two source XML files that are valid against a specific namespace to an XML file that is valid against another namespace.
- **WSDL (Web Service Definition Language):** XML schema defined in the WSDL file can be used in data transformations.
- **MFL (Message Format Language):** MFL files describe and constrain data in non-XML files (COBOL copybooks and C structure definitions, for example). The namespace of the MFL elements is derived from the name of the MFL file.
MFL files are created using the Format Builder tool and have the `.mfl` extension.

6.4.1 Importing XML Schemas and MFL Files

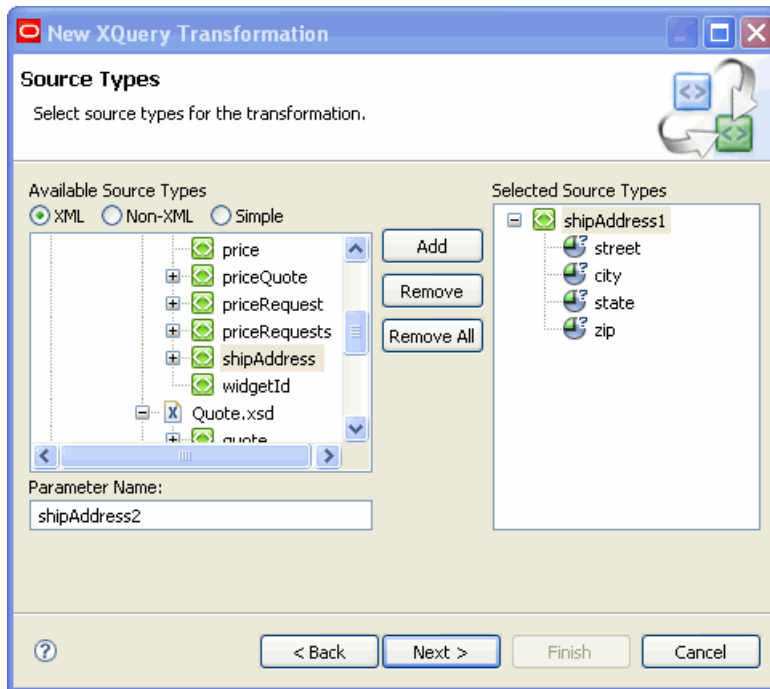
To import schemas and MFL files:

1. Switch to the XQuery transformation perspective by choosing **Window > Open Perspective > XQuery Transformation**.
2. In the **Navigator** view, select the project into which you want to import the XML schema or MFL file.
3. Choose **File > Import**.
The Import wizard is displayed.
4. You can import XML schemas and MFL files from a variety of sources. Select the appropriate source and click **Next**.
5. Browse for and select the required file, and click **Finish**.

After the schemas files or MFL files are imported, they are available in the New XQuery Transformation wizard.

The following figure shows how the imported XML schemas are displayed in the New XQuery Transformation wizard.

Figure 6–1 New XQuery Transformation



6.4.2 Creating XML Schemas

You can create XML schemas by using the XML Schema Editor.

1. Switch to the XQuery transformation perspective by choosing **Window > Open Perspective > XQuery Transformation**.
2. In the **Navigator** view, select the project in which you want to create the schema files.
3. Select **File > New > Other**.

The New screen is displayed.

4. Expand the **XML** node.
5. Select **XML Schema** and click **Next**.
6. Select the parent folder in which you want to create the schema file.
7. Enter a name for the schema file and click **Finish**.

The schema file is created in the specified project. You can now specify the details of the schema and save the file.

For information about using the XML Schema Editor, see *Introduction to the XSD Editor* at

<http://www.eclipse.org/webtools/community/tutorials/XMLSchemaEditor/XMLSchemaEditorTutorial.html>.

6.4.3 Creating XML Files from XML Schemas

To create XML files from schemas:

1. Switch to the XQuery transformation perspective by choosing **Window > Open Perspective > XQuery Transformation**.

2. In the **Navigator** view, select the project in which you want to create the schema files.
3. Select **File > New > Other**.
The New screen is displayed.
4. Expand the **XML** node.
5. Select **XML** and click **Next**.
The Create XML File dialog box is displayed.
6. Select the **Create XML file from an XML schema file** option and click **Next**.
7. Select the parent folder in which you want to create the XML file.
8. Enter a name for the XML file and click **Next**.
9. Select the XML schema based on which you want to create the XML file and click **Next**.
10. Select the root element of the XML file and click **Finish**.

The XML file is created in the specified folder. You can now specify the details of the file and save it.

For information about using the XML Editor, see *Creating XML files Tutorial* at <http://www.eclipse.org/webtools/community/tutorials/XMLWizards/XMLWizards.html>.

6.4.4 Creating WSDL Files

To create a WSDL file:

1. Switch to the XQuery transformation perspective by choosing **Window > Open Perspective > XQuery Transformation**.
2. In the **Navigator** view, select the project in which you want to create the WSDL files.
3. Select **File > New > Other**.
The New wizard is displayed.
4. Expand the **XML** node, select **WSDL**, and click **Next**.
5. Select the parent folder in which you want to create the WSDL file.
6. Enter a name for the WSDL file and click **Next**.
7. Enter the target namespace and prefix of the WSDL file.
8. If required, select the **Create WSDL Skeleton** check box.
9. Select the protocol and the binding option and click **Finish**.

The WSDL file is created in the specified project.

For information about using the WSDL Editor, see *Introduction to the WSDL Editor* at: <http://www.eclipse.org/webtools/community/tutorials/WSDLEditor/WSDLEditorTutorial.html>.

6.4.5 Creating MFL Files

You can create MFL files by using the Format Builder tool. In Eclipse, in the Oracle Service Bus perspective, choose (**File > New > MFL**).

6.5 Selecting Source and Target Data Types

Before you create a data transformation, you must define the source target types and a target data type. The source and target types can be non-XML, XML, and simple data types.

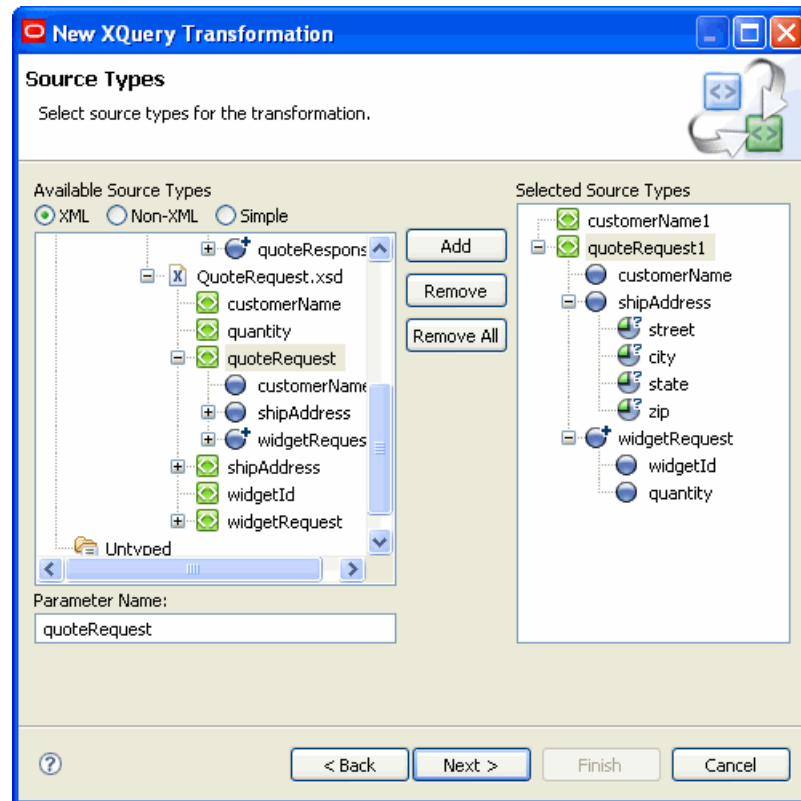
1. Select the project for which you want to select source and target data types.
2. Right-click and choose **New > XQuery Transformation**.
The New XQuery Transformation wizard is displayed.
3. Enter a name for the **.xq** file.
4. Click **Next**.
The Source Types dialog box is displayed.
5. In the **Available Source Types** pane select the source data types.
 - If the source is XML data, choose **XML**.
 - If the source is MFL data, choose **Non-XML**.
 - If the source data is of standard types, such as `boolean`, `String`, and `int`, then choose **Simple**.
6. Select the required source data elements.

Note: For schemas to be displayed in the Available Source Types and Available Target Types pane, the XML and non-XML files that contain these schemas must first be imported into or created in Eclipse.

The Available Source Types and Available Target Types panes show only schemas that exist in the **schemas** folder of the XQuery Transformation project. If you prefer to keep your schemas in any other folder, you must specify the path to that folder in the XMLBeans settings for the project (by choosing **Project > Properties**, the **XMLBeans** page, and then the **Source Paths** tab).

For example, to add input data from **schemas/Dates.xsd**, select the **date** element in the schema as the input element and click **Add**, as shown in the following figure.

Figure 6–2 Selecting Source Types



The elements and attributes that make up the selected element are displayed in the Selected Source Types pane.

7. After selecting the required source types, click **Next**.

The Target Types dialog box is displayed.

8. In the **Available Target Types** pane, select the target data type, and click **Add**.

The elements and attributes that make up the selected element are displayed in the Selected Target Type pane.

Note: You can specify only one target data type.

9. Click **Finish**.

The **.xq** file is displayed in the Design view. It shows the source and target data types that you selected.

6.6 Creating Data Transformations

You can perform the following types of data transformations:

- Basic element transformations: Mapping a source element to a target element.
- Basic attribute transformations: Mapping a source attribute to a target attribute.
- Complex transformations: Mapping a complex source (for example, a repeating element) to a complex target (for example, a non-repeating element).

6.6.1 Creating Basic Element Transformations

Basic element transformation involves mapping a source element to a target element. The source and target elements may have the same name, type, or scope.

The following are some examples of the types of basic element transformation that you can perform:

- **Element to element:** A source element is mapped to a target element.
- **Element combination:** Multiple source elements are combined to create a single target element.
- **Element explosion:** XQuery string functions are exploded from a single source element to multiple target elements.

6.6.1.1 Prerequisite

The XQuery file is created as described in [Section 6.5, "Selecting Source and Target Data Types"](#)

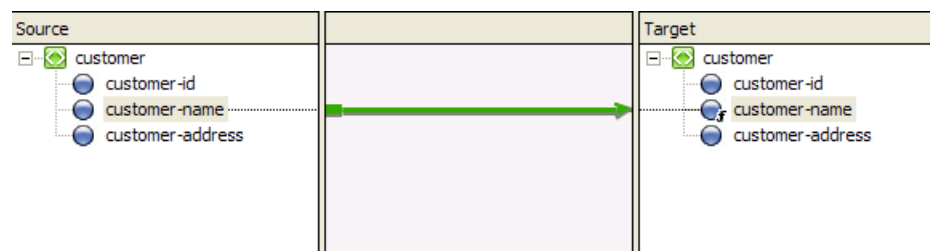
6.6.1.2 Creating Element-to-Element Links

To create element-to-element links:

1. Select the project for which you want to create element-to-element links, and open the XQuery file in which the transformation must be stored.
2. Drag the required element from the **Source** pane to the target element in the **Target** pane.

For example, to create a link between the **customer-name** element in the source schema and the **customer-name** element in the target schema, drag **customer-name** from the **Source** pane to the **Target** pane. An arrow connects the two elements, as shown in the following figure.

Figure 6–3 Element-to-Element Links



Note: While dragging from the **Source** pane to the **Target** pane, a dashed line appears temporarily between the two elements. For more information about link patterns, [Section 6.10.2, "Link Patterns."](#)

3. After creating the required element-to-element links, save the changes.

6.6.2 Creating Basic Attribute Transformations

Basic attribute transformation involves mapping a source attribute to a target attribute. The source and target attributes may have the same name, type, or scope.

The following are some examples of basic attribute transformations:

- **Element to attribute:** A source element is mapped to a target attribute.
- **Attribute to element:** A source attribute is mapped to a target element.
- **Attribute to attribute:** A source attribute is mapped to a target attribute of the same name.

6.6.2.1 Prerequisite

The XQuery file is created as described in [Section 6.5, "Selecting Source and Target Data Types"](#)

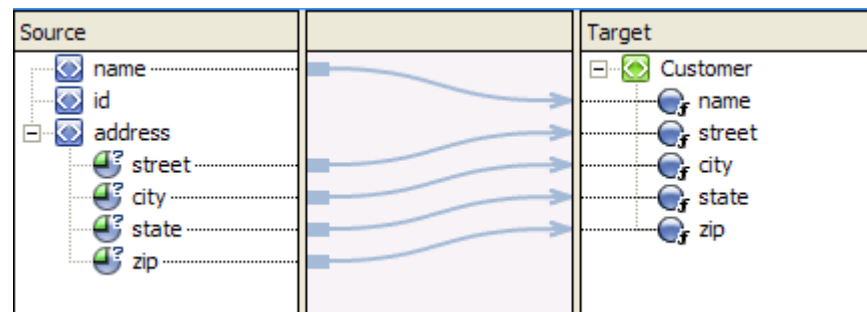
6.6.2.2 Creating an Attribute-to-Element Link

To create an attribute-to-element link:

1. Select the project for which you want to create attribute-to-element links, and open the XQuery file in which the transformation must be stored.
2. Drag the required attribute from the **Source** pane to the appropriate element in the **Target** pane.

For example, to create a link between the **street** attribute of the **address** element in the source schema and the **street** element of the target schema, drag the **street** attribute from the **Source** pane to the **Target** pane, as shown in the following figure.

Figure 6–4 Attribute-to-Element Link



3. After creating the required links, save the changes.

Similarly, you can create element-to-attribute and attribute-to-attribute links.

6.6.3 Creating Complex Transformations

Complex transformations involve mapping a complex source (for example, a repeating element) to a complex target (for example, a non-repeating element). The following are some examples of complex transformations:

- **Repeating group to repeating group:** The source contains a variable number of instances of a group of elements; each source instance is mapped to an instance of the target group.
- **Repeating group to non-repeating element:** The source contains a variable number of instances of a group of elements; each source group is mapped to an instance of the target **element**.

6.6.3.1 Prerequisite

The XQuery file is created as described in [Section 6.5, "Selecting Source and Target Data Types"](#)

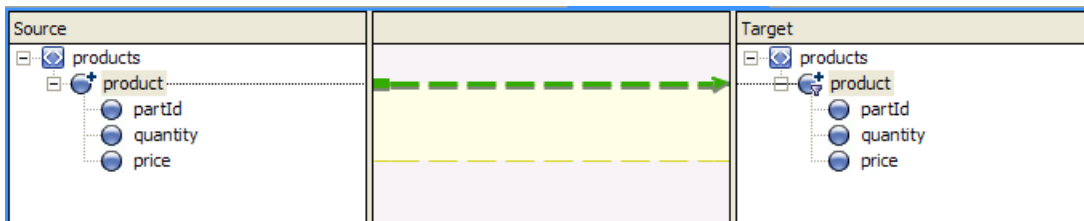
6.6.3.2 Creating a Complex Transformation

To create a complex transformation:

1. Select the project for which you want to create the links, and open the XQuery file in which the transformation must be stored.
2. Drag the required element or attribute from the **Source** pane to the appropriate element or attribute in the **Target** pane.

For example, to create a link between **product** (a repeating group in the source schema) and **product** (a repeating group in the target schema), drag **product** from the **Source** pane to the **Target** pane, as shown in the following figure.

Figure 6–5 Repeating-Group-to-Repeating-Group Link



3. After creating the required links, save the changes.

6.7 Editing Data Transformations

After creating a data transformation in the Design view, you can add, change, and delete XQuery code either by editing code directly in the **Source** view or by adding complex expressions in the **Design** view.

Note: For information about the XQuery language, see <http://www.w3.org/XML/Query>.

This section contains information about the following topics:

- [Section 6.7.1, "Viewing and Editing XQuery Files"](#)
- [Section 6.7.2, "Creating Joins and Unions"](#)
- [Section 6.7.3, "Creating If-Then-Else Expressions"](#)
- [Section 6.7.4, "Creating For-Let-Where-Order By-Return \(FLWOR\) Expressions"](#)
- [Section 6.7.5, "Creating Typeswitch Expressions"](#)
- [Section 6.7.6, "Inserting XQuery Functions"](#)
- [Section 6.7.7, "Inserting Expression Variables"](#)
- [Section 6.7.8, "Viewing Schema Properties"](#)

6.7.1 Viewing and Editing XQuery Files

To edit an XQuery file:

1. Select the project containing the XQuery file that you want to edit.
2. Double-click the XQuery file.

Note: If the XQuery file is XQuery 2002-compliant, it opens automatically in the XQuery 2002 Transformation Editor, which has only **Source** and **Test** views. For more information, see [Section 5.2, "Support for XQuery 2002 and 2004."](#)

3. Select the **Source** view.
The XQuery code is displayed. Invalid code is underlined in red.
4. Make the required changes.

Note: If necessary, you can delete the data transformations in the Source view by deleting all the code within the function except the root element.

5. Save the changes.

6.7.2 Creating Joins and Unions

The Constraints view in the XQuery transformation perspective allows you to constrain or manipulate the relationship between source and target repeating elements.

The following **Constraint Type** options are available in the **Constraints** view:

- **Repeatability/Join** option

When you create transformations between repeating elements, **for** loops are generated to iterate through the repeating elements. You can limit or constrain the target repeating elements by adding **where** clauses to the **for** loops in the **Where Clause** pane of the **Constraints** view.

You can create complex conditions (joined by OR or AND operators) for the **where** clause, as shown in the following code example:

```
((data($PurchaseOrderDoc/partId) > 200 and data($PurchaseOrderDoc/partId) <= 400))
```

At run time, the **for** loop iterates over only those repeating elements that fulfill the complex condition.

For a detailed example on using the **Constraints** view, see [Section 7.4, "Creating Repeating-Source to Nonrepeating-Target Transformations."](#)

- **Union**

See [Section 7.3, "Creating Unions."](#)

6.7.3 Creating If-Then-Else Expressions

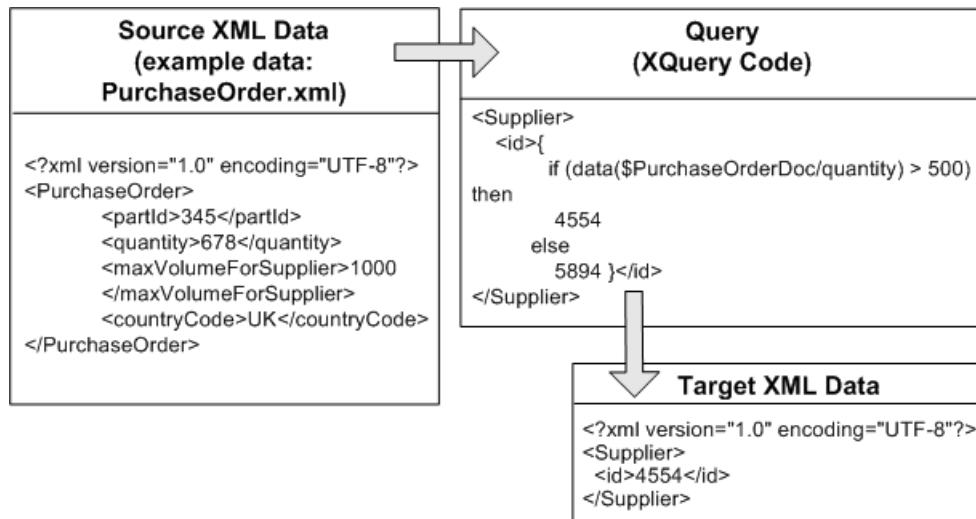
The Target Expression view allows you to create if-then-else expressions.

When a query that contains an if-then-else expression is executed, the conditions that make up the **if** expression are evaluated. Depending on the result, different values are returned for the target node.

Figure 6–6 shows XQuery code that can be used to implement the following logic:

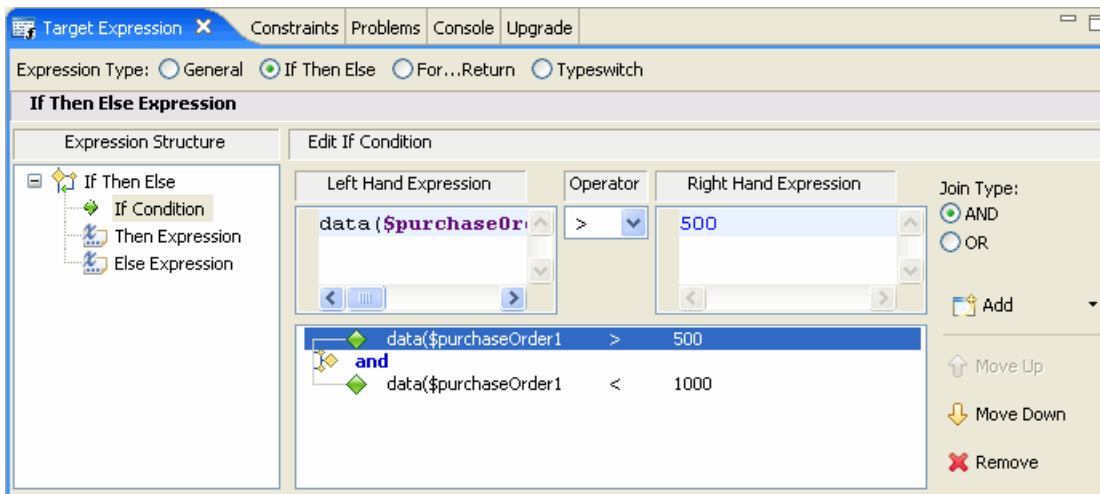
- If the value of the **quantity** source node is more than 500, then return 4554 as the value of the **ID** target node
- If **quantity** is less than or equal to 500, then return 5894 as the **ID**.

Figure 6–6 Example of If-Then-Else Expression



You can add multiple expressions to the **If** condition, as shown in the following figure.

Figure 6–7 If-Then-Else Expression in Target Expression View



You can change the position of a condition by selecting it and then clicking the **Move Up** or **Move Down** button. You can also remove a condition by selecting it and then clicking **Remove**.

Note: In the **Edit If Condition** pane, even if you remove all the expressions by using the **Remove** button, the if-then-else expression is not removed entirely in the Source view. The expressions associated with the **if** condition are removed, but the **then** and **else** expressions are retained, as shown in the following listing.

```
<ns0:partId>
{
  if (fn:boolean("true"))
    then 4554
    else 5894
}
</ns0:partId>
```

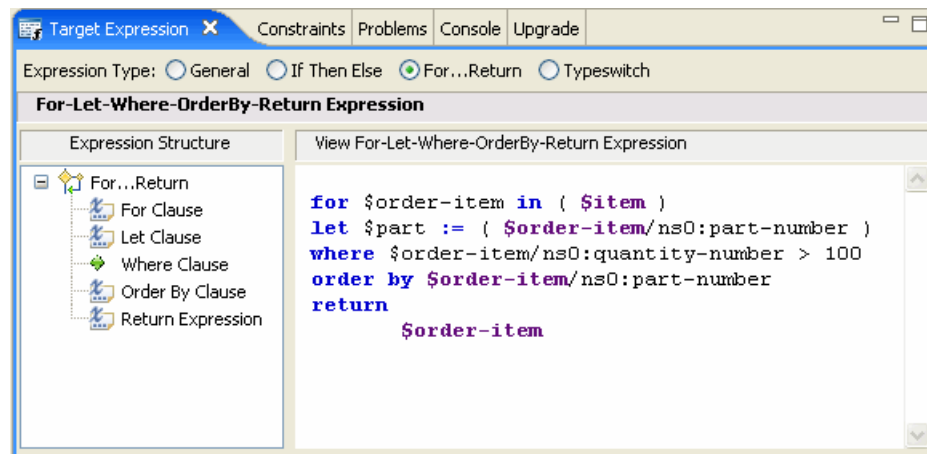
The XQuery always returns the **then** expression. The **else** expression is retained in the code so that you can reuse it in the future, if required.

For more information, see [Section 7.6, "Creating Nested If-Then-Else Expressions."](#)

6.7.4 Creating For-Let-Where-Order By-Return (FLWOR) Expressions

The Target Expression view allows you to create FLWOR expressions, as shown in the following figure.

Figure 6–8 FLWOR Expression



The following table describes the components of FLWOR expressions.

Table 6–1 Clauses of FLWOR Expressions

Component	Description	Optional or Mandatory	Allowed Nested Expressions
For clause	The For clause iterates over a sequence of input items and returns a value for each item.	At least one For or Let clause	<ul style="list-style-type: none"> ■ If-Then-Else ■ FLWOR ■ Typeswitch

Table 6–1 (Cont.) Clauses of FLWOR Expressions

Component	Description	Optional or Mandatory	Allowed Nested Expressions
Let clause	The Let clause declares a variable and assigns a value to the variable.	At least one For or Let clause	<ul style="list-style-type: none"> ■ If-Then-Else ■ FLWOR ■ Typeswitch
Where clause	The Where clause specifies the basis for filtering input data. It is similar to the If clause.	Optional	
Order By clause	The Order By clause specifies the basis for sorting the output of the query.	Optional	<ul style="list-style-type: none"> ■ If-Then-Else ■ FLWOR ■ Typeswitch
Return expression	The Return expression defines the output of the query.	Mandatory (only one)	<ul style="list-style-type: none"> ■ If-Then-Else ■ FLWOR ■ Typeswitch

You can insert nested expressions under For, Let, Order By, and Return by right-clicking on them and selecting the required expression from the menu.

Note: In the Design view, if you create a link between repeating elements, a For...Return expression (implicit FLWOR expression) is generated automatically in the **Source** view. You can add Where clauses to this FLWOR expression by using the **Constraints** view, but you cannot add Let and Order By clauses. Implicit FLWOR expressions are not shown in the **Target Expression** view.

For more information, see [Section 7.7, "Creating FLWOR Expressions."](#)

6.7.5 Creating Typeswitch Expressions

Typeswitch expressions may be required in the following situations:

- When an XML schema contains a <choice> element, which allows only one of the elements defined in the <choice> declaration to be present in the containing element, you can use a typeswitch expression to determine the type of the <choice> element that is present in the source XML file and, accordingly, return a value.
- When an XML schema contains a substitution group, which allows one element to be substituted for another, you can use a typeswitch expression to determine the type of the element that is actually in the source XML file and, accordingly, return a value.

The **Target Expression** view lets you create typeswitch expressions, as shown in the following figure.

Figure 6–9 Typeswitch Expression

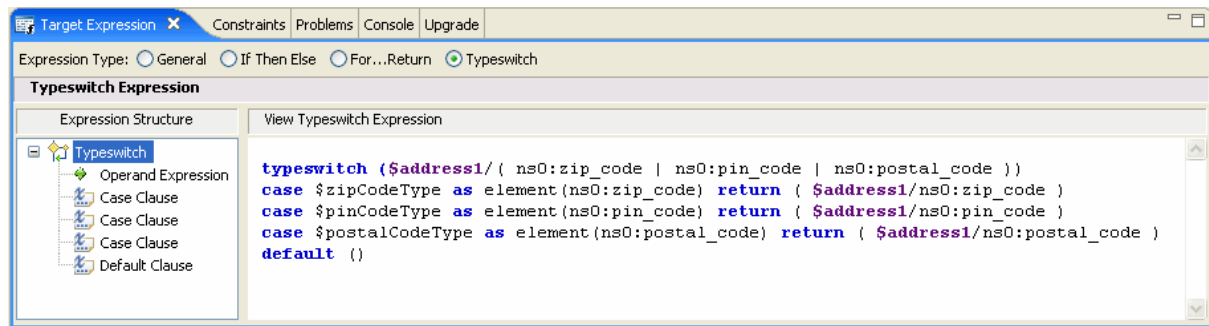


Table 6–2 describes the components of typeswitch expressions.

Table 6–2 Clauses of Typeswitch Expressions

Component	Description	Optional or Mandatory	Allowed Nested Expressions
Operand expression	The Operand is the expression for which the type is to be evaluated.	Mandatory (only one)	
Case clause	Each Case clause specifies the name of the type to be evaluated and the expression to be returned if the evaluation is true.	Mandatory (at least one)	<ul style="list-style-type: none"> ■ If-Then-Else ■ FLWOR ■ Typeswitch
Default clause	The Default clause specifies the return expression that must be used if the value of the operand expression matches none of the types specified in the Case clauses.	Mandatory (only one)	<ul style="list-style-type: none"> ■ If-Then-Else ■ FLWOR ■ Typeswitch

6.7.6 Inserting XQuery Functions

A set of standard W3C XQuery functions and operators is provided in XQuery Mapper. You can add standard XQuery or user-defined functions in XQuery files and data transformation files. For example, you can, use the **upper-case** XQuery `String` function to convert the characters in an XML String value to uppercase characters.

Note: For more information about XQuery and XPath functions and operators (W3C Working Draft 23 July 2004), see <http://www.w3.org/TR/2004/WD-xpath-functions-20040723/>.

To Insert an XQuery Function:

1. Switch to the XQuery transformation perspective by choosing **Window > Open Perspective > XQuery Transformation**.

The Expression Functions view is part of the XQuery transformation perspective.

2. Open the XQuery file in which you want to insert XQuery functions.
3. In the **Design** view, select or create a link to add the function call.

The link becomes green.

4. Select the **Target Expression** view.

If the Target Expression view is not visible, choose **Window > Show View > Target Expression**.

In the **General Expression** pane, the XQuery code linking the selected target and source node is displayed and is selected. Keep this selected for the next step.

5. Delete the existing XQuery code in the **General Expression** pane.
6. Find the function that you want to insert in the **Expression Functions** view.
For this example, from the **String Functions** folder, select the **upper-case** function, which converts all the characters of the source element to upper case.
7. Drag the **upper-case** function to the **General Expression** pane.
Leave the parameter of the selected function (the **\$string-var** parameter of the **upper-case** function in this example) in the **General Expression** pane selected.

Notes: XQuery functions that are defined by Oracle (example: **trim-left**) are prefixed with **fn-bea:**.

XQuery functions that are not listed in the Expression Functions view but defined in the XQuery specification, can be used with the **fn:** prefix.

8. Select a source parameter for the function using one of the following options:
 - From the **Source** pane of the **Design** view, select a source element, drag it to the **General Expression** pane and drop it over the **\$string-var** parameter.
 - From the **Expression Variables** view, select a source variable, drag it to the **General Expression** pane and drop it over the **\$string-var** parameter.
9. After inserting the required functions and assigning parameters to the functions, click **Apply** in the **General Expression** pane.

6.7.7 Inserting Expression Variables

The variables (and their subelements) that you can use in an XQuery are displayed in the Expression Variables view.

Note: If the **Expression Variables** view is not displayed, choose **Window > Show View > Expression Variables**.

The following types of variables are displayed in the Expression Variables view:

- **Source:** The variables displayed under the **Source** node in the **Expression Variables** view are those that are selected for the transformation in the **Source Types** dialog box of the **New XQuery Transformation** wizard.
- **Structural Link:** The variables displayed under the **Structural Link** node in the **Expression Variables** view are the loop iteration variables that are associated with the XQuery **for** loops generated by structural links.

These variables are in scope for all the subelements of the node that has the structural link.

You can insert expression variables in the following ways:

- **Drag-and-drop**
Drag the variables or their subelements from the **Expression Variables** view, and drop them in the **Constraints** or **Target Expression** view.

- Enter **\$** and choose a variable from the pop-up menu.
 1. Enter the dollar symbol (**\$**) in the required text field. For example, enter **\$** in the **General Expression** pane of the **Target Expression** view.
A pop-up menu containing a list of the available variables is displayed.
 2. Choose the required variable and then enter a forward slash (**/**).
If subelements exist, a pop-menu containing a list of the available subelements is displayed.
 3. Choose the required subelement and then enter a forward slash (**/**).
If further subelements exist, a pop-menu containing a list of the available subelements is displayed.
 4. Repeat the previous step until you finish entering the required variable.

6.7.8 Viewing Schema Properties

While editing an XQuery file in the Design view, you can view the schema properties of nodes in the current transformation in the Properties view, without opening the source and target XSD or MFL files.

To display the **Properties** view, choose **Window > Show View > Properties**.

- If you select an element or attribute in the **Source**, **Target**, or **Expression Variable** view, the associated schema properties are displayed in the Properties view.
- If you select a link, the schema properties of the target and source elements of the link are displayed in the Properties view.
- To deselect a link (including the target and source nodes of the link), click anywhere in the empty area of the pane between the **Source** and **Target** panes of the **Design** view.

Note: To change the schema properties of an element or attribute, edit the corresponding schema file (XSD for XML schema and MFL for non-XML schema).

6.8 Restricting Output of Optional Elements

If the target schema in a data transformation contains an optional element (**minOccurs="0"**), you can design the link to the element such that the element is included in the output XML file only if it contains a value (that is, the element is not empty in the source XML file).

Consider the source XML data in the following listing.

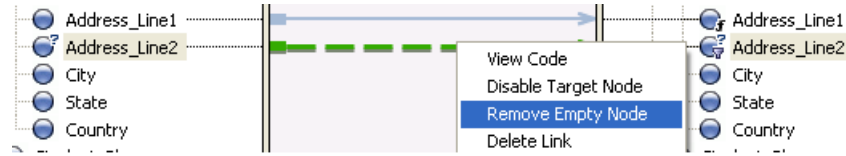
Example 6–1 XML Data with Optional Element

```
<address>
<Address_Line_1>1 Elm Street</Address_Line_1>
<Address_Line_2/>
<City>San Jose</city>
<State>California</State>
<Country>US</Country>
</address>
```

The **Address_Line_2** element is optional and empty. If **Address_Line_2** is mapped to a corresponding element in the target schema, then, by default, the output XML file contains an empty **Address_Line_2** element.

You can restrict output of such optional elements by right-clicking on the link and selecting **Remove Empty Node**, as shown in the following figure.

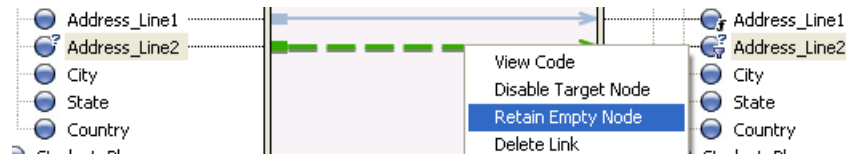
Figure 6–10 Remove Empty Node



The XQuery code underlying the link is enclosed in an if-then-else expression that causes the target element to be produced only if the transformation results in a non-empty value.

You can remove the if-then-else expression by right-clicking on the link and selecting **Retain Empty Node**, as shown in the following figure.

Figure 6–11 Retain Empty Node



Note: The **Remove Empty Node** (or **Retain Empty Node**) option is displayed only when you right-click on a link to an optional target element.

6.9 Testing Data Transformations

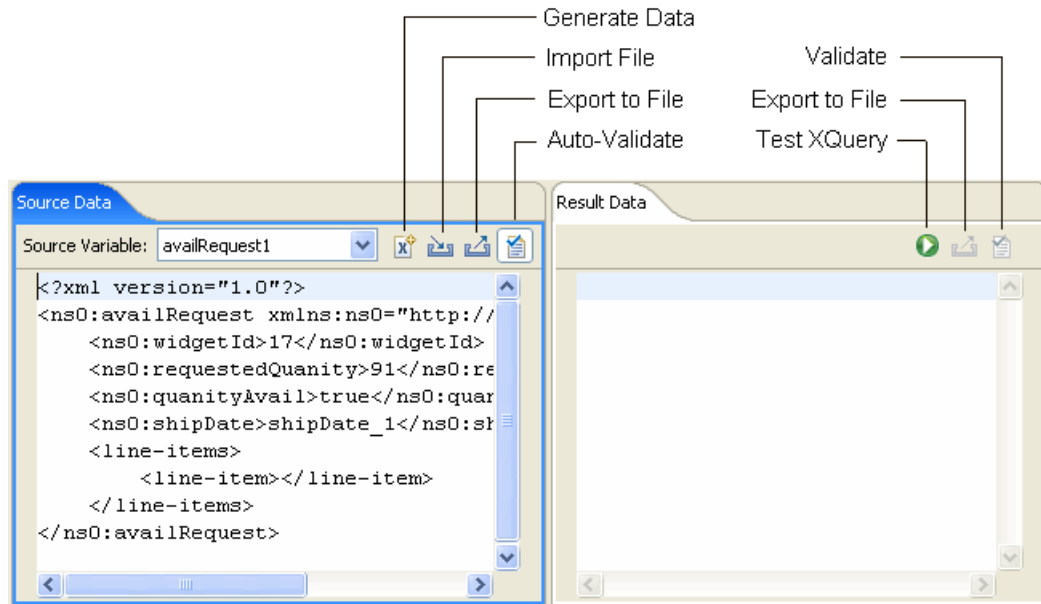
After creating an XQuery transformation in the Design view, you can test whether the expected XML or non-XML output is generated properly in the Test view.

You can use the auto-generated XML files or your own custom XML and non-XML files for testing the transformations.

6.9.1 Features of the Test View

The following figure shows the features of the Test view.

Figure 6–12 Test View



- **Source Variable**

The values available in the **Source Variables** list are based on the source XML schemas of the transformation that you are testing. When you select one of these schemas, an XML file is generated automatically and displayed.

Note: These XML files are not saved automatically; you can save them by clicking the **Export** icon.

To use a custom XML file (instead of the auto-generated XML files) or a non-XML file (such as MFL) for testing the transformation, you can import the file by clicking the **Import** icon.

- **Generate Data**

When you select the Test view, XQuery Mapper generates an initial set of sample data and displays it in the Source Data pane.

If you want to regenerate the sample data, click **Generate Data**. You might, for example, want to start testing afresh with new sample data if edits have resulted in XML data that is no longer valid for the input schema.

If the XQuery that you want to test has multiple source types, you can generate sample data for each source type by selecting the required source type in the **Source Variable** list and then clicking the **Generate Data** icon.

You can also manually edit the generated XML data.

Note: For complex input schemas, the generated XML data may not comply with the schema. Validation errors are underlined in yellow in the Source Data tab. When you place the mouse pointer over an error, details of the error are displayed. You can correct the generated XML data to make it comply with the input schema.

- **Import File**

You can import data from an XML (or non-XML) file and test the transformation using that data.

In addition, if the XQuery that you want to test has multiple source types, you can import sample data for each source type by selecting the required source type in the **Source Variable** list and then clicking the **Import File** icon.

If the **Auto Validate** option is selected, then, when you import data, the data is validated against the associated schema. Errors are underlined in yellow. When you place the mouse pointer over an error, details of the error are displayed.

Note: You can import XML data for global types and local elements, but global types and local elements are not validated and no errors or warnings are reported for invalid data. For more information, see [Section 6.11, "XML Global Elements, Global Types, Local Elements, and Attributes."](#)

- **Export to File:** You can save the data in the **Source Data** pane or the results of the transformation in the **Result Data** pane to an XML file.
- **Test XQuery:** When you select this option, the XQuery is executed on the data in the **Source Data** pane and the result of the transformation is displayed in the Results Data pane.

- **Options for validating at design time**

The **Auto Validate** option in the **Source Data** pane and the **Validate** option in the **Result Data** pane are enabled only if the source parameter or result data is an XML global element.

The validation options are not available for the following data types:

- Typed non-XML

Note: In any case, untyped non-XML (raw) data cannot be used in data transformations.

- XML global type
- XML local element

For more information, see [Section 6.11, "XML Global Elements, Global Types, Local Elements, and Attributes."](#)

Selecting the **Auto Validate** option causes the source data to be validated automatically against the source schema every time the data is changed.

You can use the **Validate** option to validate the result of the transformation against the target schema.

Note: The validation at design time in the Test view is not the same as the schema validation that occurs at run time. Validation at design time does *not* modify the resulting XML document, but it does check for existence of elements and attributes that are defined as required in the schema.

6.9.1.1 Related Topics

[Section 5.2.1, "Restrictions Applicable to the XQuery Test View"](#)

6.10 Graphical Features in Design View

This section provides information to help you use the graphical features of XQuery Mapper and interpret the graphical representations in the Design view of XQuery Mapper.

6.10.1 Right-Click Menu Options

The following table lists the options available in the Design view of XQuery Mapper when you right-click on a link or element.

Table 6–3 *Right-Click Menu Options*

Menu Option	Appears When You ...	Result ...
View Code	Right-click on any link or target element	The view switches to the Source view, and the XQuery code for the link is selected.
Create Constant	Right-click on any simple-type target element	Lets you assign a constant value to the target element.
Disable Target Node	Right-click on any link or target element	Blocking XQuery code is added around the XQuery code of the selected link. The blocking code prevents the XQuery code for the link from being executed at run time. Note: The Disable Target Node menu option is not available for the root node of the target type.
Enable Target Node	Right-click on a disabled link or target element	The blocking XQuery code is removed from around the selected link so that, at run time, the XQuery code for the link is executed.
Remove Empty Node	Right-click on a link to an optional target element.	Surrounds the link with an if-then-else expression that causes the target element to be produced only if the transformation results in a non-empty value.
Retain Empty Node	Right-click on a link (to an optional target element) for which you selected the Remove Empty Node option earlier.	Removes the if-then-else expression that causes the target element to be produced only if the transformation results in a non-empty value.
Redefine Wildcard Node	Right-click on a wildcard (any type) source or target element	Lets you define a specific data type for a wildcard element. Note: After you redefine a wildcard node (that is, define a specific data type), you cannot use Ctrl+Z to revert to the any type.
Revert to Wildcard Node	Right-click on a wildcard (any type) source or target element, for which you defined a specific data type earlier by using the Redefine Wildcard Node option.	Changes the data type of the element from the specific type that you defined earlier to wildcard (any type).

Table 6–3 (Cont.) Right-Click Menu Options

Menu Option	Appears When You ...	Result ...
Induce Map	Right-click on a structural link.	Data links or data structural links are created between the child nodes of the selected structural link if source and target child elements of the link are the same subschema type. Note: For the Induce Map option to create child links, the target and source child elements must have the same name and data type, and must be in the same order.
Delete Link	Right-click on any link.	<ul style="list-style-type: none"> ▪ Design view: The link is deleted. ▪ Source view: The XQuery code underlying the link is deleted.
Delete All Links	Right-click anywhere in the empty pane between the Source and Target panes.	<ul style="list-style-type: none"> ▪ Design view: The lines representing the transformation between source and target elements/attributes are deleted. ▪ Source view: The generated XQuery code is deleted. <p>Note: Right-clicking anywhere in the empty pane between the Source and Target panes causes all the nodes to be deselected.</p>

6.10.2 Link Patterns

Links in XQuery Mapper are shown in different colors and patterns to help you distinguish easily between different link types. The following table describes the graphical representation of the links that you create in XQuery Mapper.

Table 6–4 Link Patterns











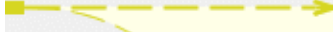

Link Type	Description	Representation
Data Link	A link that converts the value of the source node directly to the value of the target node.	Link not selected 
		Link selected 
Implied Link	A data link for which you modified the underlying XQuery code. A link for which the XQuery code cannot be interpreted by the XQuery Mapper. Examples: <ul style="list-style-type: none"> ▪ A link for which you inserted the fn:upper-case XQuery function by using the General Expression section of the Target Expression view. ▪ The data links generated between a second set of child nodes when a union constraint is applied to a set of two structural links. The child nodes must be of the same subschema. 	Link not selected 
		Link selected 

Table 6–4 (Cont.) Link Patterns

Link Type	Description	Representation
Structural Link	A link between two parent structures and that does not map data directly.	Link not selected 
		Link selected 
Data Structural Link	A data structural link is a combination of a data link and a structural link. Example: A link between the optional child nodes of a repeating element.	Link not selected 
		Link selected 
Constraint Link	A link that constrains or limits the resulting data of a join between source parent structures. The constraint link is created with two source nodes. Example: A join between two source repeating elements to return the data only when the values of a particular source element are equal to each other.	Link not selected 
		Link selected 
Copy Link	A link between two identical schema substructures. At run time, the source data is copied directly, as a block, to the target data. A copy link is also generated when you map an untyped XML node to a typed XML complex-type node.	Link not selected 
		Link selected 

6.10.3 Link Colors

When you drag a node from the **Source** pane to the **Target** pane, a temporary link (dashed line) appears between the two nodes. The color of the line changes depending on the compatibility between the source and target nodes.

- **Red:** The link cannot be created between the source node and the target node because the data type of the target node cannot be converted to the data type of the source node. For example, a node of XML String data type cannot be converted

to an XML repeating node. An error message is displayed when you drag the source node over the target node.

- **Orange:** The link can be created between the source and target nodes, but the data types are not completely compatible. A warning message describing the incompatibility or any necessary conversion is displayed when you drag the source node over the target node.
- **Green:** The link can be created between the source and target nodes. The data type of the target node is compatible with the data type of the source node.

When you finish creating the link, a dotted or dashed line (depending on the source and target nodes) is displayed.

6.11 XML Global Elements, Global Types, Local Elements, and Attributes

An XML schema type or element is considered global if it is a direct child of the **schema** element and local if it is not a direct child of the **schema** element (that is, the element is nested within another element).

The following example XML schema illustrates this difference.

Example 6–2 XML Schema with Global and Local Elements



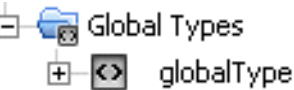
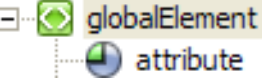
```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.acme.org/globalExample"
xmlns="http://www.acme.org/globalExample"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="globalElement">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="localElement"
          minOccurs="1" maxOccurs="1"
          type="xs:string" />
      </xs:sequence>
      <xs:attribute name="attribute"
        type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>
  <xs:complexType name="globalType">
    <xs:sequence>
      <xs:element name="anotherLocalElement"
        minOccurs="0" maxOccurs="unbounded"
        type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

In the preceding XML schema, **globalElement** is global because it is a direct child of the **schema** element, whereas **localElement** is local because it is a child of **globalElement**.

You can also define a global type, as shown by the **globalType** element toward the end of the preceding XML schema. While you can have only one global element in an XML schema, you can declare many elements (with different names) of the same global type in a single XML schema.

Table 6–5 shows the graphical representations of the different XML components in XQuery Mapper.

Table 6–5 Graphical Representation of XML Components

Component	Graphical Representation	Name in Preceding Example XML Schema
Global Element	 globalElement	globalElement
Local Element	 localElement	localElement
Global Type	 Global Types globalType	globalType
Attribute	 globalElement attribute	attribute defined for globalElement

Examples: Data Transformation Using XQuery Mapper

The examples described here are based on the sample project that is included in the product. For information about opening the sample project, see [Section 6.2, "Importing the XQuery Mapper Sample Project."](#)

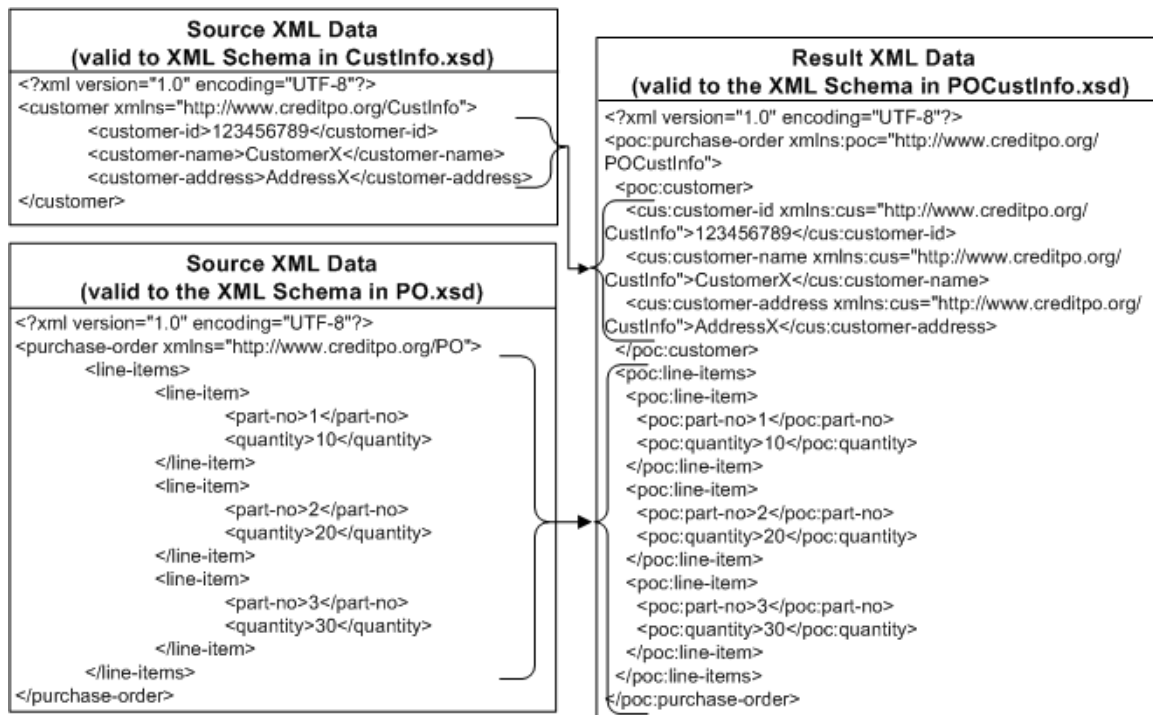
Examples are provided for the following scenarios:

- [Section 7.1, "Combining Data from Different Schemas"](#)
- [Section 7.2, "Mapping Repeating Elements and Creating Joins"](#)
- [Section 7.3, "Creating Unions"](#)
- [Section 7.4, "Creating Repeating-Source to Nonrepeating-Target Transformations"](#)
- [Section 7.5, "Creating Nonrepeating-Source to Repeating-Target Transformation"](#)
- [Section 7.6, "Creating Nested If-Then-Else Expressions"](#)
- [Section 7.7, "Creating FLWOR Expressions"](#)
- [Section 7.8, "Using Recursive Schemas"](#)
- [Section 7.9, "Grouping Data by Key Fields"](#)

7.1 Combining Data from Different Schemas

You can use XQuery Mapper to combine content from two different schemas, as shown in the following figure.

Figure 7-1 Combining Data From Different Schemas



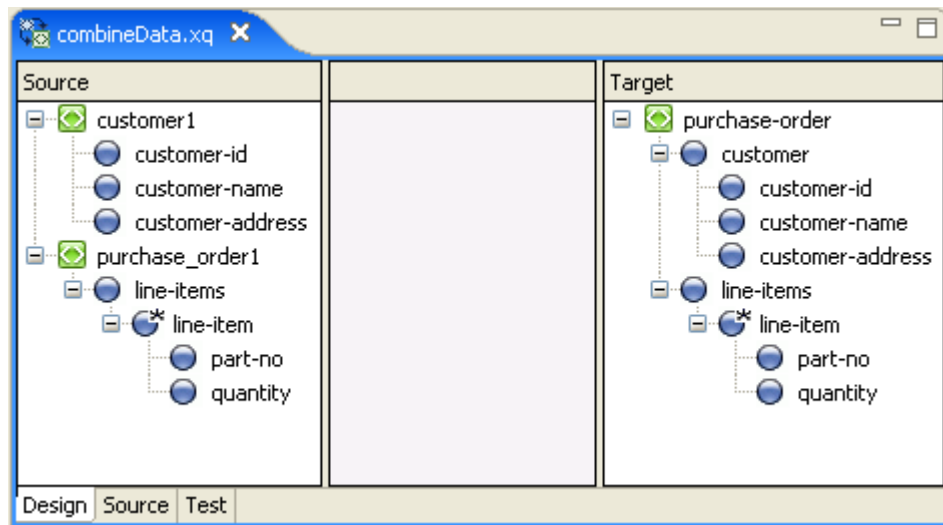
In this example, customer data (valid against **CustInfo.xsd**) is merged with a repeating element **line-items** (valid against **PO.xsd**) to form a single XML document that is valid against the **POCustInfo.xsd** schema.

1. Launch Eclipse and open the sample project.
 - For information about launching Eclipse, see [Section 6.1, "Launching XQuery Mapper."](#)
 - For information about opening the sample project, see [Section 6.2, "Importing the XQuery Mapper Sample Project."](#)
2. Right-click the **XQuery Transformations** folder.
3. Choose **New > XQuery Transformation**.
4. Verify the name of the parent folder. For this example, the parent folder is **/XQuery Transformation/XQueryTransformations**.
5. Enter **combineData** as the file name and click **Next**.
6. Select the following source elements and click **Next**:
 - **CustInfo.xsd\customer**
 - **PO.xsd\purchase-order**
7. Select **POCustInfo.xsd\purchase-order** as the target element and click **Finish**.

The **combineData.xq** file is created in the **/XQuery Transformation/XQueryTransformations** folder.

The source and target elements that you selected are displayed in the Design view, as shown in the following figure.

Figure 7-2 Design View of XQuery Transformation



8. Create links between the following source and target elements by dragging elements from the **Source** pane to the **Target** pane.

Source Element	Link Image	Target Element
customer1		purchase-order\customer
customer1\customer-id		purchase-order\customer\customer-id
customer1\customer-name		purchase-order\customer\customer-name
customer1\customer-address		purchase-order\customer\customer-address
purchase_order1\line-items\line-item		purchase-order\line-items\line-item
purchase_order1\line-items\line-item\part-no		purchase-order\line-items\line-item\part-no
purchase_order1\line-items\line-item\quantity		purchase-order\line-items\line-item\quantity

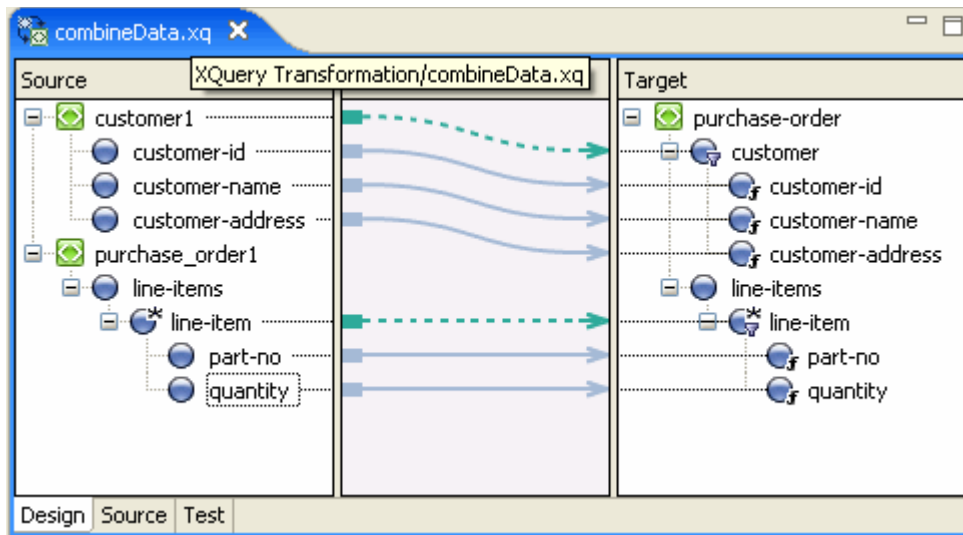
Note: Dotted lines represent **Structural** links, which are created between parent structures and do not map data directly.

Solid lines represent **Data** links, which convert the value of the source node directly to the value of the target node.

For more information, see [Section 6.10, "Graphical Features in Design View."](#)

The links between the **Source** and **Target** elements are displayed, as shown in the following figure.

Figure 7-3 Data Transformation in Design View



9. Save the changes.

For information about testing XQuery transformations, see [Section 6.9, "Testing Data Transformations."](#)

7.2 Mapping Repeating Elements and Creating Joins

You can join data from XML files that are valid against different schemas (in this example, **PriceQuote.xsd**, **AvailableQuote.xsd**, and **taxrate.xsd**), and create an XML file that is valid against a single schema: **Quote.xsd**.

This example includes the following steps:

- [Section 7.2.1, "Step 1. Create an XQuery File"](#)
- [Section 7.2.2, "Step 2. Add a Constraint"](#)
- [Section 7.2.3, "Step 3. Add Data to Return Element"](#)
- [Section 7.2.4, "Step 4. Add Function to Calculate Value of Quote"](#)
- [Section 7.2.5, "Step 5. Add a Constraint with Multiple Conditions"](#)
- [Section 7.2.6, "Test the XQuery"](#)

7.2.1 Step 1. Create an XQuery File

In this step, we create an XQuery transformation by using the **AvailQuote.xsd**, **PriceQuote.xsd**, and **taxrate.xsd** schemas. Then, we map several **priceQuote** and **availRequest** source elements to corresponding target elements.

1. Launch Eclipse and open the sample project.
 - For information about launching Eclipse, see [Section 6.1, "Launching XQuery Mapper."](#)
 - For information about opening the sample project, see [Section 6.2, "Importing the XQuery Mapper Sample Project."](#)
2. Right-click the **XQuery Transformations** folder.
3. Choose **New > XQuery Transformation**.

4. Verify the name of the parent folder. For this example, the parent folder is **/XQuery Transformation/XQueryTransformations**.
5. Enter **Join** as the file name and click **Next**.
6. Select the following source elements and click **Next**:
 - **PriceQuote.xsd\priceQuote**
 - **PriceQuote.xsd\taxRate**
 - **AvailQuote.xsd\availRequest**
7. Select **Quote.xsd\quote** as the target element and click **Finish**.

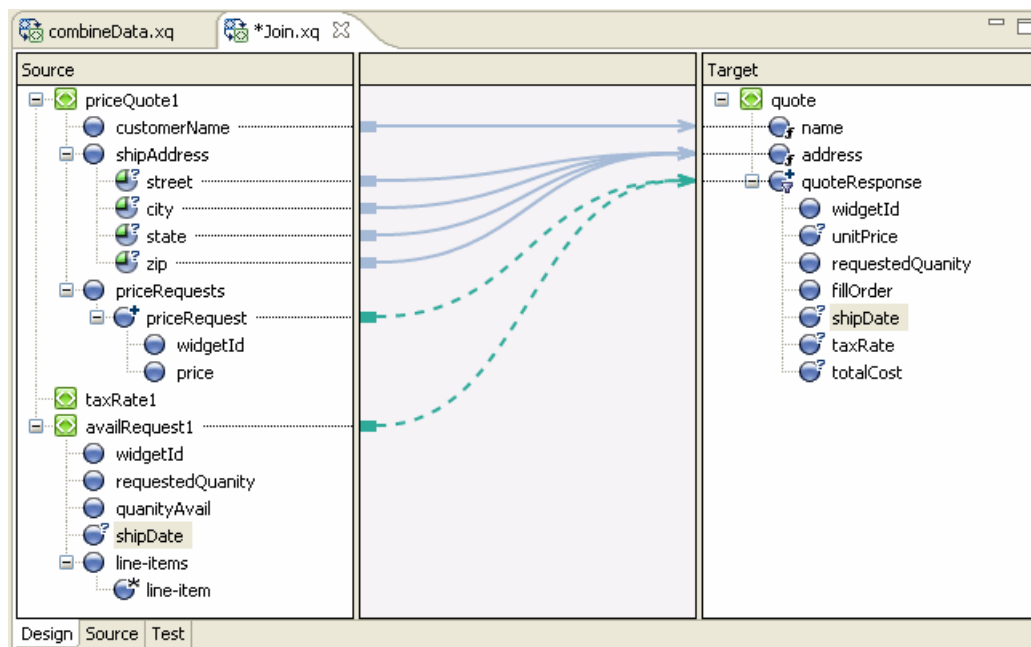
The **Join.xq** file is created in the **XQueryTransformation/XQueryTransformations** folder.

8. Create links between the following source and target elements by dragging elements from the **Source** pane to the **Target** pane.

Source Element	Target Element
priceQuote1\customerName	quote\name
priceQuote1\shipAddress\street	quote\address
priceQuote1\shipAddress\city	quote\address
priceQuote1\shipAddress\state	quote\address
priceQuote1\shipAddress\zip	quote\address
priceQuote1\priceRequests\priceRequest	quote\quoteResponse
availRequest1	quote\quoteResponse

The links are displayed, as shown in the following figure.

Figure 7–4 Data Transformation in Design View



- Save the changes.

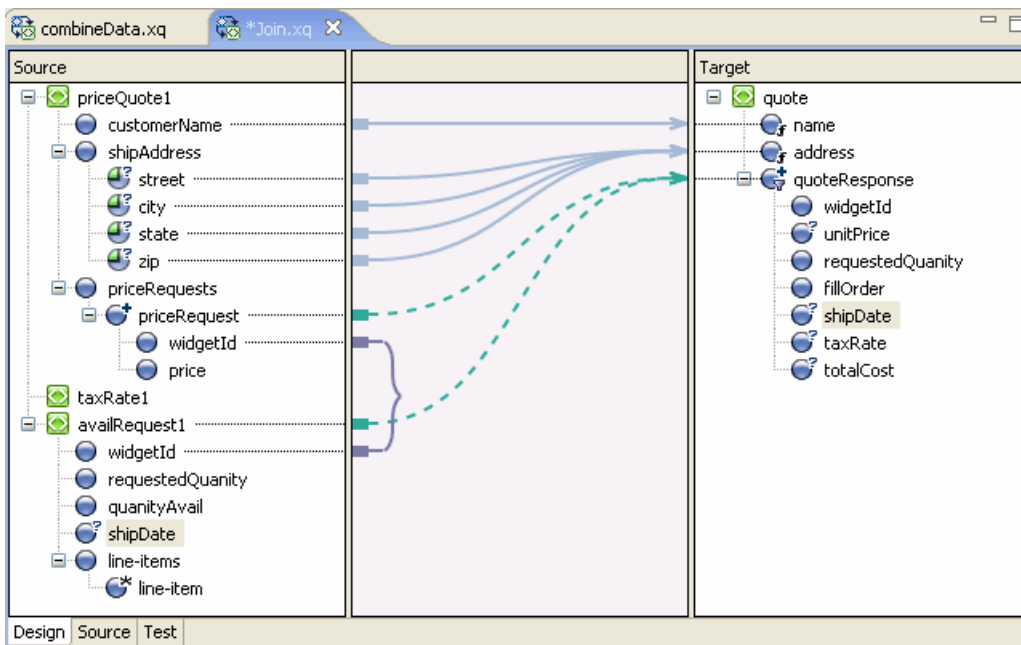
7.2.2 Step 2. Add a Constraint

The **priceQuote/priceRequests** and **availRequest** source elements share the common element **widgetId**. In this step, we add a constraint that if **widgetId** of the **availRequest** schema is equal to **widgetId** of the **priceQuote/priceRequests** element, then the query must return the target repeating element, **quoteResponse**.

- Open **Join.xq** in the Design view.
- Drag the **priceQuote1/priceRequests/priceRequest/widgetId** element from the **Source** pane and drop it on the **availRequest1/widgetId** element of the **Source** pane.

A connecting line appears between the two **widgetId** nodes in the **Source** pane, as shown in the following figure.

Figure 7–5 Adding a Constraint



- Save the changes.
- View the changes in the **Source** view.

The link between the two **widgetId** nodes is represented by a **where** clause within the **for** loop. The **where** clause specifies that the **for** loop must return the result of the expression only if the **where** clause is true. In this example, if **widgetId** of the **availRequest** element is equal to **widgetId** of the **priceRequest** element, the expression returns the XML data specified in the **quoteResponse** element.

Note: You can also view the **where** clause in the Constraints view.

The **quoteResponse** element is currently empty. We add content to the element in the next step.

7.2.3 Step 3. Add Data to Return Element

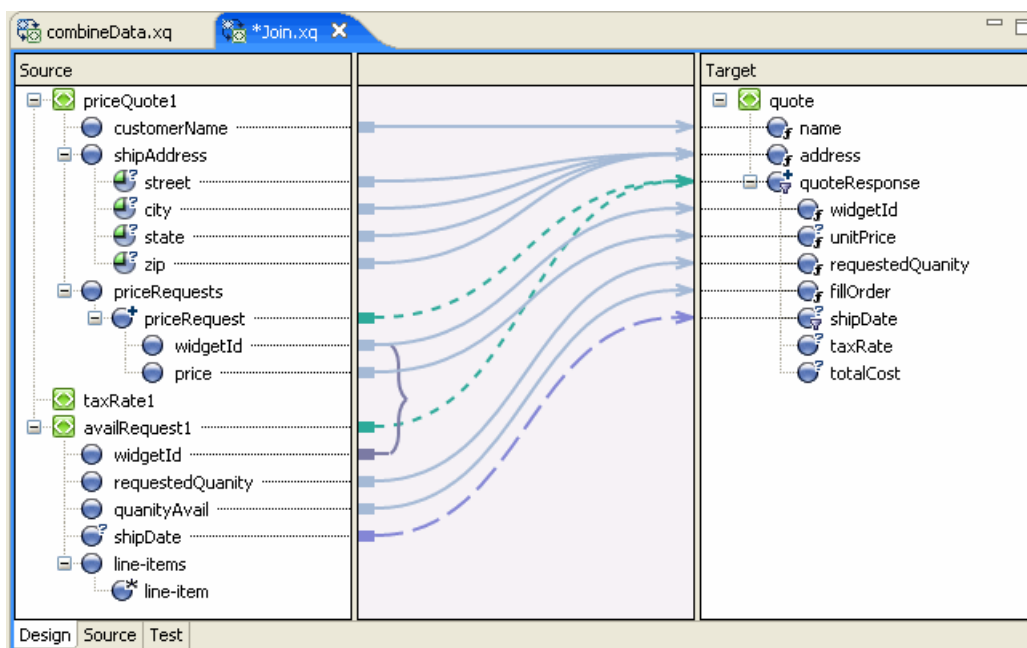
In this step, we add data links in the `quoteResponse` target element.

Open `Join.xq` in the **Design** view and create links between the following source and target elements:

Source Element	Target Element
<code>priceQuote1\priceRequests\priceRequest\widgetid</code>	<code>quote\quoteResponse\widgetid</code>
<code>priceQuote1\priceRequests\priceRequest\price</code>	<code>quote\quoteResponse\unitprice</code>
<code>availRequest1\requestedQuantity</code>	<code>quote\quoteResponse\requestedQuantity</code>
<code>availRequest1\quantityAvail</code>	<code>quote\quoteResponse\fillOrder</code>
<code>availRequest1\shipDate</code>	<code>quote\quoteResponse\shipDate</code>

The links between the **Source** and **Target** elements are displayed, as shown in the following figure.

Figure 7-6 Adding Data in the QuoteResponse Element



7.2.4 Step 4. Add Function to Calculate Value of Quote

In this step, we add a function to calculate the total value of the quote.

1. Open the `Join.xq` file in **Source** view.
2. Insert the following function declaration at any point in the source code between the namespace declarations and the `Join` function call. You can, for example, insert it just before the `Join` function declaration.

Example 7-1 calculateTotalPrice Function

```
declare function xf:calculateTotalPrice(
  $taxRate as xs:float,
```

```
    $quantity as xs:float,  
    $price as xs:float)  
  as xs:float {  
    let $taxQuantity := ($taxRate * $quantity)  
    let $totalTax := ($taxQuantity * $price)  
    let $costNoTax := ($quantity * $price)  
    let $totalCost := ($totalTax + $costNoTax)  
    return $totalCost  
  };
```

3. Switch to the **Design** view.

Note: **Join.xq** now includes two function declarations: `calculateTotalPrice` and **Join**. When more than one function exists in an XQuery file, the function with the same name as the XQ file is rendered in the Design view. In this case, the **Join** function is displayed in the Design view.

4. In the Target pane, select the **totalCost** node. Keep it selected for the next step.
5. Select the **Target Expression** view and select the **General** option.
6. Insert the following code in the General Expression pane.

```
xf:calculateTotalPrice($taxRate1,$availRequest/ns1:requestedQuantity,$priceRequest/ns0:price)
```

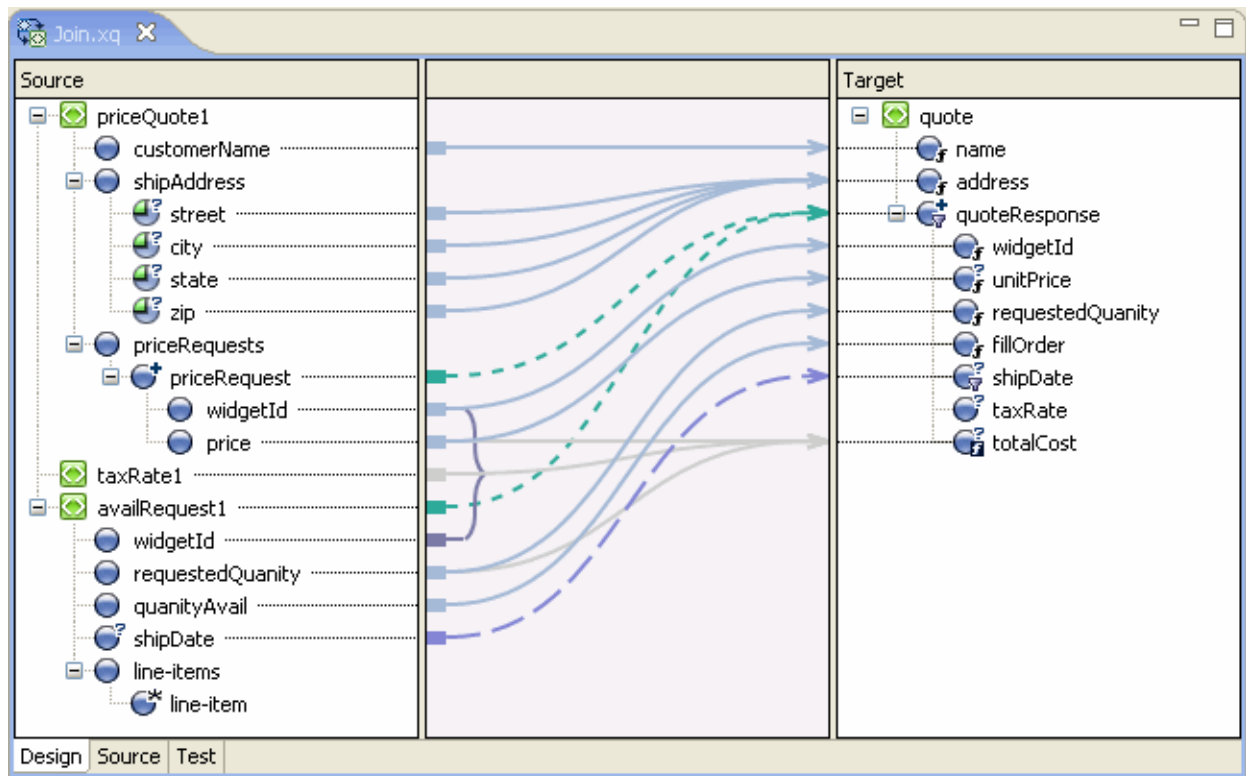
7. Click **Apply**.

The expression is added to the **totalCost** element in the XQuery.

8. Save the changes.

The **Design** view shows the calculation for the **totalCost** target element.

Figure 7-7 totalCost Calculation in Design View



7.2.5 Step 5. Add a Constraint with Multiple Conditions

You can create a constraint by using the **Where Clause** pane of the **Constraints** view to limit the target repeating elements that are returned by the XQuery. At run time, the **for** loop in the XQuery iterates over only those repeating elements that satisfy the **where** clause.

In this step, we add another condition (resulting in a complex condition) to the **where** clause of the **for** loop to further limit the data returned by the **for** loop.

1. Open the **join.xq** file.
2. In the **Design** view, select the link between the **availRequest1** source element and the **quote\quoteResponse** target element.

The single condition that makes up the **where** clause is displayed in **Where Clause** pane of the **Constraints** view.

```
data($priceRequest/ns0:widgetId) = data($availRequest/ns1:widgetId)
```

3. Drag the **availRequest1/requestedQuantity** element from the **Source** pane and drop it in the **Left Hand Expression** area of the **Where Clause** pane.

The left hand expression of the where clause is created as follows:

```
data($availRequest/ns1:requestedQuantity)
```

4. Select the **<** operator.
5. Remove the text in the **Right Hand Expression** area of the **Where Clause** pane, and enter **"50"**.

Note: Enter the number 50 within quotation marks ("**50**", not **50**).

6. From the **Join Type** field select the **AND** option.

The **Join Type** determines how the conditions that make up the **where** clause are evaluated at run time.

7. Click **Add**. The second condition is added to the **where** clause of the **for** loop.
8. Save the changes.

This step completes the creation of the following **where** clause.

Example 7–2 Where Clause

```
where (data($availRequest/ns1:widgetId) = data($priceRequest/ns0:widgetId)
and data($availRequest/ns1:requestedQuantity) < "50")
```

7.2.6 Test the XQuery

Perform the following steps to verify that the XQuery works when both the conditions of the **where** clause you created are satisfied.

1. Switch to the **Test** view.
2. In the **Source Data** pane of the **Test** view, select **priceQuote** in the **Source Variable** field, and click the **Generate Data** icon.
3. Note the value of the **widgetId** element in the test XML data.

```
<ns0:widgetId>value</ns0:widgetId>
```
4. In the **Source Data** pane, select **availRequest** in the **Source Variable** field, and click the **Generate Data** icon.
5. Edit the value the value of the **widgetId** element in the test XML data to match the value displayed in the **priceQuote** test XML data.

```
<ns0:widgetId>value</ns0:widgetId>
```
6. Locate the **requestedQuantity** element and edit the value to a number less than 50.

For example: `<ns0:requestedQuantity>25</ns0:requestedQuantity>`
7. In the **Result Data** pane, click the **Test XQuery** icon and view the results of the XQuery.

7.3 Creating Unions

In this example, we use the **Union** option in the **Constraints** view to construct an XQuery that maps data of the same type into larger sets of data.

1. Launch Eclipse and open the sample project.
 - For information about launching Eclipse, see [Section 6.1, "Launching XQuery Mapper."](#)
 - For information about opening the sample project, see [Section 6.2, "Importing the XQuery Mapper Sample Project."](#)
2. Right-click the **XQuery Transformations** folder.

3. Choose **New > XQuery Transformation**.
4. Verify the name of the parent folder. For this example, the parent folder is **/XQuery Transformation/XQueryTransformations**.
5. Enter **union** as the file name and click **Next**.
6. In the **Source Types** dialog box, select **PO.xsd\purchase-order** twice, and then click **Next**.

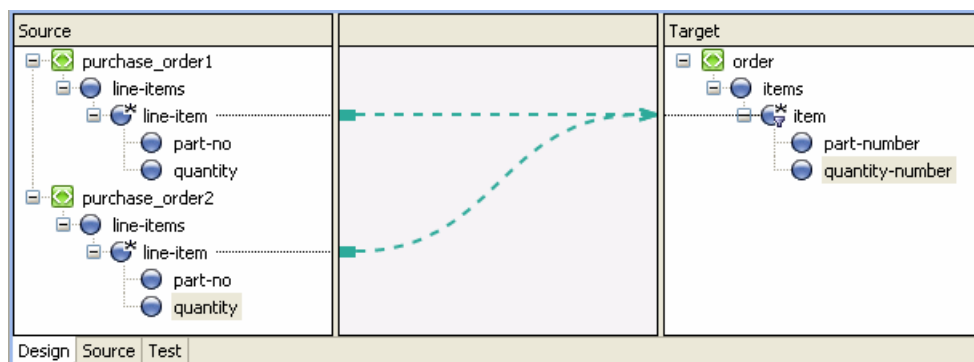
Note: To add an element more than once, you must change the parameter name.

7. Select **Order.xsd\order** as the target type and then click **Finish**.
The **union.xq** file is created.
8. Create links between the following source and target elements:

Source Element	Target Element
purchase-order1\line-items\line-item	order\items\item
purchase-order2\line-items\line-item	order\items\item

The following figure shows how the links appear in the **Design** view.

Figure 7–8 Creating a Union



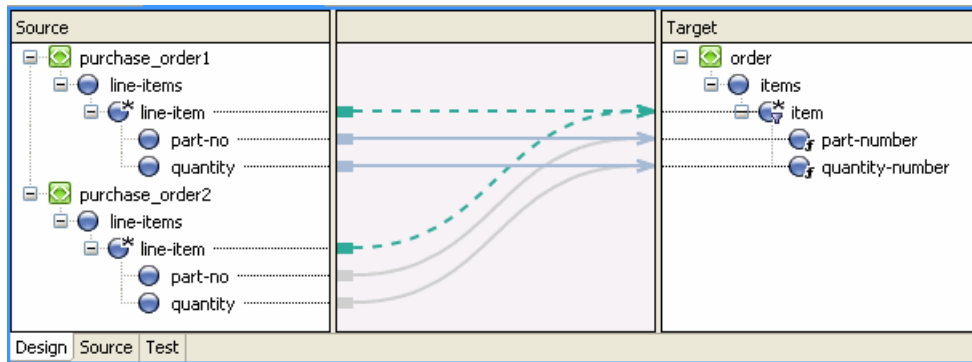
9. Select the link between the **\$purchase-order1/line-items/line-item** source element and the **order/items/item** target element.
10. In the **Constraint Type** pane of the **Constraints** view, select **Union**.
11. Create links between the following source and target elements:

Source Element	Target Element
purchase-order1\line-items\part-no	order\items\item\part-number
purchase-order1\line-items\quantity	order\items\item\quantity-number

Note: When you want to create links between source and target elements of the same name, you can use the **Induce Map** option instead of creating the links manually. For more information, see [Section 6.10.1, "Right-Click Menu Options."](#)

Since the two structural links have a union constraint, a set of implied data links between the second set of subelements is generated as shown in [Figure 7–9](#). The gray lines represent implied links that were created because **Union** was selected as the constraint type.

Figure 7–9 *Creating Implied Links*



12. Save the changes.

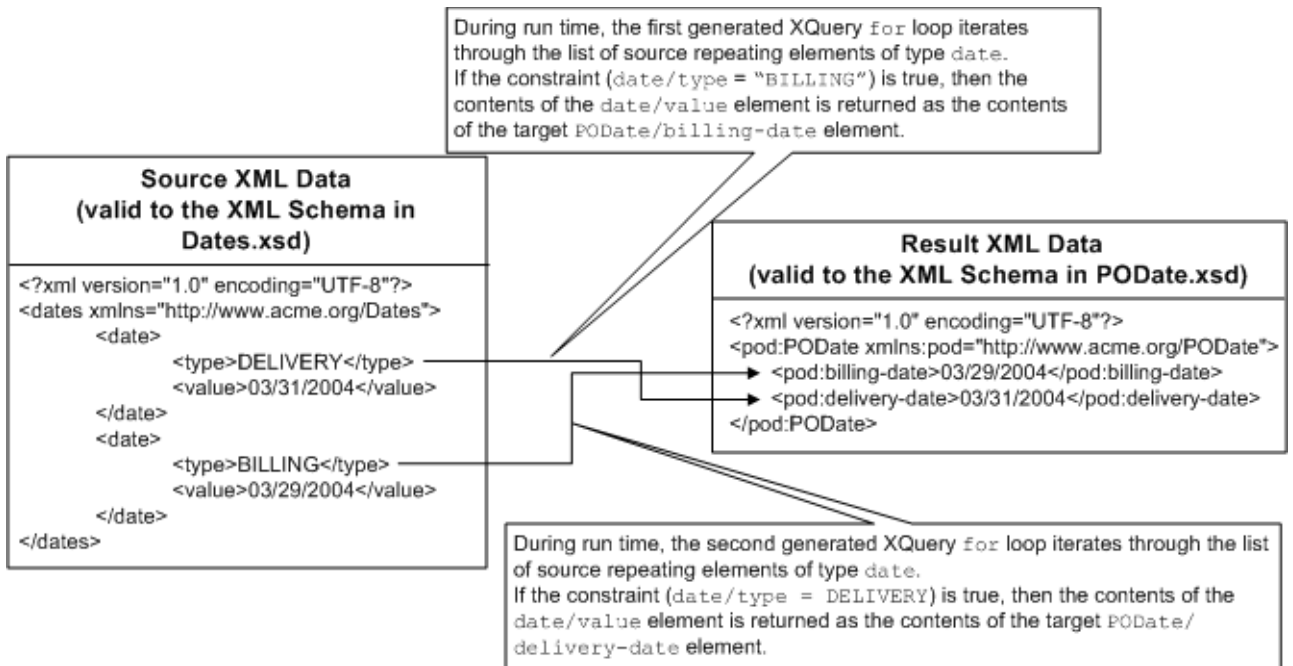
For information about testing XQuery transformations, see [Section 6.9, "Testing Data Transformations."](#)

7.4 Creating Repeating-Source to Nonrepeating-Target Transformations

In this example, we map a repeating source XML element to a nonrepeating target XML element.

The following figure depicts the transformations that we create in this example.

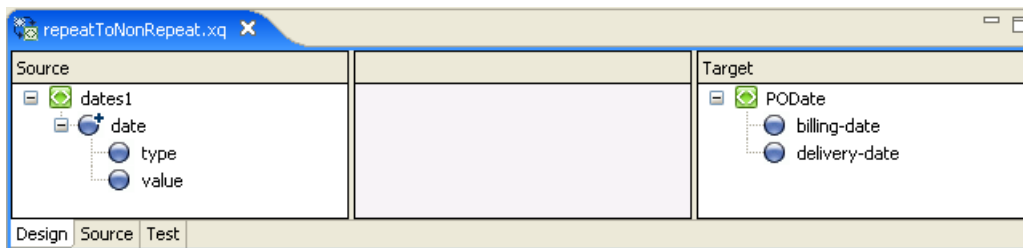
Figure 7–10 Repeating-Source-to–Nonrepeating Target Transformation



1. Launch Eclipse and open the sample project.
 - For information about launching Eclipse, see [Section 6.1, "Launching XQuery Mapper."](#)
 - For information about opening the sample project, see [Section 6.2, "Importing the XQuery Mapper Sample Project."](#)
2. Right-click the **XQuery Transformations** folder.
3. Choose **New > XQuery Transformation**.
4. Verify the name of the parent folder. For this example, the parent folder is **/XQuery Transformation/XQueryTransformations**.
5. Enter **repeatToNonRepeat** as the file name and click **Next**.
6. Select **Dates.xsd\dates** as the source schema and click **Next**.
7. Select **PODate.xsd\PODate** as the target schema and click **Finish**.

The **repeatToNonRepeat.xq** file is created and displayed, as shown in the following figure.

Figure 7–11 Repeating-Source-to–Nonrepeating-Target Data Transformation



8. Create a link between the **dates1/date** repeating element in the **Source** pane and the **PODate/billing-date** element in the **Target** pane.

Keep this link selected for the next step.

9. Select the **Constraints** view.
10. Drag the **dates1/date/type** element from the **Source** pane to the **Left Hand Expression** area of the **Where Clause** pane in the **Constraints** view.
11. Select the **=** operator.
12. In the **Right Hand Expression** area, enter **"BILLING"** (including the quotation marks), and click **Add**.
13. Create a link between the **dates1/date/value** element in the **Source** pane and the **PODate/billing-date** element in the **Target** pane.

The constraint that you created in the preceding steps specifies that the value of the **dates1/date/type** element in an XML document must be compared with the value "BILLING".

At run time, if the value of the **dates1/date/type** element is "BILLING", the XQuery returns the value of **dates1/date/value** as the value of **billing-date**.

14. Create a link between the **dates1/date** repeating element in the **Source** pane and **PODate/delivery-date** element in the **Target** pane.

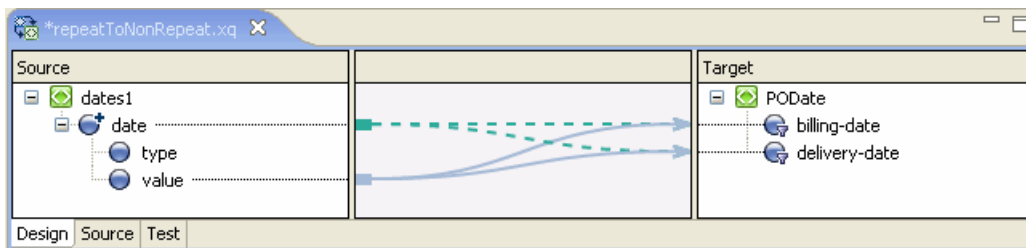
Keep this link selected for the next step.

15. Drag the **dates1/date/type** element from the **Source** pane to the **Left Hand Expression** area of the **Where Clause** pane in the **Constraints** view.
16. Select the **=** operator.
17. In the **Right Hand Expression** area, enter **"DELIVERY"** (including the quotation marks), and click **Add**.
18. Create a link between the **dates1/date/value** repeating element in the **Source** pane and **PODate/delivery-date** element in the **Target** pane.

The constraint created in the preceding steps specifies that the value of the **dates1/date/type** element in an XML document must be compared to the value "DELIVERY".

At run time, if the value of the **dates1/date/type** element is "DELIVERY", the XQuery returns the value of **dates1/date/value** as the value of **delivery-date**.

Figure 7–12 Repeating-Source-to-Nonrepeating-Target Data Transformation



19. Save the changes.

For information about testing XQuery files, see [Section 6.9, "Testing Data Transformations."](#)

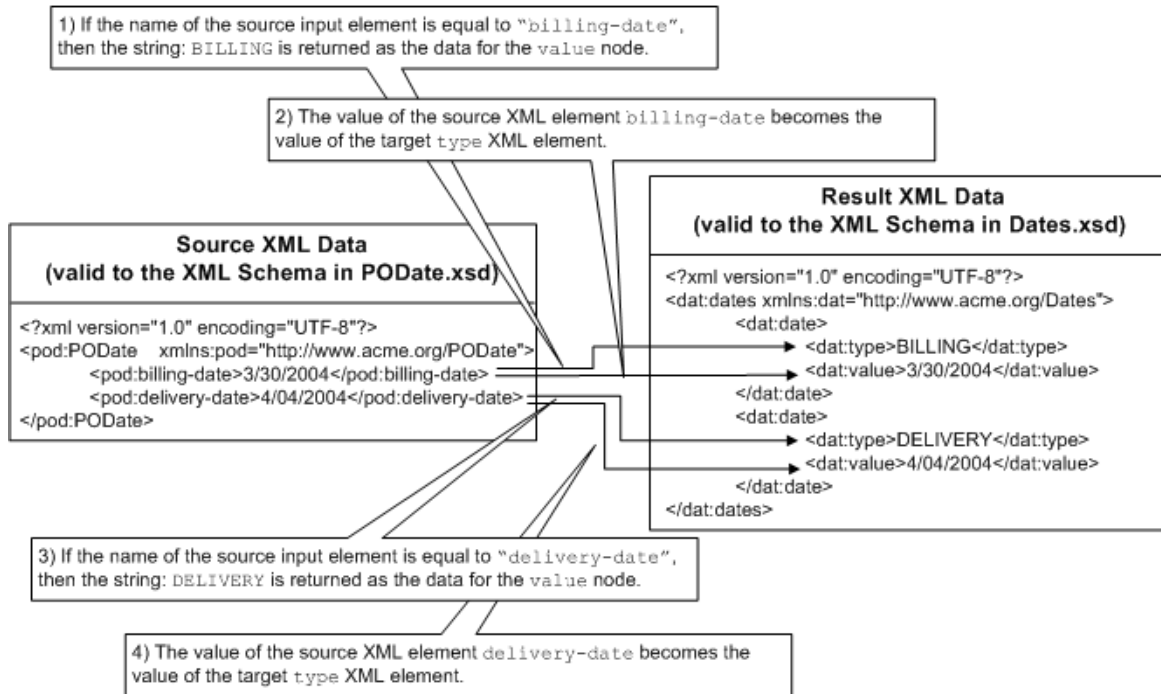
7.5 Creating Nonrepeating-Source to Repeating-Target Transformation

In this example, we map a nonrepeating source element to a repeating target element.

The following figure depicts the transformations that we create in this example.

Figure 7–13 Nonrepeating-Source-to-Repeating-Target Transformation

During run time, the query executes the following actions:



1. Launch Eclipse and open the sample project.
 - For information about launching Eclipse, see [Section 6.1, "Launching XQuery Mapper."](#)
 - For information about opening the sample project, see [Section 6.2, "Importing the XQuery Mapper Sample Project."](#)
2. Right-click the **XQuery Transformations** folder.
3. Choose **New > XQuery Transformation**.
4. Verify the name of the parent folder. For this example, the parent folder is **/XQuery Transformation/XQueryTransformations**.
5. Enter **nonRepeatToRepeat** as the file name and click **Next**.
6. Select **PODate.xsd\PODate** as the source schema and click **Next**.
7. Select **Dates.xsd\dates** as the target schema and click **Finish**.
8. The **nonRepeatToRepeat.xq** file is created and displayed in the Design view.
9. Create links between the following source and target elements:

Source Element	Target Element
pODate1/billing-date	dates/date

Source Element	Target Element
pODate1/delivery-date	dates/date

The following example shows the XQuery code that is generated.

```
<ns1:dates>
{
  for $PODate in $PODate1/ns0:billing-date union $PODate1/ns0:delivery-date
  return
  <ns1:date/>
}
</ns1:dates>
```

At run time, the **for** loop is executed twice. In the first execution, the iteration variable **\$PODate** is equal to the first element in the union **\$PODate1/ns0:billing-date**; in the second execution, **\$PODate** is equal to **\$PODate1/ns0:delivery-date**.

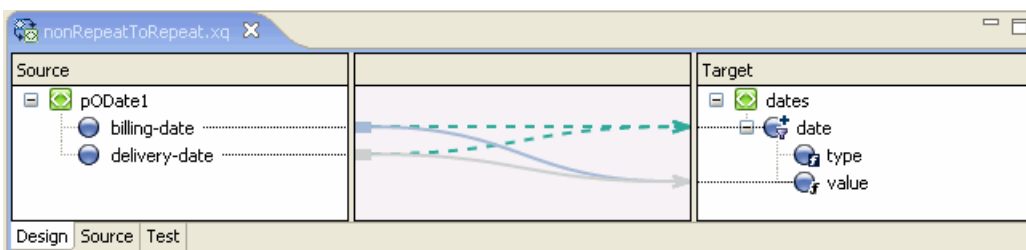
The XQuery returns two empty XML elements with the tag **<ns1:date/>**.

In the following steps, we add the XQuery code to return the billing and delivery dates to the query.

10. Switch to the **Design** view.
11. Create a link between the **pODate1/billing-date** source element and the **dates/value** target element in the Target pane.

Two data links are created, as shown in the following figure.

Figure 7–14 Creating a Union for Structural Links



The following structural links were joined when we created the link from **pODate1/billing-date** to **dates/value**.

- **pODate1/billing-date** to **dates/date**
- **pODate1/delivery-date** to **dates/date**

A second data link between the **pODate1/delivery-date** element and **dates/value** element was created automatically.

12. Create a link between the **pODate1/billing-date** source element and the **dates/type** target element.

Two data links are created.

Keep the **pODate1/billing-date** to **dates/type** link selected for the next step.

13. Select the **Target Expression** view.
14. Select the **If Then Else** option.

The following XQuery if-then-else expression is added to the link:

```
if (fn:boolean("true")) then
  data($PODate)
else
  ()
```

15. In this step, we add a condition to the **if** section of the if-then-else expression.

- a. Select **If Condition** in the **Expression Structure** area.

The Edit If Condition pane is displayed.

- b. In the **Expression Functions** view, expand **Node Functions**.

- c. Drag the **local-name** function to the **Left Hand Expression** area of the **Edit If Condition** pane. Leave the **\$node-var** argument in the function selected.

- d. Select the **Expression Variables** view.

- e. Drag the **PODate** structural link variable to the **\$node-var** argument of the **local-name** function in the **Left Hand Expression** area of the **Edit If Condition** pane.

- f. Select the = operator.

- g. In the **Right Hand Expression** area of the **Edit If Condition** pane, enter **"billing-date"** (including quotation marks), and then click **Add**.

The following condition is added to the **if** section of the if-then-else expression:

```
fn:local-name($PODate)="billing-date"
```

16. Select **Then Expression** in the **Expression Structure** area.

The Edit Then Condition pane is displayed.

17. Replace the existing text with **"BILLING"** (including quotation marks), and then click the **Apply** icon.

18. Select **Else Expression** in the **Expression Structure** area.

The **Edit Else Condition** pane is displayed.

19. Replace the existing text with **"DELIVERY"** (including quotation marks), and then click the **Apply** icon.

20. Select **If Then Else** in the **Expression Structure** area.

The following XQuery code is displayed in the **Expression Structure** pane.

```
if (fn:local-name($PODate) = "billing-date") then
  "BILLING"
else
  "DELIVERY"
```

21. Save the changes.

For information about testing XQuery files, see [Section 6.9, "Testing Data Transformations."](#)

7.6 Creating Nested If-Then-Else Expressions

In this example, we create an XQuery transformation that calculates price based on a widget ID and tax rate for a state. We create an if-then-else expression for the following logic:

- If widget ID is from 0 to 200, the price is \$10.00
- Else if the widget ID is from 201 to 400, price is \$20.00
- Else If the widget ID is from 401 to 600, price is \$30.00

This example includes the following steps:

- [Section 7.6.1, "Step 1. Create the XQuery Transformation"](#)
- [Section 7.6.2, "Step 2. Create the First "If" Condition"](#)
- [Section 7.6.3, "Step 3. Create the First Nested If-Then-Else Condition"](#)
- [Section 7.6.4, "Step 4. Create the Second Nested If-Then-Else Condition"](#)

7.6.1 Step 1. Create the XQuery Transformation

In this step, we create an XQuery transformation by using the **PurchaseAgree.xsd** and **Supplier.xsd** schemas.

1. Launch Eclipse and open the sample project.
 - For information about launching Eclipse, see [Section 6.1, "Launching XQuery Mapper."](#)
 - For information about opening the sample project, see [Section 6.2, "Importing the XQuery Mapper Sample Project."](#)
2. Right-click the **XQuery Transformations** folder.
3. Choose **New > XQuery Transformation**.
4. Verify the name of the parent folder. For this example, the parent folder is **/XQuery Transformation/XQueryTransformations**.
5. Enter **ifthenelse** as the file name and click **Next**.
6. Select **Supplier.xsd\Supplier** as the source type and click **Next**.
7. Select **PurchaseAgree.xsd\PurchaseOrder** as the target type and click **Finish**.

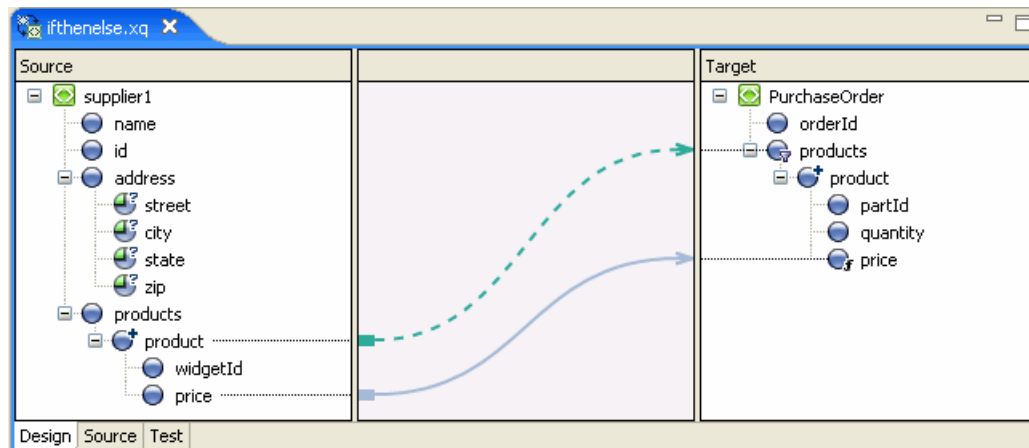
The **ifThenElse.xq** file is created.

8. Create links between the following source and target elements.

Source Element	Target Element
supplier1/products/product	PurchaseOrder/products/product
supplier1/products/product/price	PurchaseOrder/products/product/price

The links are displayed, as shown in the following figure.

Figure 7-15 XQuery Transformation for If-Then-Else Example



7.6.2 Step 2. Create the First "If" Condition

In this step, we create an **If** expression to specify that if the widget ID is from 0 to 200, then the price is \$10.00.

1. Select the link between the **supplier1/products/product/price** source element and the **PurchaseOrder/products/product/price** target element.
2. Select the **Target Expression** view.
3. Select the **If Then Else** expression type.
4. In the **Expression Structure** pane, select **If Condition**.
The **Edit If Condition** pane is displayed.
5. Drag the **supplier1\products\product\widgetID** element from the **Source** pane and drop it in the **Left Hand Expression** pane.
6. Select the **>=** operator.
7. In the **Right Hand Expression** pane, enter **"0"** (including quotation marks), and click **Add**.
8. Select the **<=** operator.
9. In the **Right Hand Expression** pane, enter **"200"**.
10. Click the arrow next to the **Update** button and select **Add**.
11. In the **Expression Structure** pane, select **Then Condition**.
The **Edit Then Condition** pane is displayed.
12. In the **Edit Then Expression** pane, delete the existing data and enter **"\$10.00"**.
13. Click the **Apply** icon.
14. In the **Expression Structure** pane, select **If Then Else**.

The if-then expression is displayed as shown in the following listing.

Example 7-3 If-Then Expression

```
if ((xs:string(data($product/ns0:widgetId)) >= "0"
    and xs:string(data($product/ns0:widgetId)) <= "200")) then
    "$10.00"
else
```

()

7.6.3 Step 3. Create the First Nested If-Then-Else Condition

In this step, we create an **If** expression to specify that if the widget ID is from 201 to 400, then the price is \$20.00. To accomplish this, we insert a nested if-then-else inside the **Else** expression we created in the previous step.

1. In the **Expression Structure** pane, right-click **Else Expression**, and select **Insert Nested If-Then-Else**.
2. In the nested **If-Then-Else** expression, select **If Condition**.
3. From the **Source** pane, drag the **widgetID** element and drop it in the **Left Hand Expression** pane.
4. Select the **>=** operator.
5. In the **Right Hand Expression** pane, enter **"201"** (including quotation marks), and click **Add**.
6. Select the **<=** operator.
7. In the **Right Hand Expression** pane, enter **"400"**.
8. Click the arrow next to the **Update** button and select **Add**.
9. In the **Expression Structure** pane, select **Then Expression**.
10. In the **Edit Then Expression** pane, enter **"\$20.00"**.
11. Click the **Apply** icon.
12. In the **Expression Structure** pane, select **If Then Else**.

The if-then-else expression appears as shown in the following listing.

Example 7-4 Nested If-Then-Else Expression

```
if ((xs:string(data($product/ns0:widgetId)) >= "0"
    and xs:string(data($product/ns0:widgetId)) <= "200")) then
    "$10.00"
else
    if ((xs:string(data($product/ns0:widgetId)) >= "201"
        and xs:string(data($product/ns0:widgetId)) <= "400")) then
        "$20.00"
    else
        ()
```

7.6.4 Step 4. Create the Second Nested If-Then-Else Condition

In this step, we create an **If** expression to specify that if the widget ID is from 401 to 600, then the price is \$30.00. To accomplish this, we insert a nested if-then-else expression within the **Else** expression that we created in the previous step.

1. In the **Expression Structure** pane, select the **Else** clause of the nested if-then-else expression created in [Section 7.6.3, "Step 3. Create the First Nested If-Then-Else Condition."](#)
2. Right-click and select **Insert Nested If-Then-Else**.
3. Select the **If** condition in the nested if-then-else expression that we just created.
4. From the **Source** pane, drag **widgetID** and drop it in the **Left Hand Expression** pane.

5. Select the `>=` operator.
6. In the **Right Hand Expression** pane, enter **"401"**, and then click **Add**.
7. Select the `<=` operator.
8. In the **Right Hand Expression** pane, enter **"600"**, and then click **Add**.
9. In the **Expression Structure** pane, select **Then Expression**.
10. In the **Edit Then Expression** pane, enter **"\$30.00"**, and click the **Apply** icon.
11. In the **Expression Structure** pane, select **If Then Else**.

The nested if-then-else expression is as shown in the following listing.

Example 7-5 Nested If-Then-Else Expression

```
if ((xs:string(data($product/ns0:widgetId)) >= "0"
    and xs:string(data($product/ns0:widgetId)) <= "200")) then
    "$10.00"
else
    if ((xs:string(data($product/ns0:widgetId)) >= "201"
        and xs:string(data($product/ns0:widgetId)) <= "400")) then
        "$20.00"
    else
        if ((xs:string(data($product/ns0:widgetId)) >= "401"
            and xs:string(data($product/ns0:widgetId)) <= "600")) then
            "$30.00"
        else
            ()
```

For information about testing XQuery files, see [Section 6.9, "Testing Data Transformations."](#)

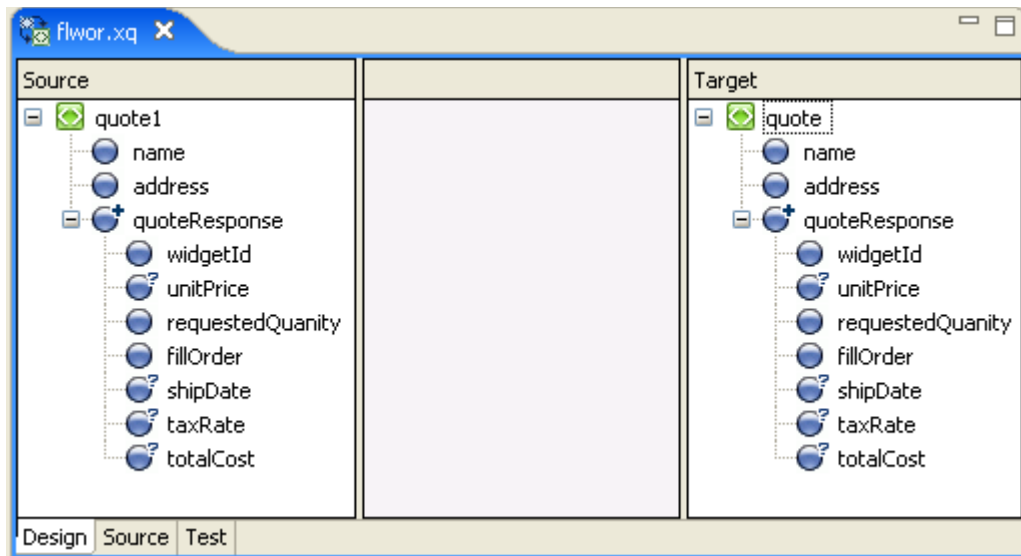
7.7 Creating FLWOR Expressions

In this example, we use a For-Let-Where-Order By-Return expression to extract widget IDs from a quotation, for items with a total value more than 2000.

1. Create the XQuery file.
 - a. Launch Eclipse and open the sample project.
 - For information about launching Eclipse, see [Section 6.1, "Launching XQuery Mapper."](#)
 - For information about opening the sample project, see [Section 6.2, "Importing the XQuery Mapper Sample Project."](#)
 - b. Right-click the **XQuery Transformations** folder.
 - c. Choose **New > XQuery Transformation**.
 - d. Verify the name of the parent folder.
 - e. Enter **flwor** as the file name and click **Next**.
 - f. Select **Quote.xsd\quote** as the source type and click **Next**.
 - g. Select **Quote.xsd\quote** as the target type and click **Finish**.

The **flwor.xq** file is created, as shown in the following figure.

Figure 7-16 XQuery Example - FLWOR Expression



2. Design the **Let** clause.

Note: For this example, the let clause is not essential. It is used here merely to illustrate how to design it in XQuery Mapper.

- a. In the **Expression Structure** pane, select **Let Clause**.
The **Edit Let Clause** pane is displayed.
 - b. In the **Variable** field, replace the existing value with **widget**.
 - c. Select the **Expression Variables** view.
 - d. Expand the **quote** node within the **Structural Link** folder.
 - e. Drag **quote/widgetID** from the **Expression Variables** view and drop it in the **Single Expression** field.
 - f. Click **Update**.
3. Design the **Where** clause.
- a. In the **Expression Structure** pane, right-click **For...Return** and select **Insert Where Clause**.
 - b. Select **Where Clause**.
The **Edit Where Condition** pane is displayed.
 - c. Drag **quote/totalCost** from the **Structural Link** folder of the **Expression Variables** view, and drop it in the **Left Hand Expression** field.
 - d. Select the **>** operator.
 - e. Enter **2000** in the **Right Hand Expression** area.
 - f. Click **Add**.
4. Design the **Order By** clause.
- a. In the **Expression Structure** pane, right-click **For...Return**, and select **Insert Order By Clause**.

- b. **Select Order By Clause.**
The Edit Order By Clause pane is displayed.
 - c. In the **Sort Order** field, select **ascending**.
 - d. In the **Single Expression** field, enter **\$widget**, which is the name of the variable that is declared in the **let** clause.
 - e. Click **Update**.
5. Design the Return expression.
 - a. In the **Expression Structure** pane, select **Return Expression**.
 - b. In the **Expression Variables** view, expand the **Structural Link** folder.
 - c. Drag **quote** from the **Expression Variables** view, and drop it in the **Single Expression** field.
 - d. Click the **Apply** icon.
 6. Save the changes.

You can view the source code of the FLWOR expression by selecting **For...Return** in the **Expression Structure** pane. The code is as shown in the following listing.

Example 7-6 Code for FLWOR Expression

```
for $quote in ($quote1/quoteResponse)
let $widget := ($quote/widgetId)
where $quote/totalCost > 2000
order by $widget ascending
return
  $quote
```

For information about testing XQuery transformations, see [Section 6.9, "Testing Data Transformations."](#)

7.8 Using Recursive Schemas

In this example, we create a data transformation with schemas that have recursive elements.

An element in a schema is considered recursive when it contains a child element of the same type as the parent, as shown in the example in [Example 7-7](#). In this example, the **product** element is a recursive element because it is of type **productType**, and **productType** contains a **child-product** element which is also of type **productType** (**productType** refers to itself).

Example 7-7 Example of Recursive Schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.acme.org/Product"
  xmlns="http://www.acme.org/Product" elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:complexType name="productType">
    <xs:sequence>
      <xs:element name="part-description" minOccurs="0"
        maxOccurs="unbounded" type="xs:string" />
      <xs:element name="child-product" minOccurs="0"
        maxOccurs="unbounded" type="producttype" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

```

    </xs:sequence>
  </xs:complexType>
  <xs:element name="product" type="productType">
  </xs:element>
</xs:schema>

```

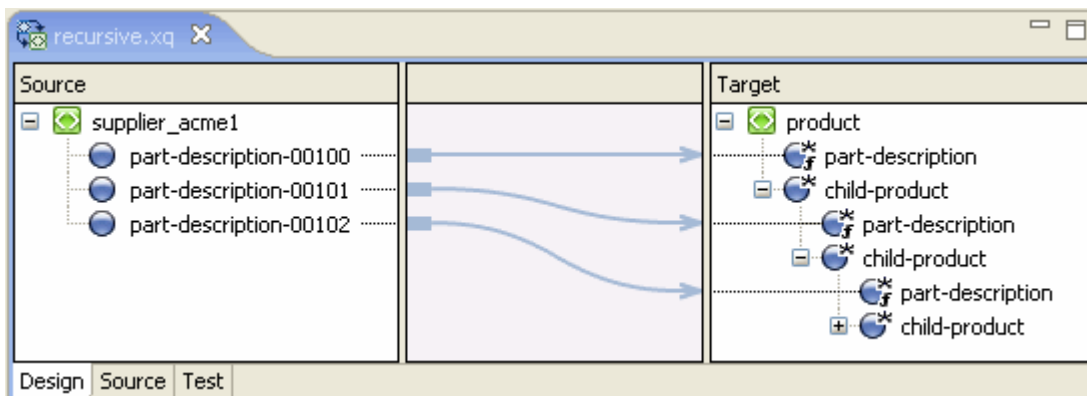
Perform the following steps to create a transformation with recursive schemas:

1. Launch Eclipse and open the sample project.
 - For information about launching Eclipse, see [Section 6.1, "Launching XQuery Mapper."](#)
 - For information about opening the sample project, see [Section 6.2, "Importing the XQuery Mapper Sample Project."](#)
2. Right-click the **XQuery Transformations** folder.
3. Choose **New > XQuery Transformation**.
4. Verify the name of the parent folder. For this example, the parent folder is **/XQuery Transformation/XQueryTransformations**.
5. Enter **recursive** as the file name and click **Next**.
6. Select **SupplierAcme.xsd\supplier_acme** as the source schema and click **Next**.
7. Select **Product.xsd\product** as the target schema and click **Finish**.
The **recursive.xq** file is created.
8. Create links between the following source and target elements:

Source Element	Target Element
supplier_acme1\part-description-00100	product\part-description
supplier_acme1\part-description-00101	product\child-product\part-description
supplier_acme1\part-description-00101	product\child-product\child-product\part-description

The following figure shows the links from the source elements to the recursive **child-product** target elements.

Figure 7–17 Mapping Recursive Elements



9. Save the changes.

For information about testing XQuery files, see [Section 6.9, "Testing Data Transformations."](#)

7.9 Grouping Data by Key Fields

You can use the **Group by Key Fields** feature to group data based on one or more key values.

Note: The Group-By feature is not supported graphically in XQuery Mapper and there is no representation of the XQuery in the Design view. You must write the Group-By expression in the Source view.

The following listing shows the XML document that we use as input in this example.

Example 7-8 Example Input XML Document

```
<input-warehouse-inventory xmlns="http://www.creditpo.org/repkeyin">
<input-line-item>
  <input-warehouse-id>Warehouse1</input-warehouse-id>
  <input-location-desc>Location1</input-location-desc>
  <input-part-no>1</input-part-no>
  <input-quantity>10</input-quantity>
</input-line-item>
<input-line-item>
  <input-warehouse-id>Warehouse2</input-warehouse-id>
  <input-location-desc>Location2</input-location-desc>
  <input-part-no>2</input-part-no>
  <input-quantity>20</input-quantity>
</input-line-item>
<input-line-item>
  <input-warehouse-id>Warehouse1</input-warehouse-id>
  <input-location-desc>Location1</input-location-desc>
  <input-part-no>3</input-part-no>
  <input-quantity>30</input-quantity>
</input-line-item>
</input-warehouse-inventory>
```

In this example, we use the **input-warehouse-id** and **input-location-desc** elements as the key fields to group data in the output document:

The first and third instances of the **input-line-item** repeating element contain the same values for the **input-warehouse-id** and **input-location-desc** elements: **Warehouse1** and **Location1** respectively.

The goal of this example is to write an XQuery that groups the first and third instances of the line items, by using the **Warehouse1** and **Location1** keys in the output document, as shown in [Example 7-9](#).

Example 7-9 Example Output XML Document

```
<ns0:output-inventory xmlns:ns0="http://www.creditpo.org/repkeyout";>
  <ns0:output-warehouse-inventory>
    <ns0:output-warehouse-id>Warehouse1</ns0:output-warehouse-id>
    <ns0:output-location-desc>Location1</ns0:output-location-desc>
    <ns0:output-line-item>
      <ns0:output-part-no>1</ns0:output-part-no>
      <ns0:output-quantity>10</ns0:output-quantity>
    </ns0:output-line-item>
```

```

    <ns0:output-line-item>
      <ns0:output-part-no>3</ns0:output-part-no>
      <ns0:output-quantity>30</ns0:output-quantity>
    </ns0:output-line-item>
  </ns0:output-warehouse-inventory>
</ns0:output-warehouse-inventory>
<ns0:output-warehouse-inventory>
  <ns0:output-warehouse-id>Warehouse2</ns0:output-warehouse-id>
  <ns0:output-location-desc>Location2</ns0:output-location-desc>
  <ns0:output-line-item>
    <ns0:output-part-no>2</ns0:output-part-no>
    <ns0:output-quantity>20</ns0:output-quantity>
  </ns0:output-line-item>
</ns0:output-warehouse-inventory>
</ns0:output-inventory>

```

Perform the following steps to create a Group-By expression:

1. Launch Eclipse and open the sample project.
 - For information about launching Eclipse, see [Section 6.1, "Launching XQuery Mapper."](#)
 - For information about opening the sample project, see [Section 6.2, "Importing the XQuery Mapper Sample Project."](#)
 2. Right-click the **XQuery Transformations** folder.
 3. Choose **New > XQuery Transformation**.
 4. Verify the name of the parent folder. For this example, the parent folder is **/XQuery Transformation/XQueryTransformations**.
 5. Enter **groupby** as the file name and click **Next**.
 6. Select **regroupKeyFldIn.xsd\input-warehouse-inventory** as the source schema, and click **Next**.
 7. Select **regroupKeyFldOut.xsd\output-inventory** as the target schema, and click **Finish**.
- The **groupby.xq** file is created.
8. Select the **Source** view.
 9. Replace the existing code with the code in the following listing.

Example 7–10 XQuery Code for Group-By Expression

```

declare namespace ns0 = "http://www.creditpo.org/repkeyin";
declare namespace ns1 = "http://www.creditpo.org/repkeyout";
declare function Regrouping($input-warehouse-inventory as
  element(ns0:input-warehouse-inventory))
  as element(ns1:output-inventory) {
  <ns1:output-inventory>
    {
      for $input-line-item in $input-warehouse-inventory/ns0:input-line-item
      group $input-line-item as $group by
        $input-line-item/ns0:input-warehouse-id as $key0,
        $input-line-item/ns0:input-location-desc as $key1
      return
        <ns1:output-warehouse-inventory>
          <ns1:output-warehouse-id>{ data($key0) }</ns1:output-warehouse-id>
          <ns1:output-location-desc>{ data($key1) }</ns1:output-location-desc>
          {

```



```

        for $group0 in $group return
        <ns1:output-line-item>
        <ns1:output-part-no>{xs:byte(data($group0/ns0:input-part-no))}
        </ns1:output-part-no>
        <ns1:output-quantity>{xs:byte
        (data($group0/ns0:input-quantity)) }
        </ns1:output-quantity>
        </ns1:output-line-item>
    }
    </ns1:output-warehouse-inventory>
}
</ns1:output-inventory>
};
declare variable $input-warehouse-inventory as
element(ns0:input-warehouse-inventory) external;
Regrouping($input-warehouse-inventory)

```

10. Save the changes.

The changes are not visible in the Design view.

11. With the **groupby.xq** file open in the **Source** view, select the **Test** view.

12. In the **Source Data** pane, click the **Import** icon.

13. Import the **Regrouping.xml** file provided in the sample project (from the **XML Transformation/XML/** folder).

14. In the **Result Data** pane, select the **Test XQuery** icon.

The result of the XQuery is displayed, as shown in [Example 7-9](#).

Upgrading XQuery Code

This section describes the procedure for upgrading inline XQuery code and stand alone XQuery files from XQuery 2002 to 2004.

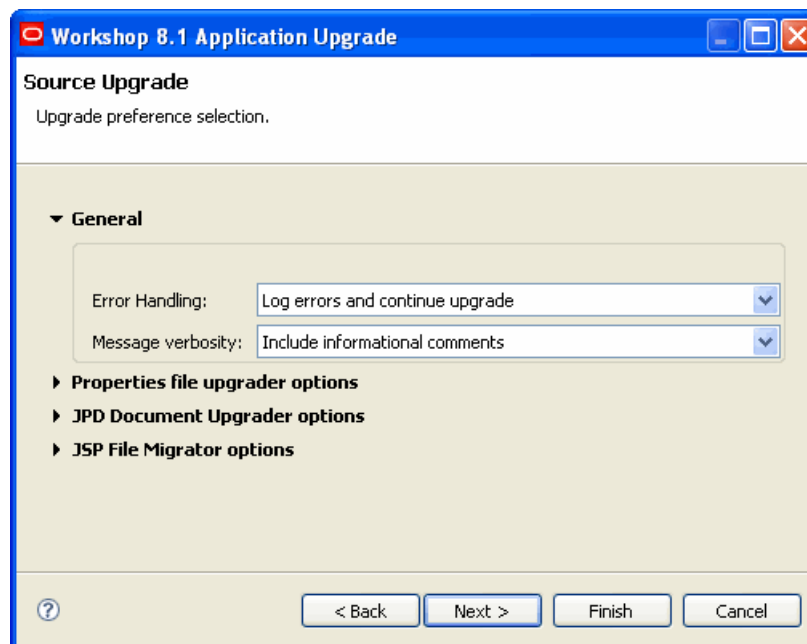
This section contains the following topics:

- [Section 8.1, "Upgrading Inline XQuery Code"](#)
- [Section 8.2, "Upgrading XQuery Files"](#)

8.1 Upgrading Inline XQuery Code

When you upgrade a WLI 8.x project to 10g Release 3 (10.1.3), by default, inline XQuery 2002 code (embedded in JPD files) is not converted to XQuery 2004. You can convert the inline XQuery code to XQuery 2004, by performing the following steps in the **Source Upgrade** screen of the upgrade wizard.

Figure 8–1 Source Upgrade Screen of the Upgrade Wizard

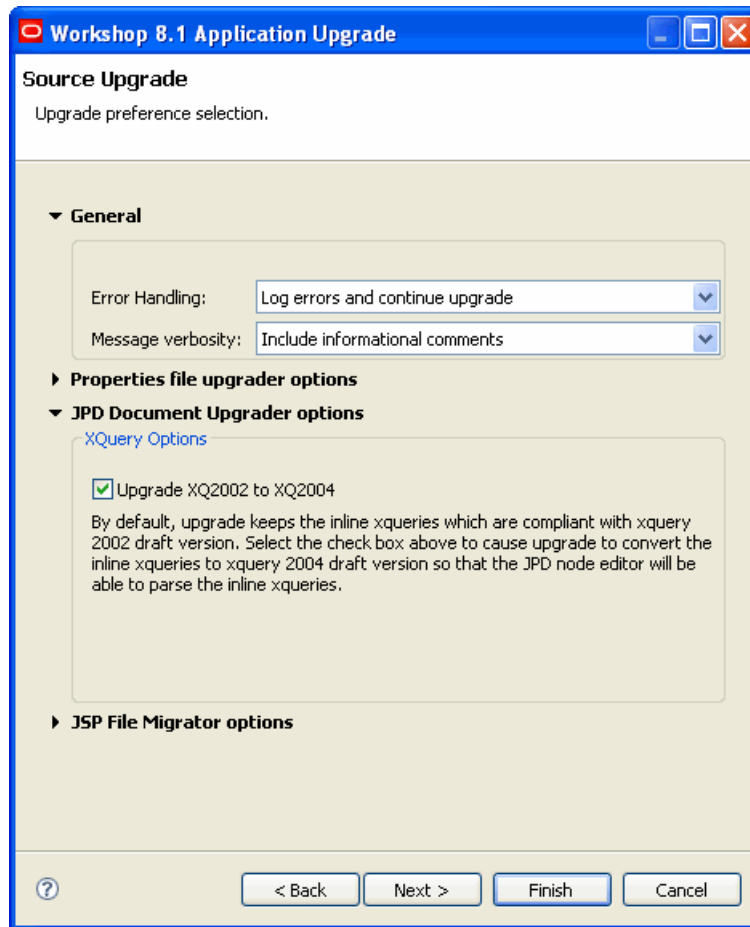


To Upgrade Inline XQuery Code:

1. Expand **JPD Document Upgrader options**.

2. Select the **Upgrade XQ2002 to XQ2004** check box, as shown in the following figure.

Figure 8–2 Upgrade XQ2002 to XQ2004



3. Continue with the application upgrade process.

When the upgrade process is completed, XQuery 2002-compliant code that is embedded in JPD files (**inline XQuery code**) is converted to XQuery 2004.

8.2 Upgrading XQuery Files

XQuery 2002-compliant **XQuery files** (unlike **inline XQuery code**) are not converted automatically to XQuery 2004 during the application upgrade process.

You can upgrade XQuery 2002-compliant XQuery files to XQuery 2004 by right-clicking the XQuery file in the **Navigator** view in Eclipse and choosing **Upgrade to XQuery 2004**. However, the following XQuery constructs and functions are not converted to XQuery 2004:

- **Constructs for which the syntax has changed in XQuery 2004:**

Table 8–1 Constructs For Which the Syntax Has Changed in XQuery 2004

XQuery 2002	XQuery 2004	Note
precedes follows	<< >>	In some cases, merely changing the syntax might not give correct results.
add-timezone-to-dateTime remove-timezone-from-dateTime	fn:adjust-dateTime-to-timezone fn:adjust-dateTime-to-timezone	The syntax and parameter type have changed. In the XQuery 2004 syntax, the operation is addition regardless of the sign of the parameter. In XQuery 2004, the operation depends on the sign of the parameter.
document()	fn:doc()	The parameter type and return type might be different.
foreach	for \$i in...	There is no direct replacement for foreach . The construct must be rewritten by using a for loop.
some \$x in (1, 2, 3), xs:integer \$y in (2, 3, 4) every \$x in (1, 2, 3), xs:integer \$y in (2, 3, 4)	some \$x in (1,2,3), \$y in (2,3,4) every \$x in (1,2,3), \$y in (2,3,4)	Existence of the optional type declaration makes it difficult to reliably convert this construct.

- Functions for which the semantics have changed in XQuery 2004:

Table 8–2 Functions For Which Parameter Types and Order Have Changed in XQuery 2004

XQuery 2002	XQuery 2004
xf:resolve-uri(anyURI \$base, anyURI \$relative)	fn:resolve-uri(\$relative as xs:string?, \$base as xs:string)
xf:distinct-values(item* \$srcval) => item*	fn:distinct-values (\$arg as xdt:anyAtomicType*) as xdt:anyAtomicType*

- Constructs for which the casting rule has changed: **eq**, **ne**, **lt**, **gt**, **le**, and **ge**.
- Functions for which the return type has changed in XQuery 2004
 - Most time/date-related functions and operators
 - Queries containing **orderby**
 - **generic-divide**
 - **div**
 - **expanded-QName**
 - **namespaces**
 - **QName**
 - **node-name**
 - **namespace-uri**
 - **subsequence**
 - **integer-divide**

- **round**
- **name**
- Function not supported in XQuery 2004: **julianDay-from-date**

During the upgrade process, XQuery 2002 constructs and functions that are not supported by XQuery 2004 are flagged with comment entries indicated by "**: Warning :**" in the code. In addition, a warning message is displayed during the upgrade process, informing you that unsupported constructs and functions were encountered.

After the upgrade process, you can identify the constructs and functions that are not upgraded to XQuery 2004 by looking for the "**: Warning :**" comments. You must then manually change the unsupported constructs and functions to make them comply with the XQuery 2004 standard.

Part III

Format Builder

This part contains the Format Builder IDE help, which includes the following chapters:

- [Chapter 9, "Introduction"](#)
- [Chapter 10, "Format Builder Main Window"](#)
- [Chapter 11, "Message Format Detail Window"](#)
- [Chapter 12, "Field Detail Window"](#)
- [Chapter 13, "Group Detail Window"](#)
- [Chapter 14, "Reference Detail Window"](#)
- [Chapter 15, "Comment Detail Window"](#)
- [Chapter 16, "Format Builder Options"](#)
- [Chapter 17, "Importing Metadata"](#)
- [Chapter 18, "Format Tester"](#)
- [Chapter 19, "Format Builder Menus"](#)
- [Chapter 20, "How Do I?"](#)
- [Chapter 21, "Using the Palette"](#)
- [Chapter 22, "Format Builder Supported Data Types"](#)

Introduction

The Format Builder tool assists you in creating descriptions of non-XML data records. Format Builder allows you to describe the layout and hierarchy of the non-XML data so that it can be transformed to or from XML. With Format Builder, you can describe sequences of bytes as fields. Each field description includes the type of data (floating point, string, etc.), the size of the data, and the name of the field. Format Builder allows you to further define groupings of fields (Groups), repetition of fields and groups, and aggregation.

The descriptions you create in Format Builder are saved in an XML grammar called Message Format Language (MFL). MFL documents are used at run-time to transform an instance of a non-XML data record to an instance of an XML document (or vice-versa).

9.1 Overview

This help system describes how to use Format Builder to define schemas for non-XML documents. These schemas can be used by Xquery Mapper tools to automatically transform data between XML and non-XML formats.

The following topics are included:

- [Chapter 10, "Format Builder Main Window"](#) describes how to navigate the main window of Format Builder.
- [Chapter 11, "Message Format Detail Window"](#) describes the fields on the Message Format detail window.
- [Chapter 13, "Group Detail Window"](#) describes the fields on the Group Details detail window.
- [Chapter 15, "Comment Detail Window"](#) describes the fields on the Field Details detail window.
- [Chapter 14, "Reference Detail Window"](#) describes the fields on the Reference Details detail window.
- [Chapter 15, "Comment Detail Window"](#) describes the fields on the Comment Details detail window.
- [Chapter 16, "Format Builder Options"](#) describes the fields on the Format Builder Options detail window.
- [Chapter 18, "Format Tester"](#) describes how to use Format Tester.
- [Chapter 17, "Importing Metadata"](#) describes how to use Format Builder to import a COBOL Copybook, or C Structure to generate XML files.

- [Chapter 19, "Format Builder Menus"](#) describes the commands you can execute from the Format Builder menus.
- [Chapter 21, "Using the Palette"](#) describes the Format Builder palette feature and how to use it.
- [Chapter 20, "How Do I?"](#) gives you step-by-step instructions for performing the basic tasks in Format Builder.
- [Chapter 22, "Format Builder Supported Data Types"](#) Provides a list of supported MFL and COBOL Copybook Importer data types.

Format Builder Main Window

The main window of the Format Builder is split into two panes. The left pane shows the structural information for the data format. The right pane shows the detail for the item selected in the left pane.

You can navigate in and execute commands from the main window by using one of the following methods:

- [Section 10.1, "Using the Menu Bar"](#)
- [Section 10.2, "Using the Toolbar"](#)
- [Section 10.3, "Using the Tree Pane"](#)
- [Section 10.4, "Using the Shortcut Menus"](#)

10.1 Using the Menu Bar

The Menu bar displays the menu headings. The menus that are available depend on what is selected in the left pane. You can open menus from the menu bar or by holding down the ALT key while pressing the underscored letter in the menu heading. For example, pressing ALT + F opens the File Menu.

For a complete description of the menu commands, see [Chapter 19, "Format Builder Menus"](#)

10.2 Using the Toolbar

The Format Builder toolbar provides buttons that access some of the frequently used commands in the menus. If a command is unavailable, its button appears "grayed-out."

The toolbar buttons provided with Format Builder are described below:

Table 10–1 List of Toolbar Buttons

Toolbar Button Name	Description
New	Creates a new Message Format
Open/Retrieve	Opens an existing Message Format.
Save/Store	Saves the current Message Format
Cut	Removes the item currently selected in the left-hand pane, and its child objects, from the tree. Note: This action is not available if the Message Format (root) item is selected.

Table 10–1 (Cont.) List of Toolbar Buttons

Toolbar Button Name	Description
Copy	Makes a copy of the item currently selected in the left-hand pane for insertion elsewhere in the tree. Note: This action is not available if the Message Format (root) item is selected.
Paste as Sibling	Inserts the cut or copied item as a sibling object of the selected item.
Paste as Reference	Inserts a reference to the cut or copied item as a sibling object of the selected item.
Undo	Reverses the previous action. The tool tip changes to indicate the action that can be undone. For example, changing the name of a field to Field1 and clicking Apply causes the tool tip to read "Undo Apply Field Field1". Note: Format Builder supports multi-level undoing and redoing.
Redo	Reverses the effects of an Undo command. The tool tip changes to indicate the action that can be redone. For example, changing the name of a field to Field1 and then undoing that action causes the tool tip to read "Redo Apply Field Field1". Note: Format Builder support multi-level undoing and redoing.
Insert Field	Inserts a field as a sibling of the item selected in the tree pane.
Insert Group	Inserts a group as a sibling of the item selected in the tree pane.
Insert Comment	Inserts a comment as a sibling of the item selected in the tree pane.
Move Up	Moves the selected item up one position under its parent.
Move Down	Moves the selected item down one position under its parent.
Promote item	Promotes the selected item to the next highest level in the tree. For example, Field1 is the child object of Group1. Selecting Field1 and clicking the Promote tool makes it a sibling of Group1.
Demote item	Demotes the selected item to the next lower level in the tree. For example, Group1 is the sibling of Field1. Field1 immediately follows Group1 in the tree. Selecting Field1 and clicking the Demote tool makes it a child of Group1.
Expand All	Expands all items in the tree pane to show child items.
Collapse All	Collapses the tree pane to show first level items only.
Format Tester	Opens the Chapter 18, "Format Tester" window.

10.3 Using the Tree Pane

The Tree Pane represents hierarchical/structural information about the format of the non-XML data in a tree. The root node of the tree will correspond to the MFL document being created or edited. The root node is referred to as the Message node. Child nodes are labeled with group or field names. Fields are represented by leaf nodes in the tree. Groups contain fields or other groups and are represented by non-leaf nodes in the tree.

The icon for each node encapsulates information about the node. The icon indicates whether the node represents a message, a group, a field, a comment, or a reference. The icon also indicates whether a group or field is repeating, whether a group is a Group Choice, and whether a group or field is optional or mandatory. You also have the ability to add, delete, move, copy, or rename nodes in the tree. This is done

through the menus or the toolbar (see [Section 10.1, "Using the Menu Bar,"](#) [Section 10.2, "Using the Toolbar,"](#) and [Section 10.4, "Using the Shortcut Menus"](#)).

The icons that appear in the Tree Pane are described in the following table.

Table 10–2 Tree Icon Descriptions

Tree Icon Name	Description
Message Format	The top level element.
Group	Collections of fields, comments, and other groups or references that are related in some way (for example, the fields <code>PAYDATE</code> , <code>HOURS</code> , and <code>RATE</code> could be part of the <code>PAYINFO</code> group). Defines the formatting for all items contained in the group.
Optional Group	A group that may or may not be included in the message format.
Repeating Group	A group that has one or more occurrences.
Optional Repeating Group	A group that may or may not be included, but if included, occurs more than once.
Group Reference	Indicates that another instance of the group exists in the data. Reference groups have the same format as the original group, but you can change the optional setting and the occurrence setting for the reference group.
Group Choice	Indicates that only one of the items in the group will be included in the message format.
Field	Sequence of bytes that have some meaning to an application. (For example, the field <code>EMPNAME</code> contains an employee name.) Defines the formatting for the field.
Optional Field	A field that may or may not be included in the message format.
Repeating Field	A field that has one or more occurrences.
Field Reference	Indicates that another instance of the field exists in the data. Reference fields have the same format as the original field, but you can change the optional setting and the occurrence setting for the reference field.
Optional Repeating Field	A field that may or may not be included, but, if included, occurs more than once in the message format.
Comment	Contains notes about the message format or the data transformed by the message format.
Collapse	A minus sign next to an object indicates that it can be collapsed.
Expand	A plus sign next to an object indicates that it can be expanded to show more objects.

10.4 Using the Shortcut Menus

Instead of using the standard menus to find the command you need, use the right mouse button to click an item in the left pane. The menu that appears shows the most frequently used commands for that item.

For a complete description of the shortcut menu commands, see [Section 19.7, "Shortcut Menu."](#)

10.5 Using Drag and Drop

You can use the drag and drop feature of Format Builder to copy and/or move the items in the tree view.

To use drag and drop to move an item:

1. Select the item you want to move.
2. Press and hold the left mouse button while you drag the item to the desired location.
3. When the item is in the desired location, release the left mouse button. The item is moved to the new location.

To use drag and drop to copy an item:

1. Select the item you want to copy.
2. Press and hold the CTRL key.
3. Keeping the CTRL key depressed, press and hold the left mouse button while you drag the item to the desired location.
4. When the item is in the desired location, release the left mouse button and the CTRL key. A copy of the item is placed at the new location.

10.6 Valid Names

Message formats, fields, and groups are identified by a name. This name is used as the XML tag when non-XML data is transformed to XML. Therefore the name must conform to the XML rules for a name.

The format guidelines for a name are as follows:

- Must start with a letter or underscore.
- Can contain letters, digits, colon, the period character, the hyphen character, or the underscore character.

The following are valid name examples:

```
MyField  
MyField1  
MyField_again  
MyField-again
```

The following are invalid name examples:

```
1MyField - may not start with a digit  
My>Field - the greater-than sign (>) is an illegal character  
My Field - a space is not permitted  
My/Field - the back slash (/), which is an illegal character  
My\Field - the forward slash (\), which is an illegal character  
My:Field - a semi-colon (;), which is an illegal character
```

Message Format Detail Window

Defines the data for which you are creating a message format (the root node of a message format file).

Table 11-1 *Message Format*

Field	Description
Name/XML Root	The name of the message format. This value will be used as the root element in the transformed XML document. This name must comply with XML element naming conventions.
Apply Button	Saves your changes to the message format document.
Reset Button	Discards your changes to the detail window and resets all fields to the last saved values.
Help Button	Displays online help information for this detail window.

Note: The Apply and Reset buttons are only enabled once changes are made to the detail panel's components.

Field Detail Window

Defines the fields contained in the message format. Fields are a sequence of bytes that have some meaning to an application. (For example, the field `EMPNAME` contains an employee name.) You can create a field as a child of the message format item, as a child of a group, or as a sibling of a group or another field.

Table 12–1 *Fields in Message Format – Field Description*

Field	Description
Name	The name of the field. This name must comply with XML element naming conventions (see Section 10.6, "Valid Names" for more information).
Optional	Select this option if this is an optional field. Optional means that the data for the field might be present in the input. If the Optional option is selected for a file, then you can set the Field is Tagged option from the Field Attributes pane. In addition, in the Field Is Tagged text box, enter a unique value for each optional field in a group. Multiple groups can use the same tag value, but the tag value for each optional field in a group must be unique.
Type	Select the data type of the field from the list. The default is String. Note: The Field Type you select dictates the Field Data Options that appear on the dialog. Refer to Chapter 22, "Format Builder Supported Data Types" for a list of data types supported by Format Builder.

Table 12–2 *Fields in Message Format – Field Occurrence*

Field	Description
Occurrence	Choose one of the following to indicate how often this field appears in the message format: <ul style="list-style-type: none"> ■ Once - Indicates that the field appears only once. ■ Repeat Delimiter - Indicates that the field repeats until the specified delimiter is encountered. ■ Repeat Field - Indicates that the value of the repeat field at run time is the number of times the field repeats. ■ Repeat Number - Indicates that the field repeats the specified number of times. ■ Unlimited - Indicates that the field repeats an unlimited number of times. <p>Note: Unless a field is defined as optional, the field occurs at least once.</p>

The fields in the following sections of the detail window depend on the Field Type selected.

Table 12–3 Fields in Message Format – Field Data Options

Field	Description
Data Base Type	If the field is a date or time field, the base type indicates what type of characters (ASCII, EBCDIC, or Numeric) make up the data.
Year Cutoff	If the field is a date field that has a 2-digit year, the year cutoff allows the 2-digit year to be converted to a 4-digit year. If the 2-digit year is greater than or equal to the year cutoff value, a '19' prefix will be added to the year value. Otherwise a '20' prefix will be used.
Code Page	The character encoding of the String field data.
Value	The value that appears in a literal field.

Table 12–4 Fields in Message Format – Field Attributes

Field	Description
Field is Tagged	Select this option if this is a tagged field. Being tagged means that a literal precedes the data, indicating that the data is present. For example: SUP:ACME INC, SUP: is a tag. ACME INC is the field data. If you have selected the Field is Tagged option, enter the tag in the text box to the right of the checkbox.
Field Default Value	Select this option if the field has a default value. Then, enter the default value in the text box to the right of the checkbox.

Table 12–5 Fields in Message Format – Termination

Field	Description
Length	Variable-sized data types can be assigned a fixed length, eliminating the need to use a delimiter to specify the termination point of the field. <ul style="list-style-type: none">▪ Length - Enter the number of bytes in the length field if the length field is a variable length.▪ Trim Leading/Trailing - Removes the specified data from the leading or trailing edge of the data.▪ String Length in Characters - By default, the string length is in bytes. Select this check box if string is multi-byte encoded to calculate the length in number of characters instead of bytes.▪ Pad - If the data is shorter than the specified length, enter the necessary value to the data until it is of correct length. Select the Trailing option to append padding at the end of a field. Select the Leading option to append padding at the beginning of a field.▪ Truncate - Removes a specified number of characters from a field. Select the Truncate First option to remove the specified number of characters from the beginning of the field. Select the Truncate After option to remove the specified number of characters from the end of the field. <p>If you select both truncation options, the Truncate First option is implemented initially, and the Truncate After option is invoked on the remaining characters.</p>

Table 12–5 (Cont.) Fields in Message Format – Termination

Field	Description
Imbedded Length	<p>Variable-sized data types can have their termination point specified by an imbedded length. An imbedded length precedes the data field and indicates how many bytes the data contains.</p> <ul style="list-style-type: none">▪ Description - Select the Type from the drop down list. Then, depending on the Type selected, choose Length and enter the number of bytes, or choose Delimiter and enter the delimiter character.▪ Tag/Length Order - Specifies the order of tag and length fields when both are present. Default is tag before length.▪ Trim Leading/Trailing - Removes the specified data from the leading or trailing edge of the data.▪ Truncate - Removes a specified number of characters from a field. For more information on truncation, see the Length field.
Delimiter	<p>Variable-sized data types can have their termination point specified by a delimiter. A delimiter is a character that marks the end of the field. The field data continues until the delimiter character is encountered.</p> <ul style="list-style-type: none">▪ Value - Enter the delimiter that marks the end of the field data.▪ Trim Leading/Trailing - Removes the specified data from the leading or trailing edge of the data.▪ Truncate - Removes a specified number of characters from a field. For more information on truncation, see the Length field.
Delimiter Field	<p>Variable-sized data types can have their termination point specified by a field that contains a delimiter character. A delimiter is a character that marks the end of the field. The field data continues until the field containing the delimiter character is encountered.</p> <ul style="list-style-type: none">▪ Field - Select the field that contains the delimiter character.▪ Default - Enter the delimiter character. You must supply a default value. The default is used when the delimiter field is not present.▪ Trim Leading/Trailing - Removes the specified data from the leading or trailing edge of the data.▪ Truncate - Removes a specified number of characters from a field. For more information on truncation, see the Length field. <p>For more information on delimiters, see Section 20.12, "Character Delimiters."</p>
Decimal Position	<p>Specifies the number of digits (0-16) to the left of the decimal point.</p>

Table 12–6 Fields in Message Format – Literal

Field	Description
Value	<p>Specify the literal value. Literal value can be defined as a single value or it can be defined a list of values separated by the literal separator. When the Value is a list of values, the data for the literal field in the binary data will be one of value in the list.</p>

Table 12–6 (Cont.) Fields in Message Format – Literal

Field	Description
Literal Separator	<p>Supports enumeration of literal values. For literal type Field in MFL definition, a literal separator can be specified when multiple choices of value is needed for the Field.</p> <p>For example, segment terminators that are supported by both EDIFACT and X12 EDI standards are: \r\n, \r, \n, ', and ~. However, you can use Format Builder to support any other custom terminator. You can append the custom terminator to the existing list of literal values and use comma (,) as literal separator to separate multiple custom values.</p> <p>In the MFL file, you should see the following structure,</p> <pre><FieldFormat name='ISA_Terminator' type='Literal' value='\r\n,\r,\n,~, ' literalSeparator=',' /></pre>

Table 12–7 Fields in Message Format – Field Update Buttons

Field	Description
Apply	Saves your changes to the message format file.
Duplicate	<p>Makes a copy of the field currently displayed. The duplicate field contains the same values as the original field. The name of the duplicate field is the same as the original field name, with the word "New" inserted before the original name. For example, duplicating a field called "Field1" results in a field with the name "NewField1".</p> <p>When you duplicate an item with a numeric value in its name, the new item name contains the next sequential number. For example, duplicating "NewField1" results in a group named "NewField2".</p>
Reset	Discards your changes to the detail window and resets all fields to the last saved values.
Help	Displays online help information for this detail window.

Note: The Apply and Reset buttons are only enabled once changes are made to the detail panel's components.

Group Detail Window

Defines the groups contained in the message format. Groups are collections of fields, comments, and other groups or references that are related in some way (for example, the fields `PAYDATE`, `HOURS`, and `RATE` could be part of the `PAYINFO` group). You can create a group as a child of the message format item, as a child of another group, or as a sibling of a group or field.

Table 13–1 *Groups in Message Format – Group Description*

Field	Description
Name	The name of the group. This name must comply with XML element naming conventions (see Section 10.6, "Valid Names" for more information).
Optional	Choose Optional if this is an optional group.
Choice of Children	Choose Choice of Children if only one of the items in the group will be included in the message format.

Table 13–2 *Groups in Message Format – Group Occurrence*

Field	Description
Occurrence	<p>Choose one of the following to indicate how often this group appears in the message format:</p> <ul style="list-style-type: none"> ■ Once — Indicates the group appears only once. ■ Repeat Delimiter — Indicates the group will repeat until the specified delimiter is encountered. ■ Repeat Field — Indicates that the value of the repeat field at run time is the number of times the field will repeat. ■ Repeat Number — Indicates the group will repeat the specified number of times. ■ Unlimited — Indicates the group will repeat an unlimited number of times. <p>Note: Unless a group is defined as Optional, all groups occur at least once.</p>

Table 13–3 *Groups in Message Format – Group Attributes*

Field	Description
Group is Tagged	<p>Select this option if this is a tagged group. If tagged, a literal precedes the data, indicating that the data is present.</p> <p>If you selected the Group is Tagged option, enter the tag in the text box to the right of the checkbox.</p>

Table 13–4 Groups in Message Format – Group Delimiter

Field	Description
None	Select this option if the group has no delimiter.
Delimited	<p>Groups can have their termination point specified by a delimiter. A delimiter is a string of characters that marks the end of the group of fields. The group continues until the delimiter characters are encountered.</p> <p>Select this option if the end of the group is marked with a delimiter.</p> <p>Value — Enter the delimiter that marks the end of the group of fields.</p> <p>Note: Normally, groups are not delimited. They are usually parsed by content (the group ends when all child objects have been parsed).</p>
Delimiter Is Not Optional	Indicates whether the binary data contains the delimiter even if the group is not present.
Delimiter Field	<p>Groups can have their termination point specified by a field that contains a delimiter character string. A delimiter is a string of characters that mark the end of the group. The group continues until the delimiter character string contained in the specified field is encountered.</p> <ul style="list-style-type: none">▪ Field — Select the field that contains the delimiter character string. A list of valid fields will be presented in a list.▪ Default — Enter the default delimiter character that will be used if the above field is not present in the data. This value is required. <p>For more information on delimiters, see Section 20.12, "Character Delimiters."</p>
Delimiter is Shared	Indicates that the delimiter marks both the end of the group of data, and the end of the last field of the group. The delimiter is shared among the group, and the last field of the group, to delimit the end of the data.

Table 13–5 Groups in Message Format – Group Update Buttons

Field	Description
Apply	Saves your changes to the message format document.
Duplicate	<p>Makes a copy of the group currently displayed. The duplicate group contains the same values as the original group. The name of the duplicate group is the same as the original group name, with the word "New" inserted before the original name. For example, duplicating a group called "Group1" results in a group with the name "NewGroup1".</p> <p>When you duplicate an item with a numeric value in its name, the new item name contains the next sequential number. For example, duplicating "NewGroup1" results in a group named "NewGroup2".</p>
Reset	Discards your changes to the detail window and resets all fields to the last saved values.
Help	Displays online help information for this detail window.

Note: The Apply and Reset buttons are only enabled once changes are made to the detail panel's components.

Reference Detail Window

A reference is used to indicate that another instance of the field or group format exists in the data. Reference fields or groups have the same format as the original field or group, but you can change the optional setting and the occurrence setting for the reference field or group. For example, if you have a "bill to" address and a "ship to" address in your data, you only need to define the address format once. You can create the "bill to" address definition and create a reference for the "ship to" address.

Note: References are given the same name as the original item. For example, the "bill to" address definition and the "ship to" address definition would be named the same.

Table 14–1 Reference Detail Window – Reference Description

Field	Description
Name	Displays the name of the original field or group for which you created this reference. This value cannot be changed.
Optional	Select this option if the reference field or group is optional.

Table 14–2 Reference Detail Window – Field Occurrence

Field	Description
Occurrence	Choose one of the following to indicate how often this reference field or group appears in the message format: <ul style="list-style-type: none"> ■ Once — Indicates the reference appears only once. ■ Repeat Delimiter — Indicates the reference will repeat until the specified delimiter is encountered. For more information on delimiters, see Section 20.12, "Character Delimiters". ■ Repeat Field — Indicates that the value of the repeat field at run time is the number of times the field will repeat. ■ Repeat Number — Indicates the reference will repeat the specified number of times. ■ Unlimited — Indicates the reference will repeat an unlimited number of times.

Table 14–3 Reference Detail Window – Field Update Buttons

Field	Description
Apply	Saves your changes to the message format document.

Table 14-3 (Cont.) Reference Detail Window – Field Update Buttons

Field	Description
Edit Reference	Displays the detail window for the original item so you can edit the details of the referenced field or group.
Reset	Discards your changes to the detail window and resets all fields to the last saved values.
Help	Displays online help information for this detail window.

Note: The Apply and Reset buttons are enabled when changes are made to the components of the detail panel.

Comment Detail Window

Comments contain notes about the message format or the data transformed by the message format. Comments are included in the message format definition for informational purposes only. You can create a comment as a child or sibling of any message format, group, or field.

Table 15–1 *Comments about Message Format*

Field	Description
Comment Details	Enter the comment text.
Apply	Saves your changes to the message format document.
Reset	Discards your changes to the detail window and resets all fields to the last saved values.
Help	Displays online help information for this detail window.

Note: The Apply and Reset buttons are only enabled once changes are made to the detail panel's components.



Format Builder Options

Defines the options for Format Builder. These options control the overall operation of Format Builder. Click **Tools > Options** from the format builder menu to invoke the following options.

Table 16–1 *Format Builder Options*

Field	Definition
Default Message Format Version	Select the MFL version used when creating new documents. Note: Message formats contain their own format version specified on the Message Format pane.

Table 16–2 *Format Builder Options – Character Encoding Options*

Field	Definition
Default Message Format (MFL) Encoding	Select the character encoding default for the Message Format Layout (MFL) from the list of encoding names and descriptions. This defines the format that your MFL document and XML output will take.
Default Field Code Page	Select the default field code page from the list of non-XML formats. This selection will be the default code page for each field that is created in your MFL document. It specifies the character encoding of the non-XML data for each field.

Table 16–3 *Format Builder Options – XML Formatting Options*

Field	Definition
Initial Indent	Enter the number of spaces to indent the first line of the XML document.
New Line Indent	Enter the number of spaces to indent a new child line of the XML document.

Table 16–4 *Format Builder Options – XML Content Model Options*

Field	Definition
Auto-generate DTD	Generates a DTD document when you save or store the MFL document. This document will be placed in the same directory as the message format when saving to a file.



Importing Metadata

Format Builder includes the following utilities that allow you to import COBOL copybooks, gXML guideline files, and convert a C structure definition into MFL Message Definition.

- [Section 17.1, "Importing a Guideline XML File"](#)
- [Section 17.2, "Importing an XML Schema"](#)
- [Section 17.3, "Importing a COBOL Copybook"](#)
- [Section 17.4, "Importing C Structures"](#)
- [Section 17.8, "Importing an FML Field Table Class"](#)

17.1 Importing a Guideline XML File

Format Builder includes a feature that allows you to import a guideline XML (gXML) file and convert it into a message definition, which you can modify and customize to suit your needs. gXML is an open specification designed to facilitate exchange of e-commerce guidelines for business documents (like purchase orders, invoices and so on) using XML. gXML version 0.71 is supported in this release.

To import a gXML file:

1. Choose **Tools > Import > EDI Importer**. The EDI Importer dialog displays.
2. Enter data in the fields as described in the following table:

Table 17-1 *EDI Importer Options*

Field	Description
gXML File Name	Type the complete path and name of the gXML file that you want to import.
Browse	Click to navigate to the location of the gXML file you want to import.
OK Button	Imports the gXML file you specified.
Cancel Button	Closes the dialog and returns to Format Builder without importing.
About Button	Displays information about the EDI Importer including the version.

17.2 Importing an XML Schema

Format Builder includes a feature that allows you to import an XML Schema representing the desired XML representation of your non-XML document. This can provide you with a jump-start on specifying the format of your non-XML document.

To import an XML schema:

1. Choose **Tools > Import > XML Schema Importer**. The XML Schema Importer dialog displays.
2. Enter data in the fields as described in the following table:

Table 17–2 XML Schema Importer Options

Field	Description
XML Schema Definition	Type the path and name of the file you want to import.
Browse	Click to navigate to the location of the file you want to import.
Root Element	This value will be used as the root element in the transformed XML document. This name must comply with XML element naming conventions
MFL Field Delimiter Default	A delimiter is a character that marks the end of the field. The field data continues until the field containing the delimiter character is encountered.
OK Button	Imports the XML Schema using the settings you defined.
Cancel Button	Closes the dialog and returns to Format Builder without importing.

17.3 Importing a COBOL Copybook

Format Builder includes a feature that allows you to import a COBOL copybook into Format Builder and create a message definition to transform the COBOL data. When importing a copybook, comments are used to document the imported copybook and the Groups and Fields it contains.

To import a COBOL copybook:

1. Choose **Tools > Import > COBOL Copybook Importer**. The COBOL Copybook Importer dialog displays.
2. Enter data in the fields as described in the following tables:

Table 17–3 COBOL Copybook Importer Options

Field	Description
File Name	Type the path and name of the file you want to import.
Browse	Click to navigate to the location of the file you want to import.

Table 17–4 COBOL Copybook Importer Options – Byte Order

Field	Description
Big Endian	Select this option to set the byte order to Big Endian. Note: This option is used for IBM 370, Motorola, and most RISC designs (IBM mainframes and most Unix platforms).
Little Endian	Select this option to set the byte order to Little Endian. Note: This option is used for Intel, VAX, and Unisys processors (Windows, VMS, Digital, Unix, and Unisys).

Table 17–5 COBOL Copybook Importer Options – Character Set

Field	Description
EBCDIC	Select this option to set the character set to EBCDIC. Note: These values are attributes of the originating host machine.
US-ASCII	Select this option to set the character set to US-ASCII. Note: These values are attributes of the originating host machine.
Other	The character encoding of the field data.

Table 17–6 COBOL Copybook Importer Options – Action Buttons

Field	Description
OK	Imports the COBOL Copybook using the settings you defined.
Cancel	Closes the dialog and returns to Format Builder without importing.
About	Displays information about the COBOL Copybook importer including version and supported copybook features.

Once you have imported a copybook, you may work with it as you would any message format definition. If an error or unsupported data type is encountered in the copybook, a message is displayed informing you of the error. You can choose to display the error or save the error to a log file for future reference.

17.4 Importing C Structures

Format Builder includes a C structure importer utility that converts a C structure definition into an MFL Message Definition by generating MFL or C Code output.

- [Section 17.5, "Starting the C Structure Importer"](#)
- [Section 17.6, "Generating MFL Data"](#)
- [Section 17.7, "Generating C Code"](#)

17.5 Starting the C Structure Importer

To start the C Structure Importer:

1. From the Format Builder main window, choose **Tools > Import > C Struct Importer**. The C Structure Importer dialog displays.
2. The C Structure Importer dialog opens with MFL specified as the default output and contains the following fields.

Table 17–7 C Structure Importer Options – Input

Field	Description
Input File	Enter the path and name of the file you want to import. You can also click the Browse button to navigate to the file you want to import.
Structure	This list box is populated with the list of structures found in the input file after it is parsed. The list box is empty if the input file is not parsed.
Parse	Click Parse to parse the input file. If successful, the Structure list box is populated with the list of structures found in the input file.

Table 17–8 C Structure Importer Options – Output

Field	Description
Name	Specify an existing profile either by entering the file name or using the Browse button.
MFL	Specifies the data must be compiled on the target machine to generate MFL.
C Code	Specifies the data must be compiled on the target machine to generate C code.

17.6 Generating MFL Data

Perform the following steps to generate MFL data:

1. Enter a file name in the **Input File** field, or click **Browse** to select a file.
2. Click **Parse** to parse the file.

Upon completion, the Structure list box is populated with the list of structures found in the input file.

3. Select the desired structure from the Structure list box.

At this point, you must provide some profile configuration data to generate the MFL directly. You can do this by creating a new hardware profile, or specifying an existing profile.

4. Specify an existing profile or create a new one by performing one of the following procedures.
 - Specify an existing profile either by entering the file name in the Hardware Profile Name field, or click **Browse** to select a file. Click **Edit** to open the hardware profile editor if you need to view or edit the profile parameters.
 - Click **New** to create a new hardware profile. This opens the Hardware Profile editor loaded with the default parameters. Specify a Profile Name, a description, and modify the primitive data types and byte order to suit your needs.

5. Click **OK** to save your hardware profile changes and return to the C Structure Importer dialog.
6. Click **OK** to generate your MFL. If the generation is successful, you are returned to Format Builder with an MFL object listed in the navigation tree. The MFL object reflects the same name as the input file used in the parse operation.

If errors are detected during the generation process, the MFL Generation Errors dialog displays providing you the opportunity to view or file the error log.

7. Click **Display Error Log** to view any errors encountered, click **Save Error Log** to save the error log to the location of your choice, or click **Cancel** to dismiss the MFL Generation Errors dialog box.

Once you have determined what errors were generated, you can return to the C Structure Importer and repeat the prior steps.

17.7 Generating C Code

Perform the following steps to generate C code.

1. Enter a file name in the **Input File** field, or click **Browse** to select a file.

2. Click **Parse** to parse the file.
Upon completion, the Structure list box is populated with the list of structures found in the input file.
3. Select the desired structure from the **Structure** list box.
4. Select the **C Code** option button.
5. Enter a file name in either the **MFL Gen** or **Data Gen** fields, or click **Browse** to select a file.
6. Click **OK**. You will be warned about overwriting existing files and notified about the success or failure of the code generation.
7. Copy the generated source code to the platform in question and compile and execute it.

Note: You must copy the input file containing the structure declarations as well. Both programs, when compiled, take an argument of the output file name.

8. Copy the generated MFL or data back to the platform running Format Builder.

17.8 Importing an FML Field Table Class

The FML Field Table Class Importer facilitates the integration of WebLogic Tuxedo Connector and business process management (BPM) functionality. Tuxedo application buffers are translated to and from XML by the FML to XML Translator that is a feature of WebLogic Tuxedo Connector.

The integration of Tuxedo with BPM functionality requires the creation of the XML that is passed between the WebLogic Tuxedo Connector Translator and the process engine. To create the necessary XML, use the FML Field Table Class Importer and the XML generation feature of Format Tester.

17.8.1 FML Field Table Class Importer Prerequisites

Before starting Format Builder:

1. Move the field tables associated with the FML buffer from the Tuxedo system to the Oracle WebLogic Server/WebLogic Tuxedo Connector environment.
2. Use the `weblogic/wtc/jatmi/mkfldclass` utility to build Java source code representing the field tables. For information about FML Field Table Administration, see the Oracle WebLogic Server documentation.
3. Compile the source code. The resulting class files are called `fldtbl` classes because they implement the `FldTbl` interface. These class must be packaged in a JAR file that can be selected from the FML Field Table Class Importer dialog.

The `WLI_HOME\samples\di\fm1` directory contains several `fldtbl` class fields that you can use as samples. These samples allow you to start Format Builder without having to completing the previous three steps.

Note: Because most users perform these steps when configuring WebLogic Tuxedo Connector, these class files may already exist.

17.8.2 Sample FML Field Table Class Files

The following table provides a listing and descriptions of the sample files installed for the FML Field Table Class Importer. All files are in the `WLI_HOME\samples\di\fml` directory.

Table 17–9 FML Field Table Class Sample Files

Field	Description
<code>bankflds.class</code>	Compiled source file that serves as input to the FML Field Table Class Importer
<code>bankflds.java</code>	<code>fldtbl</code> source file generated by the <code>mkfldclass</code> utility
<code>crdtflds.class</code>	Compiled source file that serves as input to the FML Field Table Class Importer
<code>crdtflds.java</code>	<code>fldtbl</code> source file generated by the <code>mkfldclass</code> utility
<code>tBtest1flds32.class</code>	Compiled source file that serves as input to the FML Field Table Class Importer
<code>tBtest1flds32.java</code>	<code>fldtbl</code> source file generated by the <code>mkfldclass</code> utility

17.8.3 Creating XML with the FML Field Table Class Importer

Note: If you create Java classes using WebLogic Tuxedo Connector, you can place the `.class` files in the `\ext` directory. You can then populate the Available Fields list automatically from the FML Field Table Class Importer dialog box.

To create an XML document with the FML Field Table Class Importer:

1. Choose **Tools > Import > EDI Importer**. The FML Field Table Class Importer dialog displays.
2. Enter data in the fields as described in the following table:

Table 17–10 FML Field Table Class Importer Options

Field	Description
Fld Table Jar File	<p>Click Select to select the JAR file containing the <code>fldtbl</code> classes. After selecting the JAR file, all <code>fldtbl</code> classes are displayed in the Classes list. If the selected JAR file contains no <code>fldtbl</code> classes, an error message is displayed and the Fld Table Jar File and Classes fields are cleared.</p> <p>The Classes section contains a list of all <code>fldtbl</code> classes for the currently selected JAR file. Because a single FML buffer may contain fields from several field tables, you can select one or more <code>fldtbl</code> class names in the list. All the fields in the selected classes are displayed in the Available Fields list.</p>

Table 17–11 FML Field Table Class Importer Options – FML Field Selector

Field	Description
Available Fields	Displays the list of names from the field tables. Select the desired fields from the Available Fields list and click Add. The Available Fields list does not allow duplicate names. Even if the name of a field appears in different field tables, it is included only once in the list.
Selected Fields	Displays the list of selected fields. To remove fields from this list, select the fields and click Remove .

Table 17–12 FML Field Table Class Importer Options – Action Buttons

Field	Description
Add	Moves the selected field from the list of Fields Available, to the Selected Fields list.
Remove	Removes the selected field from the list of Selected Fields, to the Fields Available list.
OK	Click OK after completing the list of selected fields. The dialog box closes and the name of the generated MFL is added to the Format Builder navigation tree. The selected fields are listed in the order in which they appear in the Selected Fields list.
Cancel	Closes the dialog and returns to Format Builder without importing.

3. Edit the created MFL document to specify the order and number of occurrences of the fields in the XML document to be passed to the WebLogic Tuxedo Connector FML/XML Translator from business process management (BMP).
4. Choose **Tools > Test** to display the Format Tester tool.
5. From the Format Tester menu bar, choose **Generate > XML** to create an XML document that conforms to the MFL document in Format Builder.
6. Edit the data content of the fields in the XML document as desired.
7. From the Format Tester menu bar, choose **File > Save XML** to save the XML document in a file with a specified name and location.

The created XML can be imported and used in business process management functions by using the XML instance editor. For information about importing XML, see the BPM documentation.

Once you have build a format definition, you can test it using Format Tester. Format Tester parses and reformats data as a validation test and generates sample non-XML or XML data. This sample data can be edited, searched, and debugged to product the expected results.

- [Section 18.1, "Format Tester Window"](#)
- [Section 18.2, "Format Tester Menus"](#)

18.1 Format Tester Window

The following topics discuss the elements of the Format Tester main window and provide instructions for navigating and executing commands from the Format Tester main window.

- [Section 18.9, "Using the Non-XML Window"](#)
- [Section 18.12, "Using the XML Window"](#)
- [Section 18.13, "Using the Debug Window"](#)
- [Section 18.14, "Using the Resize Bars"](#)

18.2 Format Tester Menus

The following menus are available in Format Tester. All Format Tester menus are expandable from your keyboard by pressing Alt + *underlined letter*. Some menu commands are also executable using Ctrl + *letter* keystrokes.

- [Section 18.3, "File Menu"](#)
- [Section 18.4, "Edit Menu"](#)
- [Section 18.5, "Display Menu"](#)
- [Section 18.6, "Generate Menu"](#)
- [Section 18.7, "Transform Menu"](#)
- [Section 18.8, "Shortcut Menu"](#)

18.3 File Menu

The following commands are available from the File menu.

Table 18–1 File Menu Commands

Menu Command	Description
Open Non-XML	Allows you to select a non-XML file to be displayed in the Non-XML window. Note: The default file extension for non-XML files is .DATA.
Open XML	Allows you to select a file to be displayed in the XML section of the Format Tester window. Note: The default file extension for XML files is .XML.
Save Non-XML	Allows you to save the contents of the Non-XML window.
Save XML	Allows you to save the contents of the XML window.
Debug Log	Allows the debug information to be saved in a text file.
Close	Closes the Format Tester window.

18.4 Edit Menu

The following commands are available from the Edit menu.

Table 18–2 Edit Menu Commands

Menu Command	Description
Cut	Removes the currently selected text and places it on the clipboard for pasting into another location.
Copy	Copies the currently selected text and places it on the clipboard for pasting into another location.
Paste	Inserts the cut or copied text at the cursor location.
Find	Allows you to search for a hex or text value in the non-XML data.
Find Next	Continues your search to the next instance of the specified value.
Go To	Allows you to move the cursor in the Non-XML editor to a specified byte offset.

18.5 Display Menu

The following commands are available from the Display menu.

Table 18–3 Display Menu Commands

Menu Command	Description
XML	Allows the XML data panel to be hidden or shown. If hidden, the non-XML data window expands to fill the width of the tester. The <i>To XML</i> button remains, but the splitter disappears.
Debug	Allows the Debug output window to be hidden or shown.
Clear > Non-XML	Resets the contents of the Non-XML data window to be empty.
Clear > XML	Resets the contents of the XML window to be empty.
Clear > Debug	Resets the contents of the debug window to be empty.
Hex > Offsets as Hexadecimal	Displays the offset values as hexadecimal. Selecting this option turns off the <i>Offsets as Decimal</i> display.
Hex > Offsets as Decimal	Displays the offset values as decimal. Selecting this option turns off the <i>Offset as Hexadecimal</i> display.

18.6 Generate Menu

The following commands are available from the Generate menu.

Table 18–4 *Generate Menu Commands*

Menu Command	Description
Non-XML	Generates non-XML data to match the current format specification.
XML	Generates XML data to match the current format specification.
Prompt while generating data	If selected, you are prompted during the generation process to determine if optional fields or groups should be generated, determine which choice of children should be generated, and determine how many times a repeating group should repeat.

18.7 Transform Menu

The following commands are available from the Transform menu.

Table 18–5 *Translate Menu Command*

Menu Command	Description
Non-XML to XML	Converts the contents of the Non-XML window to XML.
XML to Non-XML	Converts the contents of the XML window to non-XML data.

18.8 Shortcut Menu

Instead of using the standard menus to find the command you need, use the right mouse button to click an item in the pop-up shortcut menu.

The following commands are available from the Shortcut menu.

Note: Some commands may be unavailable, depending on which display panel currently contains the mouse pointer.

Table 18–6 *Shortcut Menu Commands*

Menu Command	Description
Cut	Removes the currently selected text and places it on the clipboard for pasting into another location.
Copy	Copies the currently selected text and places it on the clipboard for pasting into another location.
Paste	Inserts the cut or copied text at the cursor location.
Clear	Resets the contents of the Non-XML or XML window to be empty.
Generate	Generates Non-XML or XML data to match the current format specification.
To XML	Converts the contents of the Non-XML window to XML.
To Non-XML	Converts the contents of the XML window to non-XML data.

18.9 Using the Non-XML Window

The Non-XML data display panel acts as a hexadecimal editor or a text editor, depending on which tab is selected.

The hexadecimal editor panel displays data offsets, the hex value of individual bytes, and the corresponding text. The corresponding text can be optionally displayed as ASCII or EBCDIC characters. The editor allows for editing of the hex byte or the text value. If a hex data value is modified, the corresponding text value is updated, and vice versa.

18.9.1 See Also

- [Section 18.10, "Using the Data Offset Feature"](#)
- [Section 18.11, "Using the Text Feature"](#)

18.10 Using the Data Offset Feature

The data offset feature of the hexadecimal editor allows you to display your data offsets as Hexadecimal or Decimal.

To change your data offsets:

Choose **Display > Hex**. The following two data offset options display.

- Offsets as Hexadecimal
- Offsets as Decimal

Click the display option that best suits your needs. The data offset panel of the Non-XML window dynamically changes to reflect your choice.

18.11 Using the Text Feature

To use the Text feature, select the Text tab from within the Non-XML window to view all printable characters, such as carriage returns. The Text window shows these as text with line breaks.

18.12 Using the XML Window

The XML data panel displays XML data that has been converted or transformed from the contents of the Non-XML panel. The contents of the XML panel can be cleared or edited to suit your needs.

You can also use this window to enter or generate the XML data to be transformed into non-XML format.

18.13 Using the Debug Window

The Debug window displays the actions that take place during the transformation operation, any errors that are encountered, and field and group values and delimiters. To determine the location of the error, determine the last field that parsed successfully and examine the specification of the next field on the [Section 10.3, "Using the Tree Pane"](#) of Format Builder.

When you open the Format Tester, only the Non-XML and XML windows are visible. To open the Debug window, choose **Display > Debug** to toggle the Debug window on and off. The Debug window opens below the Non-XML and XML windows.

Note: Debug output is restricted to the most recent 64 KB of messages. Full debug information can be captured to a file. See [Section 18.18, "Using the Debug Log"](#) for more information.

18.14 Using the Resize Bars

Resize bars are located between the Non-XML, XML, and Debug windows. These resize bars enable each window to be resized to suit your needs. Each resize bar can be selected and dragged up and down, or left and right, as appropriate, to enlarge one of the windows and reduce the other.

Each resize bar also contains two directional buttons that can be clicked to enlarge or diminish any of the three windows.

18.15 Debugging Format Definitions

The following topics discuss the various Format Tester utilities you can use to debug and correct your data.

- [Section 18.16, "Searching for Values"](#)
- [Section 18.17, "Searching for Offsets"](#)
- [Section 18.18, "Using the Debug Log"](#)

18.16 Searching for Values

The Find feature allows you to search for hex or text values in the Non-XML data. The following fields are available from the Find dialog.

Table 18-7 Find Options

Field	Description
Value	Enter the value you want to find.
Text	Select this option if you want to find a text value.
Hex	Select this option if you want to find a hex value.
Forwards	Select this option if you want to search from the selected location to the end of the document.
Backwards	Select this option if you want to search from the selected location to the beginning of the document.
Beginning of File	Select this option if you want to start the search at the beginning of the file.
Current Position	Select this option if you want to start the search at the current cursor location.
End of File	Select the option if you want to start the search at the end of the file.
OK Button	Begins the search operation.
Cancel Button	Closes the Find dialog without performing a search.

18.17 Searching for Offsets

The Goto feature allows you to move the cursor in the Non-XML editor to a byte offset you specify. The following fields are available from the Goto dialog.

Table 18–8 *Goto Options*

Field	Description
Offset	Enter the offset value you want to find.
Dec	Select this option if you want to go to a decimal value.
Hex	Select this option if you want to go to a hex value.
Forwards	Select this option if you want to search from the selected location to the end of the document.
Backwards	Select this option if you want to search from the selected location to the beginning of the document.
Beginning of File	Select this option if you want to start the search at the beginning of the file.
Current Position	Select this option if you want to start the search at the current cursor location.
End of File	Select the option if you want to start the search at the end of the file.
OK Button	Begins the search operation.
Cancel Button	Closes the Goto dialog without performing a search.

18.18 Using the Debug Log

The debug log allows you to save your debug information to a text file.

To use the debug log, choose **File > Debug Log**. A dialog displays allowing you to enter a new path and file name or choose an existing file in which to save the debug information.

Note: If you select an existing file, the new debug information is appended to the end of the file.

Format Builder Menus

Format Builder provides the following menus:

- [Section 19.1, "File Menu"](#)
- [Section 19.2, "Edit Menu"](#)
- [Section 19.3, "Insert Menu"](#)
- [Section 19.4, "View Menu"](#)
- [Section 19.5, "Tools Menu"](#)
- [Section 19.7, "Shortcut Menu"](#)

19.1 File Menu

The following commands are available from the File Menu.

Table 19–1 File Menu Commands

Menu Command	Description
New	Creates a new Message Format document.
Open	Opens an existing Message Format document.
Close	Closes the current Message Format document.
Save	Saves the current Message Format document.
Save As	Saves the current Message Format under a different name document.
Properties	Opens the Properties window for the selected file or message format. You can change the MFL encoding or MFL version for the selected item.
Exits	Closes the application.

19.2 Edit Menu

The following commands are available from the Edit Menu.

Note: Some commands may be unavailable, depending on the actions you have taken.

Table 19–2 Edit Menu Commands

Menu Command	Description
Undo	Reverses the previous action. The Undo command in the Edit Menu changes to indicate the action that can be undone. For example, changing the name of a field to Field1 and clicking Apply causes the Edit Menu to read "Undo Apply Field Field1".
Redo	Reverses the effects of an Undo command. The Redo command in the Edit Menu changes to indicate the action that can be redone. For example, changing the name of a field to Field1 and then undoing that action causes the Edit Menu to read "Redo Apply Field Field1".
Cut	Removes the item currently selected in the left-hand pane, and its child objects, from the tree. This action is not available if the Message Format (root) item is selected.
Copy	Makes a copy of the item currently selected in the left-hand pane for insertion elsewhere in the tree. Note: This action is not available if the Message Format (root) item is selected.
Paste	Inserts the copied item. An additional menu is displayed when you select Paste. You can choose to paste the item as a child or sibling of the selected item. In addition, you can choose to paste a reference as a sibling of the selected item.
Duplicate	Makes a copy of the item selected in the tree. The duplicate item contains the same values as the original item. The name of the duplicate item is the same as the original item name, but the word "New" is inserted before the original name. For example, duplicating an item called "Field1" results in an item with the name "NewField1". When you duplicate an item with a numeric value in its name, the new item name contains the next sequential number. For example, duplicating "NewGroup1" results in a group named "NewGroup2".
Delete	Deletes the item selected in the tree, as well as all child objects of that item.
Move Up	Moves the selected item up one position under its parent.
Move Down	Moves the selected item down one position under its parent.
Promote	Promotes the selected item to the next highest level in the tree. For example, Field1 is the child object of Group1. Selecting Field1 and clicking the Promote tool makes it a sibling of Group1.
Demote	Demotes the selected item to the next lower level in the tree. For example, Group1 is the sibling of Field1. Field1 immediately follows Group1 in the tree. Selecting Field1 and clicking the Demote tool makes it a child of Group1.

19.3 Insert Menu

The following commands are available from the Insert Menu.

Table 19–3 Insert Menu Commands

Menu Command	Description
Field	Inserts a new field. You can choose whether to insert the field as a child or sibling of the item selected in the tree.

Table 19-3 (Cont.) Insert Menu Commands

Menu Command	Description
Group	Inserts a new group. You can choose whether to insert the group as a child or sibling of the item selected in the tree.
Comment	Inserts a comment. You can choose whether to insert the comment as a child or sibling of the item selected in the tree.

19.4 View Menu

The following commands are available from the View Menu.

Table 19-4 View Menu Commands

Menu Command	Description
Show Palette	Displays the palette window. For more information on the palette, see Chapter 21, "Using the Palette"
Expand All	Expands the entire tree pane to show the child objects of all items in the tree.
Collapse All	Collapses the entire tree pane to show only the root message format.

19.5 Tools Menu

The following commands are available from the Tools Menu.

Table 19-5 Tools Menu Commands

Menu Command	Description
Import	Displays a list of the installed importers. Choose the importer from which you want to import a message.
Test	Opens the Format Tester. Refer to Chapter 18, "Format Tester" for more information.
Options	Displays the Format Builder Options dialog. Refer to Chapter 16, "Format Builder Options" for more information.

19.6 Help Menu

The following commands are available from the Help Menu.

Table 19-6 Help Menu Commands

Menu Command	Description
Help Topics	Displays the main Help screen.
How Do I	Provides step-by-step instructions for performing the basic tasks in Format Builder.
About	Displays version and copyright information about Format Builder.

19.7 Shortcut Menu

The following commands are available from the Shortcut Menus. You can access the Shortcut Menus by right-clicking an item in the tree.

Note: Some commands may be unavailable, depending on the item you have selected in the tree.

Table 19–7 *Shortcut Menu Commands*

Menu Command	Description
Cut	Removes the item currently selected in the left-hand pane, and its child objects, from the tree.
Copy	Makes a copy of the item currently selected in the left-hand pane for insertion elsewhere in the tree.
Paste	Inserts the cut or copied item. An additional menu displays when you select Paste. You can choose to paste the item as a child or sibling of the selected item. In addition, you can choose to paste a reference to the cut or copied item as a sibling of the selected item.
Insert Group	Inserts a new group. You select whether to insert the group as a child or sibling of the selected item.
Insert Field	Inserts a new field. You select whether to insert the field as a child or sibling of the selected item.
Insert Comment	Inserts a comment. You select whether to insert the comment as a child or sibling of the selected item.
Duplicate	<p>Makes a copy of the currently selected item. The duplicate item contains the same values as the original item. The name of the duplicate item is the same as the original item name, with the word "New" inserted before the original name. For example, duplicating a group called "Group1" results in a group with the name "NewGroup1".</p> <p>When you duplicate an item with a numeric value in its name, the new item name contains the next sequential number. For example, duplicating "NewGroup1" results in a group named "NewGroup2".</p>
Delete	Deletes the selected item.

This section contains the following topics:

- [Section 20.1, "Create a Message Format"](#)
- [Section 20.2, "Create a Group"](#)
- [Section 20.3, "Create a Field"](#)
- [Section 20.4, "Create a Comment"](#)
- [Section 20.5, "Create a Reference"](#)
- [Section 20.6, "Save a Document"](#)
- [Section 20.7, "Use Format Tester"](#)
- [Section 20.8, "Debug Format Definitions"](#)
- [Section 20.9, "Search for Values"](#)
- [Section 20.10, "Search for Offsets"](#)
- [Section 20.11, "Use the Debug Log"](#)
- [Section 20.12, "Character Delimiters"](#)
- [Section 20.13, "Specify a Delimiter"](#)
- [Section 20.14, "Delimiter Match Rule"](#)

20.1 Create a Message Format

The first step in creating a Message Format Definition file is to create a message format (the root node of a message format file).

To create a message format:

1. Choose **File > New**. The [Message Format Detail Window](#) displays in the right pane.
2. Enter data in the fields as appropriate. See [Message Format Detail Window](#) for field definitions.

20.2 Create a Group

Groups define fields that are related in some way (for example, the fields PAYDATE, HOURS, and RATE could be part of the PAYINFO group). You can create a group as a child of the message format item, as a child of another group, or as a sibling of a group or field.

To create a group:

1. Select an item in the tree view in the left pane.
2. Choose **Insert > Group > As Child** if you want to create the group as the child of the message format or another group. Choose **Insert > Group > As Sibling** if you want to create the group as the sibling of another group or a field. The Group Details window displays in the right pane.
3. Enter data in the fields as appropriate. See [Group Detail Window](#) for field definitions.
4. Click **Apply** to save your changes to the message format file, or click **Reset** to discard your changes to the detail window and reset all fields to the last saved value.

Note: The Apply and Reset buttons are only enabled once changes are made to the detail panel's components.

20.3 Create a Field

Fields are a sequence of bytes that have some meaning to an application. (For example, the field `EMPNAME` contains an employee name.) You can create a field as a child of the message format item, as a child of a group, or as a sibling of a group or another field.

To create a field:

1. Select an item in the tree view in the left pane.
2. Choose **Insert > Field > As Child** if you want to create the field as the child of the message format or group. Choose **Insert > Field > As Sibling** if you want to create the group as the sibling of another group or a field. The Field Details window displays in the right pane.
3. Enter data in the fields as appropriate. See [Field Detail Window](#) for field definitions.
4. Click **Apply** to save your changes to the message format file, or click **Reset** to discard your changes to the detail window and reset all fields to the last saved value.

Note: The Apply and Reset buttons are only enabled once changes are made to the detail panel's components.

20.4 Create a Comment

Comments contain notes about the message format or the data transformed by the message format. Comments are included in the message format definition for informational purposes only. You can create a comment as a child or sibling of any message format, group, or field.

To create a comment:

1. Select an item in the tree view in the left pane.
2. Choose **Insert > Comment > As Child** if you want to create the comment as the child of the selected item. Choose **Insert > Comment > As Sibling** if you want to create the comment as the sibling of the selected item. The Comment Details window displays in the right pane.

3. Enter the desired comment text. For more information, see [Comment Detail Window](#).
4. Click **Apply** to save your changes to the message format file, or click **Reset** to discard your changes to the detail window and reset all fields to the last saved value.

Note: The Apply and Reset buttons are only enabled once changes are made to the detail panel's components.

20.5 Create a Reference

References indicate that the description of the field or group format has been previously defined and you want to reuse this description without re-entering the data. Reference fields or groups have the same format as the original field or group, but you can change only the optional setting and the occurrence setting for the reference field or group. For example, if you have a "bill to" address and a "ship to" address in your data and the format for the address is the same, you only need to define the address format once. You can create the "bill to" address definition and create a reference for the "ship to" address.

Note: References are named exactly the same as the original item. For example, the "bill to" address definition and the "ship to" address definition would be named the same. If you want to reuse a group definition, create a generic group and embed it within a specific group. For example, in the previous example, you can create an *address* group within a *bill_to* group and reference *address* within a *ship_to* group.

To create a reference:

1. Select a field or group in the tree pane.
2. Choose **Edit > Copy**.
3. Choose the proper sibling in the tree.
4. Choose **Edit > Paste > As Reference**.
5. Enter data in the fields as appropriate. See [Section 14, "Reference Detail Window"](#) for field definitions.
6. Click **Apply** to save your changes to the message format file, or click **Reset** to discard your changes to the detail window and reset all fields to the last saved value.

Note: The Apply and Reset buttons are only enabled once changes are made to the detail panel's components.

20.6 Save a Document

To save a message format file for the first time:

1. Choose **File > Save As**. The Save As dialog displays.
2. Navigate to the directory where you want to save the file.

3. In the **File Name** text box, type the name you want to assign to the file.
4. If no extension has been given, Format Builder automatically assigns the extension `.MFL` to message format files.
5. Click **Save As** to save the file in the specified location with the specified name and extension.

To save a message format file using the same name, choose **File > Save**. The file is saved in the same location with the same name and extension.

To save a message format file using a different name, choose **File > Save As** and follow steps 1 through 5 above.

20.7 Use Format Tester

Format Tester parses and reformats data as a validation test. Using Format Tester, you can make sure the message formats you build using Format Builder produce the expected results.

Format Tester is available from Format Builder.

To start Format Tester:

1. In Format Builder, open a message format document (MFL file).

Note: To run Format Tester, you must have a message format document open in Format Builder.

2. From the **Format Builder** menu bar, choose **Tools > Test**.
3. The Format Tester dialog box is displayed.

Note: Format Tester uses the currently loaded message definition document.

Refer to [Section 18, "Format Tester"](#) for more information.

20.8 Debug Format Definitions

- [Section 20.9, "Search for Values"](#)
- [Section 20.10, "Search for Offsets"](#)
- [Section 20.11, "Use the Debug Log"](#)

20.9 Search for Values

The find feature allows you to search for hex or text values in the non-XML data.

To search for values:

1. From within the Format Tester, choose **File > Open Non-XML** to open the non-XML data file you want to search.
2. Choose **Edit > Find**. The Find dialog opens.
3. Enter data in the fields as appropriate (see [Section 20.9, "Search for Values"](#) for more information).

4. Click **OK** to begin the Search operation.

20.10 Search for Offsets

The Goto feature allows you to move the cursor in the Non-XML editor to a byte offset you specify.

To move to a specified offset:

1. Choose **Edit > Find**. The Goto dialog opens.
2. Enter data in the fields as appropriate (see [Section 20.10, "Search for Offsets"](#) for more information).
3. Click **OK** to begin the Search operation.

20.11 Use the Debug Log

The debug log allows you to save your debug information to a text file.

To use the debug log, choose **File > Debug Log**. A dialog displays allowing you to enter a new path and file name or choose an existing file in which to save the debug information.

Note: If you select an existing file, the new debug information is appended to the end of the file.

20.12 Character Delimiters

You can specify delimiters in Format Builder by entering the correct syntax. For example, to specify a tab character as the delimiter ('`\u009`'), enter the construct `\t` to match it.

Table 20–1 Character Delimiters

Construct	Matches
<code>x</code>	The character <code>x</code>
<code>\\</code>	The backslash
<code>\0n</code>	The character with octal value <code>0n</code> ($0 \leq n \leq 7$)
<code>\0nn</code>	The character with octal value <code>0nn</code> ($0 \leq n \leq 7$)
<code>\0mnn</code>	The character with octal value <code>0mnn</code> ($0 \leq m \leq 3, 0 \leq n \leq 7$)
<code>\xhh</code>	The character with hexadecimal value <code>0xhh</code>
<code>\uhhhh</code>	The character with hexadecimal value <code>0xhhhh</code>
<code>\t</code>	The tab character (' <code>\u009</code> ')
<code>\n</code>	The newline (line feed) character (' <code>\u000A</code> ')
<code>\r</code>	The carriage-return character (' <code>\u000D</code> ')
<code>\f</code>	The form-feed character (' <code>\u000C</code> ')
<code>\a</code>	The alert (bell) character (' <code>\u0007</code> ')
<code>\e</code>	The escape character (' <code>\u001B</code> ')
<code>\cx</code>	The control character corresponding to <code>x</code>

For more information, see

<http://java.sun.com/j2se/1.4/docs/api/java/util/regex/Pattern.html>.

20.13 Specify a Delimiter

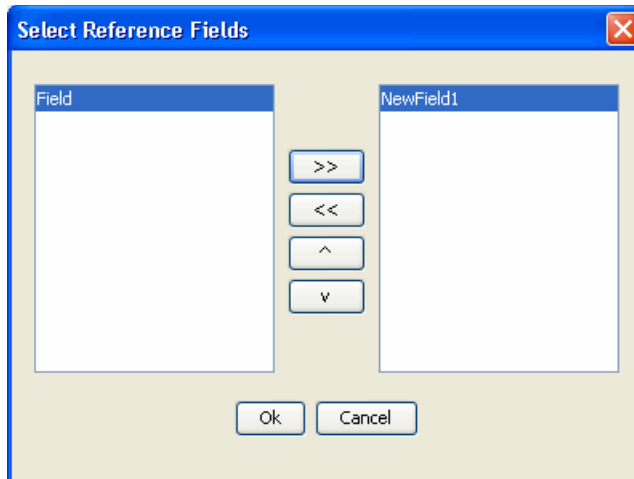
You can specify a delimiter either by reference or by value.

Variable-sized data types can have their termination point specified by a delimiter. A delimiter is a character that marks the end of the field. The field data continues until the delimiter character is encountered.

20.13.1 Specify by Reference

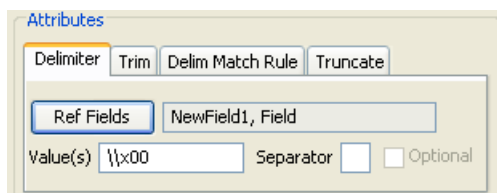
Specify one or more fields that contain the delimiter character. The delimiter is matched in the order the fields are listed. Clicking the **Ref Fields** button opens the **Select Reference Fields** dialog box. In this dialog box, you can select reference fields and set the order in which the delimiter is matched.

Figure 20–1 Select Reference Fields



In [Figure 20–1](#), The left pane lists fields that can be used as reference. Select a field by clicking on it and clicking the double arrow button to add the field. After selecting the fields, you can arrange the order of the fields by selecting a field on the right pane and clicking on the up arrow or down arrow button to move the field up or down. Click **OK**. The selected fields names are displayed in the field to the right of the **Ref Fields** button. See [Figure 20–2](#).

Figure 20–2 Reference Field



20.13.2 Specify by Value

Enter the delimiter or delimiters separated by the specified Separator character. When both **Ref Fields** and **Values** are specified, the Ref Fields take precedence. If no delimiter match is found, the specified delimiter values are used to match the delimiter. Example values for Value(s) and Separator are:

Values = ,|~|;

Separator = |

The field delimiter can be comma, tilde, or semi-colon.

If the field is optional, the **Optional** check box is enabled and selected by default. To ensure that the binary data contain the delimiter even if the field is not present, you must clear the optional check box.

20.14 Delimiter Match Rule

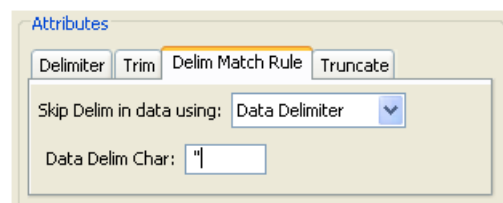
You can specify rules to skip delimiters that occur inside the field data, so they are treated as field data instead of a delimiter. For instance, if the delimiter character is ;(semicolon), but the field data is A;B, the ; character occurring between A and B must not be considered as delimiter, but a part of field data. You can specify the following options to specify the delimiter match rule:

- Data Delimiter
- Escape Character
- None

20.14.1 Data Delimiter

Specify the character that appears at the beginning and at the end of the field data. For example, if **Data Delim Char** is "(double quotation mark), the data appears as "A;B";. See [Figure 20–3](#).

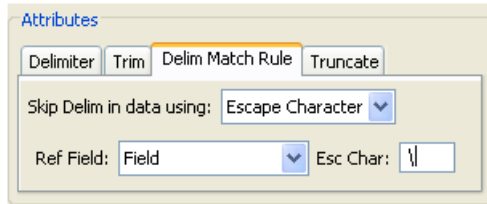
Figure 20–3 Data Delimiter



20.14.2 Escape Character

Specify an escape character that precedes the delimiter character occurring as part of the field data. The escape character value can be obtained via a reference field and by specifying the value in the **Esc Char** field. For example, if the **Esc Char** is \, then the data appears as A\;B;. See [Figure 20–4](#)

Figure 20–4 Escape Character



Note: The field delimiter, and all the shared delimiters of the parent structure are skipped even if the field is not the last field in the structure. For example, consider the following MFL structure:

```

Root Element
Group1 with shared delimiter *
  Field1 with delimiter ,
  Field2 with delimiter "
Group2 with shared delimiter ~
  Field3 with delimiter ,
Optional field4 with delimiter ~
    
```

In this example, Field3 data might contain *, ~ or, that are skipped by the specified escape character even if it is not the last field in Group2

20.14.3 None

Select this option if the delimiter match rule is not required.

Using the Palette

The Format Builder palette allows you to store commonly used message format items and insert them into your message format definitions. These items are stored in an XML document, and you can use the standard Windows drag and drop feature to copy items from the palette into your message format definition.

The palette contains some common date formats, literals, and strings. You can use these items in the message formats you create, as well as adding your own items to the palette.

- [Section 21.1, "Displaying the Palette Window"](#)
- [Section 21.2, "Adding Items to the Palette"](#)
- [Section 21.3, "Adding Palette Items to a Message Format"](#)
- [Section 21.4, "Using the File Menu"](#)
- [Section 21.5, "Using the Shortcut Menu"](#)

21.1 Displaying the Palette Window

To turn the palette display on or off, choose **View > Show palette**. If the palette is not currently displayed, it opens in a separate window next to the Format Builder window. If the palette is currently displayed, its window closes.

21.2 Adding Items to the Palette

To add items to the palette:

1. From the navigation tree, choose the item you want to add to the palette.
2. Click and hold the left mouse button and drag the item into the palette window.
3. When the item is placed in the position you want it (as a sibling of the selected item), release the mouse button. The item is copied from the navigation tree to the palette window.

Note: You cannot add any node that depends on the existence of another node to the palette. For example, you cannot add Field or Group References, and you cannot add items that have a Repeat Field specified.

Adding comments is possible, but not recommended since comments do not have unique names and therefore are indistinguishable on the palette.

21.3 Adding Palette Items to a Message Format

To copy items from the palette to a message format:

1. From the palette window, choose the item you want to add to your message format.
2. Click and hold the left mouse button and drag the item into the left pane of the Format Builder window.
3. When the item is placed in the position you want it (as the child or sibling of the desired item), release the mouse button. The item is copied from the palette to the message format.

21.4 Using the File Menu

The following commands are available from the palette's File menu.

Table 21–1 File Menu Commands

Menu Command	Description
Open	Opens an existing message format.
Save	Saves any message format items you have added to the palette, or any existing items you have modified.
Hide palette	Closes the palette window.

21.5 Using the Shortcut Menu

The following commands are available from the palette's shortcut menu. You can access the shortcut menu by right-clicking within the palette window.

Note: Some commands may be unavailable, depending on the item you have selected in the tree.

Table 21–2 Shortcut Menu Commands

Menu Command	Description
Insert	Inserts a new group in the palette. When you select this command, a window displays asking you to supply the name of the new group.
Rename	Displays a window asking you to supply the new name of the group.
Delete	Deletes the selected item.
Move Up	Moves the selected item up one position under its parent.
Move Down	Moves the selected item down one position under its parent.
Promote	Promotes the selected item to the next highest level in the tree. For example, Field1 is the child object of Group1. Selecting Field1 and clicking the Promote tool makes it a sibling of Group1.
Demote	Demotes the selected item to the next lower level in the tree. For example, Group1 is the sibling of Field1. Field1 immediately follows Group1 in the tree. Selecting Field1 and clicking the Demote tool makes it a child of Group1.

Format Builder Supported Data Types

This section provides information about the following topics:

- [Section 22.1, "MFL Data Types"](#) – This section lists the supported metadata data types used in non-XML to XML or XML to non-XML conversions.
- [Section 22.2, "COBOL Copybook Importer Data Types"](#) – The Format Builder tool provides a utility for the conversion of COBOL copybooks into MFL files. This section lists the supported COBOL data types that can be converted to metadata data types. (This conversion occurs at design time.)
- [Section 22.3, "Unsupported C Language Features"](#) – The Format Builder tool provides a utility for the conversion of C Structures into MFL files. This section lists the C Language constructs that *cannot* be converted to metadata data types. (This conversion occurs at design time.)

22.1 MFL Data Types

[Table 22-1](#) lists the MFL data types that data transformer supports. These types are specified in the "type" attribute of a FieldFormat element.

Table 22-1 Supported MFL Data Types

Data Type	Description
Binary (Base64 encoding)	Any character value accepted. Requires a length, length field, delimiter, or a delimiter field. Resulting XML data for this field is encoded using base-64.
Binary (Hex encoding)	Any character value accepted. Requires a length, length field, delimiter, or a delimiter field. Resulting XML data for this field is encoded using base-16.
Date: DD-MMM-YY	A string defining a date, i.e. 22-JAN-00.
Date: DD-MMM-YYYY	A string defining a date, i.e. 22-JAN-2000.
Date: DD/MM/YY	A string defining a date, i.e. 22/01/00.
Date: DD/MM/YYYY	A string defining a date, i.e. 22/01/2000.
Date: DDMMMYY	A string defining a date, i.e. 22JAN00.
Date: DDMMMYYYY	A string defining a date, i.e. 22JAN2000.
Date: MM/DD/YY	A string defining a date, i.e. 01/22/00.
Date: MM/DD/YYYY	A string defining a date, i.e. 01/22/2000.
Date: MMDDYY	A six digit numeric string defining a date, i.e. 012200.
Date: MMDDYYYY	An eight digit numeric string defining a date, i.e. 01222000.

Table 22–1 (Cont.) Supported MFL Data Types

Data Type	Description
Date: MMM-YY	A string defining a date, i.e. JAN-00.
Date: MMM-YYYY	A string defining a date, i.e. JAN-2000.
Date: MMMDDYYYY	A string defining a date, i.e. JAN222000.
Date: MMMYY	A string defining a date, i.e. JAN00.
Date: MMMYYYY	A string defining a date, i.e. JAN2000.
Date: Wed Nov 15 10:55:37 CST 2000	The default date format of the Java platform, i.e. 'WED NOV 15 10:55:37 CST 2000'
Date: YY-MM-DD	A string defining a date, i.e. 00-01-22. (The string: 00-01-22 defines the date January 22, 2000.)
Date: YY/MM/DD	A string defining a date, i.e. 00/01/22. (The string: 00/01/22 defines the date January 22, 2000.)
Date: YYMMDD	A string defining a date, i.e. 000122. (The string: 000122 defines the date January 22, 2000.)
Date: YYYY-MM-DD	A string defining a date, i.e. 2000-01-22. (The string: 2000-01-22 defines the date January 22, 2000.)
Date: YYYY/MM/DD	A string defining a date, i.e. 2000/01/22. (The string: 2000/01/22 defines the date January 22, 2000.)
Date: YYYYMMDD	An eight byte numeric string of the format YYYYMMDD. A base data of String or EBCDIC may be specified to indicate the character encoding.
DateTime: DD/MM/YY hh:mm	A string defining a date and time, i.e. 22/01/00 12:24.
DateTime: DD/MM/YY hh:mm AM	A string defining a date and time, i.e. 22/01/00 12:24 AM.
DateTime: DD/MM/YY hh:mm:ss	A string defining a date and time, i.e. 22/01/00 12:24:00.
DateTime: DD/MM/YY hh:mm:ss AM	A string defining a date and time, i.e. 22/01/00 12:24:00 AM.
DateTime: MM/DD/YY hh:mm	A string defining a date and time, i.e. 01/22/00 12:24.
DateTime: MM/DD/YY hh:mi AM	A string defining a date and time, i.e. 01/22/00 12:24 AM.
DateTime: MM/DD/YY hh:mm:ss	A string defining a date and time, i.e. 01/22/00 12:24:00.
DateTime: MM/DD/YY hh:mm:ss AM	A string defining a date and time, i.e. 01/22/00 12:24:00 AM.
DateTime: MMDDYYhhmm	A string of numeric digits defining a date and time, i.e. 0122001224.
DateTime: YYYYMMDDhhmmss	A fourteen byte numeric string of the format YYYYMMDDHHMISS. A Base data type may be specified.
DateTime: MMDDYYhhmmss	A string of numeric digits defining a date and time, i.e. 012200122400.
EBCDIC	A string of characters in IBM Extended Binary Coded Decimal Interchange Code. Requires a length, length field, delimiter, or a delimiter field.

Table 22–1 (Cont.) Supported MFL Data Types

Data Type	Description
Filler	A sequence of bytes that is not transformed to XML. This field of data is skipped over when transforming non-XML data to XML. When transforming XML to non-XML data, this field is written to the binary output stream as a sequence of spaces.
FloatingPoint: 4 bytes, Big-Endian	A four byte big endian floating point number that conforms to the IEEE Standard 754.
FloatingPoint, 4 bytes, Little-Endian	A four byte little endian floating point number that conforms to the IEEE Standard 754.
FloatingPoint: 8 bytes, Big-Endian	A eight byte big endian floating point number that conforms to the IEEE Standard 754.
FloatingPoint: 8 bytes, Little-Endian	A eight byte little endian floating point number that conforms to the IEEE Standard 754.
Integer: Signed, 1 byte	A one byte signed integer, i.e. '56' is 0x38.
Integer: Unsigned, 1 byte	A one byte unsigned integer, i.e. '128' is 0x80.
Integer: Signed, 2 byte, Big-Endian	A signed two-byte integer in big endian format, i.e. '4660' is 0x1234.
Integer: Signed, 4 byte, Big-Endian	A signed four-byte integer in big endian format, i.e. '4660' is 0x00001234.
Integer: Signed, 8 bytes, Big-Endian	A signed eight-byte integer in big endian format, i.e. '4660' is 0x0000000000001234.
Integer: Unsigned, 2 byte, Big-Endian	An unsigned two-byte integer in big endian format, i.e. '65000' is 0xFDE8.
Integer: Unsigned, 4 byte, Big-Endian	An unsigned four-byte integer in big endian format, i.e. '65000' is 0x0000FDE8.
Integer: Unsigned, 8 bytes, Big-Endian	An unsigned eight-byte integer in big endian format, i.e. '65000' is 0x000000000000FDE8.
Integer: Signed, 2 bytes, Little-Endian	A signed two-byte integer in little endian format, i.e. '4660' is 0x3412.
Integer: Signed, 4 bytes, Little-Endian	A signed four-byte integer in little endian format, i.e. '4660' is 0x34120000.
Integer: Signed, 8 bytes, Little-Endian	A signed eight-byte integer in little endian format, i.e. '4660' is 0x3412000000000000.
Integer: Unsigned, 2 bytes, Little-Endian	An unsigned two-byte integer in little endian format, i.e. '65000' is 0xE8FD.
Integer: Unsigned, 4 bytes, Little-Endian	An unsigned four-byte integer in little endian format, i.e. '65000' is 0xE8FD0000.
Integer: Unsigned, 8 bytes, Little-Endian	An unsigned eight-byte integer in little endian format, i.e. '65000' is 0xE8FD000000000000.
Literal	A literal value determined by the contents of the value attribute. When non-XML data is transformed to XML, the presence of the specified literal in the non-XML data is verified by WLXT. The literal is read, but is not transformed to the XML data. When XML data is transformed to a non-XML format, and a literal is defined as part of the non-XML format, WLXT writes the literal in the resulting Non-XML byte stream.
Numeric	A string of characters containing only digits, i.e. '0' through '9'. Requires a length, length field, delimiter, or a delimiter field.

Table 22–1 (Cont.) Supported MFL Data Types

Data Type	Description
Packed Decimal: Signed	IBM signed packed format. Requires a length, length field, delimiter, or a delimiter field to be specified. The length or length field should specify the size of this field in bytes.
Packed Decimal: Unsigned	IBM unsigned packed format. Requires a length, length field, delimiter, or a delimiter field to be specified. The length or length field should specify the size of this field in bytes.
String	A string of characters. Requires a length, a length field, a delimiter, or a delimiter field. If no length, length field, or delimiter is defined for a data type String, a delimiter of "\x00" (a NUL character) will be assumed.
String: NUL terminated	A string of characters, optionally NUL (\x00) terminated, residing within a fixed length field. This field type requires a length attribute or length field which determines the amount of data read for the field. This data is then examined for a NUL delimiter. If a delimiter is found, data following the delimiter is discarded. If a NUL delimiter does not exist, the fixed length data is used as the value of the field.
Time: hhmmss	A string defining a time, i.e. 122400.
Time: hh:mm AM	A string defining a time, i.e. 12:24 AM.
Time: hh:mm	A string defining a time, i.e. 12:24.
Time: hh:mm:ss AM	A string defining a time, i.e. 12:24:00 AM.
Time: hh:mm:ss	A string defining a time, i.e. 12:24:00.
Zoned Decimal: Leading sign	Signed zoned decimal format (US-ASCII or EBCDIC) where the sign indicator is in the first nibble. Requires a length, length field, delimiter, or a delimiter field to be specified. The length or length field should specify the size of this field in bytes. Note: This data type is supported with US-ASCII data only with Message Format Language Version 2.02
Zoned Decimal: Leading separate sign	Signed zoned decimal format (US-ASCII or EBCDIC) where the sign indicator is in the first byte. The first byte only contains the sign indicator and is separated from the numeric value. Requires a length, length field, delimiter, or a delimiter field to be specified. The length or length field should specify the size of this field in bytes. Note: This data type is supported with US-ASCII data only with Message Format Language Version 2.02.
Zoned Decimal: Signed	Signed zoned decimal format (US-ASCII or EBCDIC). Requires a length, length field, delimiter, or a delimiter field to be specified. The length or length field should specify the size of this field in bytes. Note: This data type is supported with US-ASCII data only with Message Format Language Version 2.02.
Zoned Decimal: Trailing separate sign	Signed zoned decimal format (US-ASCII or EBCDIC) where the sign indicator is in the last byte. The last byte only contains the sign indicator and is separated from the numeric value. Requires a length, length field, delimiter, or a delimiter field to be specified. The length or length field should specify the size of this field in bytes. Note: This data type is supported with US-ASCII data only with Message Format Language Version 2.02.
Zoned Decimal: Unsigned	Unsigned zoned decimal format (US-ASCII or EBCDIC). Requires a length, length field, delimiter, or a delimiter field to be specified. The length or length field should specify the size of this field in bytes. Note: This data type is supported with US-ASCII data only with Message Format Language Version 2.02.

22.2 COBOL Copybook Importer Data Types

Table 22–2 lists the COBOL data types and the support provided by the Importer. Support for these data types is limited. The following formats:

```
05 pic 9(5) comp-5
05 pic 9(5) comp-x
```

will be converted to an unsigned 4 byte integer type, while the following will generate errors:

```
05 pic X(5) comp-5
05 pic X(5) comp-x
```

In these samples, `pic9(5)` could be substituted for `pic x(5)`.

Table 22–2 COBOL Data Types

COBOL Type	Support
BLANK WHEN ZERO (zoned)	supported
COMP-1, COMP-2 (float)	supported
COMP-3, PACKED-DECIMAL	supported
COMP, COMP-4, BINARY (integer)	supported
COMP, COMP-4, BINARY (fixed)	supported
COMP-5, COMP-X	supported
DISPLAY (alphanumeric)	supported
DISPLAY numeric (zoned)	supported
edited alphanumeric	supported
edited float numeric	supported
edited numeric	supported
group record	supported
INDEX	supported
JUSTIFIED RIGHT	ignored
OCCURS (fixed array)	supported
OCCURS DEPENDING (variable-length)	supported
OCCURS INDEXED BY	ignored
OCCURS KEY IS	ignored
POINTER	supported
PROCEDURE-POINTER	supported
REDEFINES	supported
SIGN IS LEADING SEPARATE (zoned)	supported
SIGN IS TRAILING (zoned)	supported
SIGN IS TRAILING SEPARATE (zoned)	supported
SIGN IS LEADING (zoned)	supported
SYNCHRONIZED	ignored
66 RENAMES	not supported

Table 22–2 (Cont.) COBOL Data Types

COBOL Type	Support
66 RENAMES THRU	not supported
77 level	supported
88 level (condition)	ignored

The following values are defined as follows:

- Supported - the data type will be correctly parsed by the importer and converted to a message format field or group.
- Unsupported - this data type is not supported and the importer reports an error when the copybook is imported.
- Ignored - the data type is parsed and a comment is added to the message format. No corresponding field or group is created.

Some vendor-specific extensions are not recognized by the importer, however, any copybook statement that conforms to ANSI standard COBOL will be parsed correctly by the Importer. The Importer's default data model, which is based on the IBM mainframe model, can be changed in Format Builder to compensate for character set and data "endianness".

When importing copybooks, the importer may identify fields generically that, upon visual inspection, could easily be identified by a more specific data type. For this reason, the copybook importer creates comments for each field found in the copybook. This information is useful in assisting you in editing the MFL data to better represent the original Copybook. For example:

original copybook entry:

```
05 birth-date    picxx/xx/xx
```

results in:

A field of type EBCDIC with a length of 8

Closer inspection indicates that this is intended to be a date format and could be defined as

A field of type Date: MM/DD/YY or a field of type Data: DD/MM/YY

22.3 Unsupported C Language Features

The C struct Importer utility does not parse files containing anonymous unions, bit fields, or in-line assembler code. The following samples of unsupported features are taken from the preprocessor output of a `hello.c` file that contained a `#include <windows.h>` statement:

- Anonymous unions


```
#line 353 "e:\\program files\\microsoft visual studio\\vc98\\include\\winnt.h"
typedef union_LARGE_INTEGER{
    struct {
        DWORD LowPart;
        LONG HighPart;
    };
    struct {
        DWORD LowPart;
        LONG HighPart;
```



```

    } u;
#line 363 "e:\\program files\\microsoft visual studio\\vc98\\include\\winnt.h"
    LONGLONG QuadPart;
} LARGE_INTEGER

```

- Bit fields

```

typedef struct_LDT_ENTRY {
    WORD LimitLow;
    WORD BaseLow;
    union {
        struct {
            BYTE BaseMid;
            BYTE Flags1;
            BYTE Flags2;
            BYTE BaseHi;
        } Bytes;
        struct
            DWORD BaseMid : 8;
            DWORD Type : 5;
            DWORD Dpl : 2;
            DWORD Pres : 1;
            DWORD LimitHi : 4;
            DWORD Sys : 1;
            DWORD Reserved_0 : 1;
            DWORD Default_Big : 1;
            DWORD Granularity : 1;
            DWORD BaseHi : 8;
        } Bits;
    } HighWord;
} LDT_ENTRY, *PLDT_ENTRY;

```

- Inline assembler code

```

_inline ULONGLONG
_stdcall
Int64Shr1Mod32(
    ULONGLONG Value,
    DWORD ShiftCount
)
{
    _asm {
        mov ecx, ShiftCount
        mov eax, dword ptr [Value]
        mov edx, dword ptr [Value+4]
        shrd eax, edx, cl
        shr edx, cl
    }
}

```


Part IV

General Development Topics

This part, which contains general development topics, includes the following chapters:

- [Chapter 23, "Creating and Using Custom XPath Functions"](#)

Creating and Using Custom XPath Functions

Oracle Service Bus provides an extensible framework for creating custom XPath functions you can use in the XQuery expression editors in the development or run-time tooling, such as in proxy service message flows, Split-Joins, and XQuery Mapper transformations.

This section describes how to create, register, and use custom XPath functions in XQuery expressions. The high-level process includes:

- [Section 23.1, "Registering Custom Functions with Oracle Service Bus"](#)
- [Section 23.2, "Creating and Packaging the Custom Function Java Classes"](#)
- [Section 23.3, "Using Custom Functions"](#)
- [Section 23.5, "Deploying Custom Functions in a Cluster"](#)

Note: Oracle Service Bus does not support custom functions that have side effects; for example, updating a database table or participating in a global transaction. Create custom functions that contribute only to an XQuery result, and perform side-effect behavior with other features such as Java Callouts.

23.1 Registering Custom Functions with Oracle Service Bus

Custom functions are available to all Oracle Service Bus projects and services in an installation; they are not scoped to specific domains.

Registering a custom function involves creating an XML file with an optional properties file for localization. The built-in functions that Oracle Service Bus provides use this function framework, so you can use those existing registration resources for guidance. Those files are located at:

OSB_ORACLE_HOME/config/xpath-functions/

The Oracle Service Bus functions file is called *osb-built-in.xml*. In that file, keys wrapped in % symbols, such as *%OSB_FUNCTIONS%*, get their value from the corresponding *.properties* file.

Following is the basic structure of a custom function registration file, followed by descriptions of the elements.

```
category id
  group id
    function
      name
      comment
```

```

namespaceURI
className
method
isDeterministic

```

Elements have an xpf: prefix.

Custom Function Element Descriptions

- `category id` – The name of the group that physically categorizes your group of functions in the expression editors. For example, "Service Bus Functions." Use the `id` attribute to provide the name. If you are using a corresponding `.properties` file for localization, enter the key that contains the text value in the `.properties` file. For example, `%MY_FUNCTIONS%`. Category ids, which include the properties file key name and the actual name value, must be unique.
- `group id` – Optional. The name of a subcategory for grouping functions in the user interface; for example, "General" or "Accessors." The naming guidelines for category id apply to group id.
- `name` – Name of the function as it appears in XQuery expressions. Function names (composed of the namespaceURI and prefix as well) must be unique. Oracle Service Bus does not support function overloading with different method arguments. Identical function names that have different namespaces (hence different prefixes) are allowed.
- `comment` - Function description. While the description does not appear in the Oracle Service Bus user interface, you should provide guidance that shows how to invoke the function with meaningful argument names.
- `namespaceURI` – The namespace of the function. For example, the Oracle Service Bus functions namespace is `http://www.bea.com/xquery/xquery-functions`. Namespaces and namespace prefixes must be unique. Custom namespaces that you provide appear in the Oracle Service Bus default namespaces list in the XQuery editor.
- `className` – The fully qualified custom Java class that implements the function.
- `method` – The custom Java method that implements the function, preceded by the return type. For example: `boolean isUserInGroup(java.lang.String, java.lang.String)`.

If your method uses a single-dimensional array, see [Section 23.2.1.1, "Using Single-Dimensional Arrays"](#) for guidance in making the entry in the XML file.

- `isDeterministic` – A value of `true` or `false` declaring whether or not the function is deterministic. Deterministic functions always provide the same results; for example, a function that concatenates Strings. Non-deterministic functions return unique results; for example, a function that returns the time of day. Though you can use non-deterministic functions, the XQuery standard recommends that functions be deterministic to ensure XQuery engine optimization.

Your custom functions do not appear in the XQuery expression editor until Oracle Service Bus can find your custom class. The following section describes how to create your custom class and package it so that Oracle Service Bus can locate it.

23.2 Creating and Packaging the Custom Function Java Classes

Use the following guidelines to create and package the class for a custom XPath function.

23.2.1 Creating the Class and Method

Use the following guidelines for creating Java class and method for a custom function.

- class – The class must be public.
- method – The method must be public and static.
- arguments and return values – [Table 23–1](#) lists the supported types for method arguments and return values. If a type is not listed, it is not supported. Inner classes and multi-dimensional arrays are not supported.

Table 23–1 Supported Java Method Types for Custom Functions

Java Type	XQuery Type	XSLT Type
java.lang.String	xs:string	string
int, java.lang.Integer	xs:int	number
boolean, java.lang.Boolean	xs:boolean	boolean
long, java.lang.Long	xs:long	number
short, java.lang.Short	xs:short	number
byte, java.lang.Byte	xs:byte	number
double, java.lang.Double	xs:double	number
float, java.lang.Float	xs:float	number
char, java.lang.Char	xs:string	object
java.math.BigInteger	xs:integer	number
java.math.BigDecimal	xs:decimal	number
java.util.Date	xs:datetime	See footnote ¹
java.sql.Date	xs:date	See footnote
java.sql.Time	xs:time	See footnote
javax.xml.namespace.QName	xs:Qname	See footnote
org.apache.xmlbeans.XmlObject	element()	See footnote
org.w3c.dom.Element	element()	The XSLT node-set type is not supported with custom XPath functions.

¹ Converted to a string, then passed back as its original type.

23.2.1.1 Using Single-Dimensional Arrays

Single-dimension arrays (using supported Java types) are mapped to corresponding XQuery types with an asterisk *, which is a wild card to imply the multiple cardinality of the array. For example:

```
public static XmlObject[] getArrayOfXmlObjects(XmlObject[] a)
```

is mapped to

```
namespace:getArrayOfXmlObjects($arg1 as element(*) as element(*)
```

In function signatures that have single-dimensional array input arguments or return values, you must use the type encoding described at <http://java.sun.com/javase/6/docs/api/java/lang/Class.html#getNa>

`me()`. The following examples show how to specify single-dimensional array methods in your custom function XML file using the required array encoding:

Java Method	Entry in Custom Function XML File
<pre>public static String[] myUppercaseStringArray(String[] arg)</pre>	<pre>Ljava.lang.String; myUppercaseStringArray([Ljava.lang.String;)</pre>
<pre>public static int[] myAddInts(int[] arg)</pre>	<pre>[I myAddInts([I)</pre>

23.2.2 Packaging the Custom Function Class

Oracle Service Bus must know about your custom function class in order to include your custom functions in the XQuery editors and let you use those functions.

Package your custom function class in a JAR file, then put the JAR in the `OSB_ORACLE_HOME/config/xpath-functions/` directory. At IDE and server start-up, Oracle Service Bus looks for custom function classes in this directory. Be sure to correctly reference your custom class and method in the custom function XML file, described in [Section 23.1, "Registering Custom Functions with Oracle Service Bus."](#)

Note: You can also provide run-time-only access to your custom function class by putting your custom JAR anywhere in the Oracle Fusion Middleware home and adding the JAR to the server classpath.

After you add new custom functions, you must restart the IDE and any servers that will use the new functions.

23.3 Using Custom Functions

This section describes how to use custom functions in Oracle Service Bus.

23.3.1 Using Custom Functions in Inline XQuery Expressions and XQuery Resources

You can include custom functions in both inline XQuery expressions and in XQuery resources just as you would use functions provided by Oracle Service Bus.

23.3.2 Using Custom Functions in XSLT Resources

The syntax for invoking a custom function from within an XSLT resource varies by the XSLT engine you use with Oracle Service Bus. Given the following custom function code, [Example 23–1](#) shows the syntax for invoking a custom function using the Xalan XSLT engine (the default on Microsoft Windows with the Sun JDK).

```
package tests.pipeline;
public class CustomXQFunctions
{
    public static String myUppercaseString(String arg)
    {
        return arg.toUpperCase();
    }
}
```

Example 23–1 Syntax for Invoking a Custom Function with the Xalan Engine

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
```



```

<xsl:param name="arg-string" />
<xsl:template name="myUppercaseString"
    xmlns:ns0="xalan://tests.pipeline.CustomXQFunctions">
  <xsl:variable name="upcase" select="ns0:myUppercaseString($arg-string)" />
  <originalInput>
    <xsl:value-of select="$arg-string" />
  </originalInput>
  <result>
    <xsl:value-of select="$upcase" />
  </result>
</xsl:template>
<xsl:template match="*">
  <xsl:copy>
    <xsl:call-template name="myUppercaseString" />
  </xsl:copy>
</xsl:template>
</xsl:stylesheet>

```

With an input document of `<example />` and an input `arg-string` value of `hello`, the transformation becomes:

```

<example>
  <originalInput>hello</originalInput>
  <result>HELLO</result>
</example>

```

23.4 Testing Custom XPath Functions in Eclipse

In Eclipse, when testing XQuery resources that contain custom XPath functions, the best practice is to run the XQuery resources on a connected server using the **Run > Run As > Run on Server** command, which runs the resource in the Test Console.

23.5 Deploying Custom Functions in a Cluster

In a multiple-server environment with multiple Oracle Fusion Middleware product homes, you must manually add all custom function resources to any of those environments where the custom functions will be used. Clustering does not automatically distribute custom function resources across managed servers.

Part V

Transports

This part, which provides information and configuration details on all transports provided by Oracle Service Bus, contains the following chapters:

- [Chapter 24, "Oracle SOA Suite Transport \(SOA-DIRECT\)"](#)
- [Chapter 25, "JCA Transport"](#)
- [Chapter 26, "HTTP and Poller Transports"](#)
- [Chapter 27, "SB Transport"](#)
- [Chapter 28, "EJB Transport"](#)
- [Chapter 29, "JEJB Transport"](#)
- [Chapter 30, "JMS Transport"](#)
- [Chapter 31, "Local Transport"](#)
- [Chapter 32, "WS Transport"](#)
- [Chapter 33, "MQ Transport"](#)
- [Chapter 34, "Oracle BPEL Process Manager Transport"](#)
- [Chapter 35, "Tuxedo Transport"](#)
- [Chapter 36, "DSP and Oracle Data Service Integrator Transport"](#)

Oracle SOA Suite Transport (SOA-DIRECT)

Oracle Service Bus provides a SOA-DIRECT transport that lets you invoke Oracle SOA Suite service components, such as BPEL processes, human tasks, rules, and Oracle Mediator components.

This document, which describes the SOA-DIRECT transport, contains the following sections:

- [Section 24.1, "About the SOA-DIRECT Transport"](#)
- [Section 24.2, "Using SOA Suite Services with Oracle Service Bus"](#)
- [Section 24.3, "Transport Configuration Reference"](#)
- [Section 24.4, "WS-Addressing Reference"](#)
- [Section 24.5, "XML Examples"](#)

24.1 About the SOA-DIRECT Transport

The SOA-DIRECT transport provides native connectivity between Oracle Service Bus and Oracle SOA Suite service components. Oracle SOA Suite provides a "direct binding" framework that lets you expose Oracle SOA Suite service components in a composite application, and the Oracle Service Bus SOA-DIRECT transport interacts with those exposed services through the SOA direct binding framework, letting those service components interact in the service bus layer and leverage the capabilities and features of Oracle Service Bus.

For more information on SOA binding components, see "Getting Started with Binding Components" and "Using the Direct Binding Invocation API" in the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

Note: The SOA-Direct transport supports WSDL type services with SOAP 1.1, SOAP 1.2, or, alternatively, XML bindings. The SOA direct binding framework only exposes direct binding services as WSDL with SOAP 1.1 and SOAP 1.2 bindings, not XML. However, if you want to use an XML binding, you must manually customize the imported SOA service WSDLs for the direct binding services. An XML binding has no effect on the message payload, since messages between the SOA-DIRECT transport and SOA binding components are always abstract (no binding).

The SOA-DIRECT transport supports the following features:

- Invocation of any SOA binding component services through Java remote method invocation (RMI)
- WS-Addressing, including optional auto-generation of ReplyTo properties for asynchronous callbacks
- Identity propagation
- Transaction propagation
- Attachments
- Optimized RMI transport for invoking SOA services
- High availability and clustering support
- Failover and load balancing (not available for services in the Service Callback role)
- Connection and application retries on errors

24.1.1 WS-Addressing

The SOA-DIRECT transport uses only WS-Addressing for message correlation in synchronous and asynchronous communications. The transport automatically generates the following WS-Addressing properties in the SOAP header when you configure a callback proxy in the business service configuration:

- ReplyTo – For setting the callback address and connection details in asynchronous callbacks.
- ReferenceParameters – Contains the callback properties for ReplyTo, including JNDI and connection factory properties, for the following supported WS-Addressing versions:
 - <http://www.w3.org/2005/08/addressing>
 - <http://schemas.xmlsoap.org/ws/2004/08/addressing>
- ReferenceProperties – Contains the callback properties for ReplyTo, including JNDI and connection factory properties, for the following supported WS-Addressing version: <http://schemas.xmlsoap.org/ws/2003/03/addressing>.

For ReplyTo and ReferenceParameters examples, see [Section 24.4, "WS-Addressing Reference."](#)

For all other WS-Addressing properties, you must add or transform them in Oracle Service Bus proxy service message flows if they are not available or suitable for pass-through to the SOA-DIRECT business service.

If you use correlation and callback mechanisms other than WS-Addressing, you must transform messages in proxy service pipelines to support WS-Addressing between Oracle Service Bus and SOA framework service components.

For WS-Addressing examples with the SOA-DIRECT transport, see [Section 24.4, "WS-Addressing Reference"](#) and [Section 24.5, "XML Examples."](#)

24.1.2 Security

The SOA-DIRECT transport supports one-way SSL. To use SSL, enable SSL in the domain, use the secure protocol in the endpoint URI, such as https, iiops, or t3s, and reference the secure port in the URI. For more information on the SOA-DIRECT URI, see [Section 24.3.1, "SOA-DIRECT Endpoint URI."](#)

You can provide identity propagation with the SOA-DIRECT transport by passing the caller's subject through the service or with a service account bound to the service. Because the SOA-DIRECT transport deals with only normalized, abstract messages, the transport does not support WS-Security. For more information on security settings, see [Section 24.3.2, "SOA-DIRECT Transport Configuration for Business Services."](#)

24.1.3 Environment Values

The SOA-DIRECT transport stores the following environment values for SOA-DIRECT services:

- JNDI Service Account (security category)
- Pass Caller's Subject (security category)
- Invocation Service account (security category)
- Work Manager (environment category)

For information on these values, see [Section 24.3.2, "SOA-DIRECT Transport Configuration for Business Services."](#)

24.1.4 Error Handling

The SOA-DIRECT transport recognizes connection and application errors, letting you configure the appropriate retry settings in the transport configuration. The transport throws generic errors for errors that are neither connection nor application related.

24.1.4.1 Connection Errors

The SOA-DIRECT transport raises connection errors in the following situations:

- The target service does not exist
- A naming exception occurs during the RMI lookup or invocation (with the exception of `javax.naming.NamingSecurityException`, which is a generic error).
- A remote exception occurs during the RMI lookup or invocation.

24.1.4.2 Application Errors

The SOA-DIRECT transport raises application errors when the outbound business service receives a SOAP fault.

You can deselect Retry Application Errors on the service configuration page to prevent retries on application errors—errors that are likely to keep failing despite retries.

24.1.4.3 Generic Errors

The SOA-DIRECT transport raises a generic error in the following situations:

- All errors other than connection and application errors.
- A `javax.naming.NamingSecurityException`, which is thrown during the JNDI lookup, is not considered a connection error as are other naming exceptions.

24.2 Using SOA Suite Services with Oracle Service Bus

This section describes synchronous and asynchronous communication patterns between Oracle Service Bus and Oracle SOA Suite composites.

24.2.1 Simple Use Cases – Synchronous

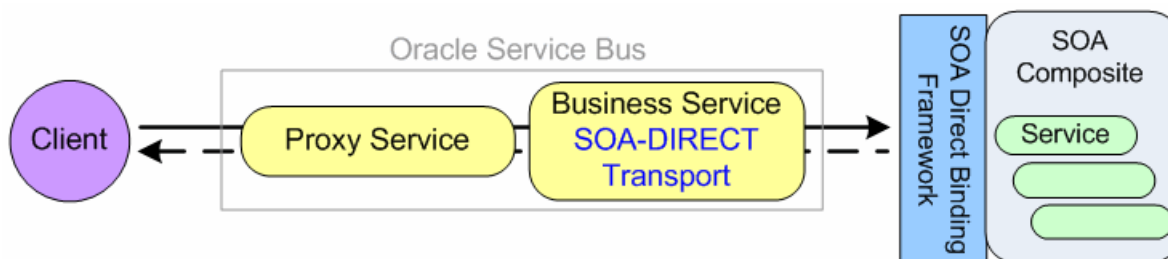
This section describes the simple, most common use cases for communicating natively to and from SOA composites through Oracle Service Bus: synchronous communication.

24.2.1.1 Synchronous Invocation of a SOA Composite

The Oracle Service Bus SOA-DIRECT transport can invoke any component in a SOA composite that is exposed as a direct binding service.

Figure 24–1 illustrates a synchronous communication pattern between a client and an Oracle SOA service component through Oracle Service Bus using a SOA-DIRECT business service and direct binding service.

Figure 24–1 Client Invoking a SOA Binding Service Synchronously



24.2.1.1.1 Creating and Configuring the Services Use the following guidelines to invoke an Oracle SOA direct binding service from a client through Oracle Service Bus:

- Create a SOA-DIRECT business service in Oracle Service Bus that represents the SOA service component you want to invoke.
 - In Oracle Service Bus, create a WSDL resource based on the corresponding Oracle SOA direct binding service WSDL.

You can locate the SOA direct binding service WSDL in Oracle JDeveloper using the SOA Resource Browser, as described in "Developing SOA Composite Applications with Oracle SOA Suite" in the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.

 - Create a new business service.
 - Select **WSDL Web Service** as the Service Type.
 - Select the WSDL resource you created, and choose the appropriate port or binding.

Note: If you select the port, the transport type and URI will be automatically propagated in the next configuration page.

- Select the **soa-direct** transport in the business service configuration.
- Set the endpoint URI as described in [Section 24.3.1, "SOA-DIRECT Endpoint URI."](#)
- Configure the remainder of the business service, described in [Section 24.3.2, "SOA-DIRECT Transport Configuration for Business Services."](#)

- Create a proxy service in Oracle Service Bus that invokes the business service. Choose the transport type that is used by the client. For proxy configuration details, see [Section 4.3, "Proxy Service Configuration"](#) and [Chapter 3, "Transport Configuration."](#)

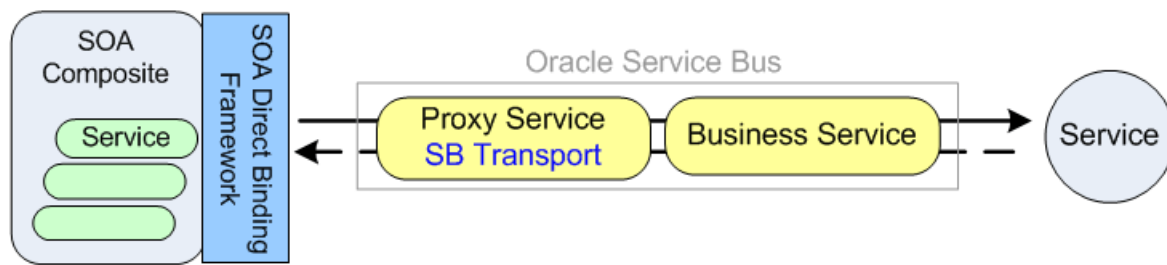
If you are using stateful services to ensure that messages are associated with the correct conversation, see [Section 24.2.1.3, "Associating Messages with the Correct Conversation."](#)

24.2.1.2 Synchronous Invocation from a SOA Composite

A SOA composite can invoke any Oracle Service Bus SB WSDL-based proxy service. To invoke an SB proxy service, the SOA service component must use a direct binding reference of Target Type "Oracle Service Bus." (For more information on target types, see the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite.*)

[Table 24–2](#) illustrates a synchronous communication pattern between an Oracle SOA service component and an external service through Oracle Service Bus.

Figure 24–2 SOA Binding Service Invoking an External Service Synchronously



24.2.1.2.1 Creating and Configuring the Services Use the following guidelines to invoke an external service from a SOA composite using direct binding references:

- Create a business service in Oracle Service Bus that represents the external service you want to invoke. Choose the transport type that is supported by this service. For business service configuration details, see [Section 4.2, "Business Service Configuration."](#) and [Chapter 3, "Transport Configuration."](#)
- Create an SB proxy service in Oracle Service Bus that invokes the business service.
 - Create a WSDL resource to be used by the proxy that invokes the business service.
 - Create a new proxy service.
 - Select **WSDL Web Service** as the Service Type.
 - Select the WSDL for the proxy service, and select the desired port or binding.
 - Select the **sb** transport in the proxy service configuration.
 - Configure the remainder of the proxy service. For more information, see [Section 4.3, "Proxy Service Configuration"](#) and [Chapter 27, "SB Transport."](#)

Note: Use the SB proxy service effective WSDL and port type to define the direct binding reference that invokes Oracle Service Bus. You can import this WSDL into an Oracle SOA Suite project.

If you are using stateful services, ensure that messages are associated with the correct conversation. See [Section 24.2.1.3, "Associating Messages with the Correct Conversation."](#)

24.2.1.3 Associating Messages with the Correct Conversation

When using stateful services, the messages sent synchronously between Oracle Service Bus and Oracle SOA composites are known as a conversation. To ensure that messages are correctly associated with each other as part of a conversation, the Oracle Service Bus SOA-DIRECT transport provides built-in support for WS-Addressing.

For more information on WS-Addressing, see [Section 24.4.2, "MessageID / RelatesTo Headers."](#) For an example of conversation ID setting, see [Section 24.5.1, "Conversation ID Examples."](#)

24.2.2 Advanced Use Cases – Asynchronous

This section describes asynchronous communications between a SOA composite and Oracle Service Bus using the SOA-DIRECT transport.

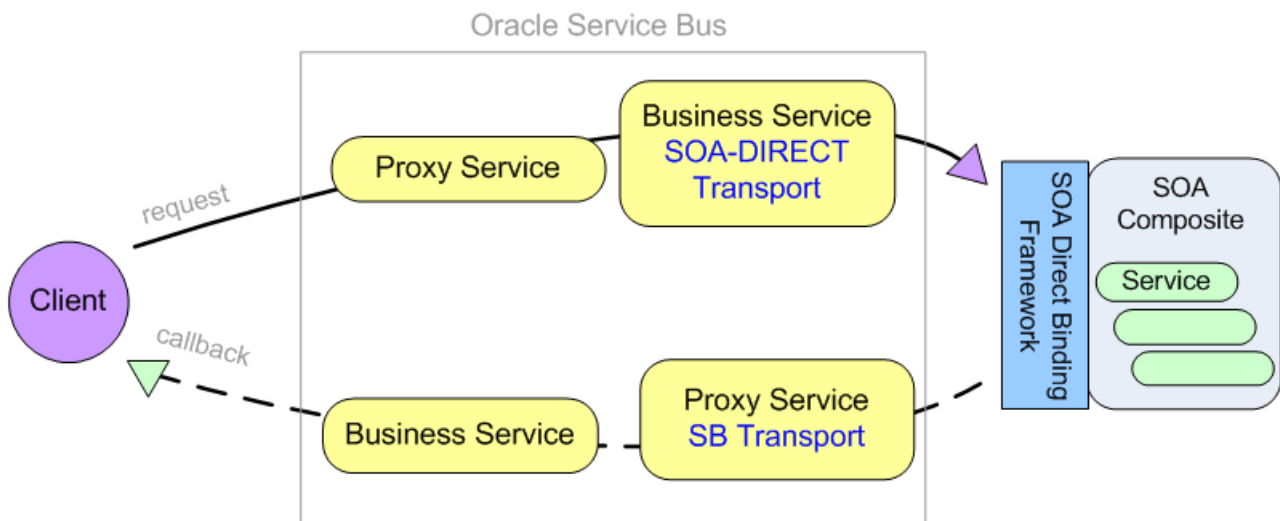
Note: Only the following SOA service components currently support asynchronous conversations using WS-Addressing: BPEL Process, Mediator, and Human Task.

24.2.2.1 Asynchronous Invocation of a SOA Composite

The Oracle Service Bus SOA-DIRECT transport can invoke asynchronous SOA service components that are exposed as direct binding services.

[Figure 24–3](#) illustrates an asynchronous communication pattern between a client and an Oracle SOA composite through Oracle Service Bus using a direct binding service, the SOA-DIRECT transport, and the SB transport.

Figure 24–3 Client Invoking a SOA Binding Service Asynchronously



24.2.2.1.1 Creating and Configuring the Services Use the following guidelines to invoke the SOA direct binding service asynchronously from a client through Oracle Service Bus:

- On the inbound client side, create the Oracle Service Bus artifacts to interact with the client: a request proxy service that invokes the outbound SOA-DIRECT business service, and a callback business service that handles the callback to the client. Use the transport type used by the client.
 - **Request Proxy Service**

Configure the proxy service that receives the client request. This proxy service invokes the outbound request SOA-DIRECT business service.

Since the callback is sent to a different connection, Oracle Service Bus must be able to remember the original callback location when calling back the client. When using WS-Addressing, the callback address is sent to the request proxy service in the ReplyTo address header. Before invoking the SOA-DIRECT business service, the request proxy can pass this address as a referenceParameter property inside the ReplyTo header. Following the WS-Addressing specification, the referenceParameter property is inserted in the SOAP header block of the callback. The callback SB proxy can then extract this callback address and set the callback business service URI.

For information on setting a callback address, see [Section 24.4.1, "ReplyTo Header"](#) and [Section 24.5.2, "Asynchronous Composite to Composite Native Communication Through Oracle Service Bus Example."](#)
 - **Callback Business Service**

Configure the business service you need to handle the callback. This business service is invoked by the outbound callback SB proxy service.

For service and transport configuration guidance, see [Section 4.2, "Business Service Configuration"](#) and [Chapter 3, "Transport Configuration"](#).
- On the Oracle Service Bus outbound side, create the artifacts to interact with the SOA composite: a request SOA-DIRECT business service that makes the request to the Oracle SOA direct binding service exposing the asynchronous service component you want to invoke, and a callback SB proxy service that handles the callback from the direct binding service and invokes the inbound callback business service.
 - **Request SOA-DIRECT Business Service**
 - * In Oracle Service Bus, create a WSDL resource based on the corresponding Oracle SOA direct binding service WSDL.

You can locate the SOA direct binding service WSDL in Oracle JDeveloper using the SOA Resource Browser, as described in "Developing SOA Composite Applications with Oracle SOA Suite" in the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.
 - * Create a new business service.
 - * Select **WSDL Web Service** as the Service Type. Browse to the WSDL resource you created and select the appropriate port or binding for the direct binding service.

If you select the port, the transport type and URI are automatically propagated in the next configuration page.
 - * Select the **soa-direct** transport in the business service configuration.
 - * Set the URI, described in [Section 24.3.1, "SOA-DIRECT Endpoint URI."](#)
 - * On the transport configuration page, set the Role to **Asynchronous Client**.

- * Optionally use the **Callback Proxy** option to select the SB callback proxy service you created.

When you select a callback proxy, the SOA-DIRECT transport automatically generates the WS-Addressing headers to tell the SOA direct binding service that it expects the asynchronous callback response to be sent to the specified callback proxy.

For approaches to setting a callback address if you do not select a callback proxy in the SOA-DIRECT business service, see [Section 24.4, "WS-Addressing Reference"](#) and [Section 24.5.2, "Asynchronous Composite to Composite Native Communication Through Oracle Service Bus Example."](#)

- * Configure the remainder of the business service. For more information, see [Section 24.3, "Transport Configuration Reference"](#) and [Section 4.2, "Business Service Configuration."](#)
- * Invoke this business service from the request proxy service.

- **Callback SB Proxy Service**

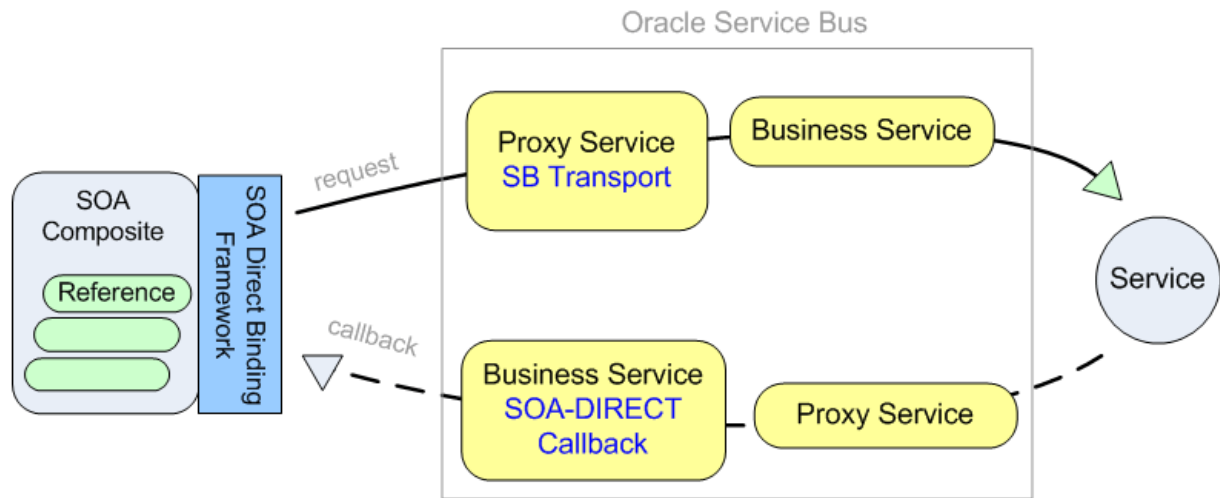
- Create a new proxy service.
- Select **WSDL Web Service** as the Service Type.
- Browse to the WSDL corresponding to direct binding service's WSDL, and select the appropriate port or binding.
- Select the **sb** transport in the proxy service configuration.
- Complete the proxy service configuration. For more information, see [Section 4.3, "Proxy Service Configuration"](#) and [Chapter 27, "SB Transport."](#)

24.2.2.2 Asynchronous Invocation from a SOA Composite

An asynchronous SOA service component in a SOA composite can invoke external services through Oracle Service Bus. To do so, the SOA service component must use a direct binding reference of Target Type of "Oracle Service Bus." (For more information on target types, see the *Oracle Fusion Middleware Developer's Guide for Oracle SOA Suite*.)

[Figure 24-4](#) illustrates an asynchronous communication pattern between an Oracle SOA service component and an external service through Oracle Service Bus using a direct binding reference, the SB transport, and the SOA-DIRECT transport.

Figure 24–4 SOA Binding Service Invoking an External Service Asynchronously



24.2.2.2.1 Creating and Configuring the Services Use the following guidelines to invoke an external service asynchronously from an Oracle SOA direct binding reference through Oracle Service Bus.

- In Oracle Service Bus, on the inbound side, create the artifacts to interact with the SOA composite: a request SB proxy service that receives the SOA direct binding reference request and a callback SOA-DIRECT business service that handles the callback to the SOA direct binding reference.

– **Request SB Proxy Service**

- * Create a WSDL resource representing the interface used to interact with the direct binding reference.
- * Create a new proxy service.
- * Select **WSDL Web Service** as the Service Type.
- * Browse to the WSDL you created and select the appropriate port or binding.
- * Select the **sb** transport in the proxy service configuration page.
- * Complete proxy service configuration. For more information, see [Section 4.3, "Proxy Service Configuration"](#) and [Chapter 27, "SB Transport."](#)

Since the callback is sent to a different connection, Oracle Service Bus must be able to "remember" the original callback location when calling back the client. When using WS-Addressing, the callback address is sent to the request proxy service in the ReplyTo address header. Before invoking the external service, the request proxy service passes this address as a referenceParameter property inside the ReplyTo header. Following the WS-Addressing specification, the referenceParameter property is inserted in the SOAP header block of the callback. The callback proxy service can then extract this callback address and set the callback business service URI.

For information on setting a callback address, see [Section 24.4.1, "ReplyTo Header"](#) and [Section 24.5.2, "Asynchronous Composite to Composite Native Communication Through Oracle Service Bus Example."](#)

– **Callback Business Service**

- * Create a new business service.
- * Select **WSDL Web Service** as the Service Type.
- * Browse to the WSDL representing the callback interface with the direct binding reference, and select the appropriate port or binding.
- * Select the **soa-direct** transport in the business service configuration.
- * Set the service URI to "callback," as described in [Section 24.3.1, "SOA-DIRECT Endpoint URI."](#)

In general, the callback URI is dynamically set in the invoking proxy using URI rewriting. However, if the callback address is always known, you can provide the exact callback address instead of "callback."

- * Set the role to **Service Callback** on the SOA-DIRECT transport configuration page.
 - * Configure the remainder of the business service, as described in [Section 24.3.2, "SOA-DIRECT Transport Configuration for Business Services"](#) and [Section 4.2, "Business Service Configuration."](#)
- On the Oracle Service Bus outbound side, create the artifacts to interact with the external service: a request business service that makes the request to the external service and a callback proxy service that handles the callback from this service.

- **Request Business Service**

Configure the business service to invoke the external service. This business service will be invoked by the request SB proxy service. Choose the transport type that is supported by this service. For business service configuration details, see [Section 4.2, "Business Service Configuration"](#) and [Chapter 3, "Transport Configuration."](#)

- **Callback Proxy Service**

Configure the proxy service to pass the callback address to the business service. The callback URI is provided in the request. Use URI rewriting to extract the callback URI and forward it to the SOA-DIRECT business service. Choose the transport type that is supported by this service. For proxy service configuration details, see [Section 4.3, "Proxy Service Configuration"](#) and [Chapter 3, "Transport Configuration."](#)

For information on setting the callback addresses using WS-Addressing, see [Section 24.4, "WS-Addressing Reference."](#)

24.3 Transport Configuration Reference

This section describes the endpoint URL format and configuration options for the SOA-DIRECT transport.

24.3.1 SOA-DIRECT Endpoint URI

Following is the URI pattern for the SOA-DIRECT transport. Optional elements are in brackets [].

callback – This is the URI for SOA-DIRECT business services in the Service Callback role handling the inbound request. The actual URI is specified dynamically at run time in the proxy service pipeline. However, if the callback address is always known, you can provide the exact callback address.

For all other SOA-DIRECT business service roles:

*[protocol://authority]/default/compositeName[!versionNumber[*label]]/serviceName*

- **protocol**

Use one of the following RMI / JNDI protocols:

- **iiop / iiops** – For generic, server-agnostic use.
- **t3 / t3s** – For use with Oracle WebLogic Server.
- **http / https** – For tunneling and use with Oracle WebLogic Server.

For HTTP(S) protocols, enable HTTP tunneling on the server. For SSL protocols, enable SSL on the server.

The protocol and authority are optional when the SOA services are co-located on the same server as Oracle Service Bus.

Following are descriptions of the other endpoint URI elements:

- **authority** – The IP address or host name and the port of the SOA server or cluster hosting the SOA service components.

The protocol and authority are optional when the SOA services are co-located on the same server as Oracle Service Bus.

- **default** – This domain name value is always "default."
- **compositeName** – The name of the composite application where the binding component service is defined.
- **!versionNumber** – Optional. The composite application version number. If you do not specify a version, the current version is used.
 - ***label** – Optional, used with !versionNumber; the label hash used in the SOA service WSDL.
- **serviceName** – The name of the SOA binding component service.

While you can specify more than one URI on a service for load balancing and failover, only one URI is allowed for services in the Service Callback role, described in [Table 24–1](#). Therefore, load balancing and failover are not available for services in the Service Callback role.

24.3.1.1 Cluster URI

Following is the format for the endpoint URI in a cluster:

t3://example_managed1:port1,example_managed2:port2/service_path

Where *t3://example_managed1:port1,example_managed2:port2* is the JNDI provider URL.

24.3.1.2 URI Examples

Following are URI examples for the SOA-DIRECT transport:

- *t3s://example:7002/default/compositeApp/1.0/myService*
Points to a service deployed on a single node.
- */default/compositeApp!1.0/myService*
Points to a service co-located on the same server as Oracle Service Bus.
- *t3://soaserver.example.com:7001/default/VacationRequest!1.0*ec2dd6c5-1667-4885-a634-2364547beb2d/directclient*

Points to a service deployed on a single node using a version and hash code. This is the default format in SOA binding component service WSDLs.

- t3://example_managed1:8001,example_managed2:8002/default/myComposite/myService

Points to a clustered SOA framework environment identified by "myService." Because no specific version is specified, the most recent version of the service is used.

24.3.2 SOA-DIRECT Transport Configuration for Business Services

Table 24–1 describes the transport-specific configuration options for the SOA-DIRECT transport.

Table 24–1 SOA-DIRECT Transport Configuration

Property	Description
JNDI Service Account	Optional. Specifies the security credentials for the JNDI lookup of the target SOA service. The service account must be static. Click Browse and select a service account. If you do not specify a service account, an anonymous subject is used.
Role	<p>Required. Identifies the communication pattern the service uses. Select one of the following options:</p> <ul style="list-style-type: none"> ■ Synchronous Client (default) – In this role, because there is no asynchronous callback, the Callback Proxy option is disabled. The WS-Addressing Version option is also disabled. ■ Asynchronous Client – In this role, because asynchronous callback is usually required, you can identify a Callback Proxy, and you must select a WS-Addressing Version. The asynchronous option is enabled only when the WSDL service is of type SOAP. ■ Service Callback – This role is for returning the asynchronous callback to a SOA service after an external service invocation. <p>Load balancing or failover are not available for services in the Service Callback role.</p>
Callback Proxy	<p>Optional. Enabled only for the Asynchronous Client role.</p> <p>This option lets you specify the proxy service that receives callbacks. When you select a callback proxy, if no WS-Addressing is provided by the request or the proxy service pipeline, Oracle Service Bus automatically populates the ReplyTo property in the SOAP header. You must select a WSDL proxy service that uses the SB transport (for RMI), and the callback proxy service must understand WS-Addressing.</p> <p>WS-Addressing properties that are sent in the request or set in the proxy service pipeline are used instead of the callback proxy you set in this option.</p> <p>If you do not specify a Callback Proxy, and the request does not contain ReplyTo properties, you must provide ReplyTo properties in the SOAP header through the proxy service pipeline.</p>
Fault Proxy	This option is not currently supported. You must configure your callback services to handle faults in an asynchronous pattern.

Table 24–1 (Cont.) SOA-DIRECT Transport Configuration

Property	Description
WS-Addressing Version	<p>Required. Enabled only for the Asynchronous Client role.</p> <p>Specify the default WS-Addressing version to use when no WS-Addressing is provided in the request or the proxy service pipeline. WS-Addressing properties that are sent in the request or set in the proxy service pipeline are used instead of the WS-Addressing version you set in this option.</p> <p>For WS-Addressing version mismatches between environments, perform any necessary transformations in the proxy service pipeline. For more information, see Section 24.5.1.4, "Transformation Examples."</p>
Dispatch Policy	<p>Select the instance of Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists.</p> <p>For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i>.</p>
Pass Caller's Subject	<p>Optional. Select this option to have Oracle Service Bus pass the authenticated subject from the proxy service when invoking the SOA service. The Invocation Service Account option, an alternative to Pass Caller's Subject, is disabled when you select this option.</p> <p>Note: Make sure that domain trust is enabled between client and target server if they are in different domains. For more information, see "Important Information Regarding Cross-Domain Security Support" in <i>Oracle Fusion Middleware Securing Oracle WebLogic Server</i>.</p>
Invocation Service Account	<p>Optional. Alternative to Pass Caller's Subject, which lets you specify custom security credentials by selecting a service account for RMI invocation. You can specify any type of service account (Pass Through, Static, Mapping).</p> <p>Click Browse and select a service account. If you do not specify a service account, an anonymous subject is used.</p>

24.4 WS-Addressing Reference

This section describes specific WS-Addressing properties that the SOA-DIRECT transport uses to communicate natively with an Oracle SOA composite. This section also describes ways to provide callback addresses in asynchronous communication, as described in [Section 24.2.2, "Advanced Use Cases – Asynchronous."](#)

See [Section 24.5, "XML Examples"](#) for WS-Addressing examples.

24.4.1 ReplyTo Header

In an asynchronous communication, a service callback is sent on a different connection than the request. As a service developer, you must supply the correct callback address in an asynchronous exchange so that the callback is sent to the correct client. When using the SOA-DIRECT transport with WS-Addressing correlation, the callback address is specified in the "ReplyTo" WS-Addressing header.

24.4.1.1 Calling a SOA Composite Asynchronously with a SOA-DIRECT Business Service

The SOA-DIRECT business service can optionally generate the ReplyTo header. In the business service configuration, if you select a Callback Proxy to handle the callback, the SOA-DIRECT transport sets the correct callback address corresponding to this callback proxy in the ReplyTo header. Note that this header is generated only if the incoming message does not already contain a ReplyTo header.

For more information, see [Section 24.2.2.1, "Asynchronous Invocation of a SOA Composite."](#)

24.4.1.2 Calling Back to a SOA Composite Asynchronously with a SOA-DIRECT Business Service

When calling an external service from an Oracle SOA composite through Oracle Service Bus, you must manually set a callback address. To do this, set the callback address as the ReplyTo value in the proxy service that invokes the callback SOA-DIRECT business service.

For more information, see [Section 24.2.2.2, "Asynchronous Invocation from a SOA Composite."](#)

24.4.2 MessageID / RelatesTo Headers

MessageID and RelatesTo WS-Addressing headers are used to store the conversation ID in conversations between Oracle Service Bus and Oracle SOA service components, ensuring related messages remain in the same conversation.

Unlike ReplyTo, the SOA-DIRECT transport does not provide built-in support for the MessageID or RelatesTo headers. Instead, you must set the correct values for those headers in the pipeline of the proxy service that invokes a SOA-DIRECT business service.

The following describe when MessageID and RelatesTo headers are used in synchronous and asynchronous conversations:

- **Synchronous conversation:** The MessageID header value determines the conversation ID in the initial request. Then, for subsequent requests within the same conversation, the conversation ID must be provided in the RelatesTo header.
- **Asynchronous callbacks** - The MessageID header value determines the conversation ID in the initial request. Then, for the callback, the conversation ID must be provided in the RelatesTo header.

For more implementation on establishing a conversation ID to make sure messages participate in the correct conversation, see [Section 24.2.1.3, "Associating Messages with the Correct Conversation"](#) and the [Section 24.5.1, "Conversation ID Examples."](#)

24.5 XML Examples

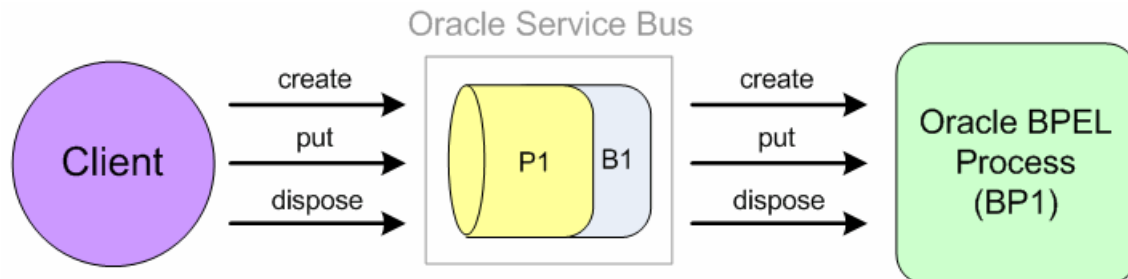
Following are examples of XML messaging between Oracle Service Bus and Oracle SOA service Components.

24.5.1 Conversation ID Examples

This section provides different examples of establishing a conversation ID among messages in a conversation between Oracle Service Bus and Oracle SOA composites.

In [Figure 24–5](#), a client is synchronously invoking a BPEL Process component in an Oracle SOA composite. The business service (B1) uses the SOA-DIRECT transport to invoke a process. The proxy service (P1) handles any necessary conversation ID mapping. The SOA composite exposes the BPEL Process as a direct binding service.

Figure 24–5 Operations in a Synchronous Exchange Through Oracle Service Bus



24.5.1.1 Port and Message Definitions

The examples in this section use the following port and message definitions defined in the WSDL.

```

<wsdl:types>
  <xsd:schema
    targetNamespace="http://www.sample.org/spec/samples/types"
    elementFormDefault="qualified">
    <xsd:complexType name="ValueHolder">
      <xsd:all>
        <xsd:any minOccurs="1"/>
      </xsd:all>
    </xsd:complexType>
  </xsd:schema>
</wsdl:types>
<message name="create"/>
<message name="putRequest">
  <part name="key" type="xsd:string"/>
  <part name="value" type="types:ValueHolder"/>
</message>
<message name="putResponse">
  <part name="value" type="types:ValueHolder"/>
</message>
...
<message name="dispose"/>
<portType name="ServiceMap">
  <operation name="create">
    <input message="tns:create"/>
  </operation>
  <operation name="put">
    <input message="tns:putRequest"/>
    <output message="tns:putResponse"/>
  </operation>
  ...
  <operation name="dispose">
    <input message="tns:dispose"/>
  </operation>
</portType>
  
```

24.5.1.2 WS-Addressing that Sets the Conversation ID

This example shows how WS-Addressing is used to set the conversation ID among messages in a conversation.

Figure 24–5 shows the communication pattern.

Create Operation

```
<soap:Envelope>
  <soap:Header>
    <wsa03:MessageID>uuid:123456789</wsa03:MessageID>
  </soap:Header>
  <soap:Body>
    <create/>
  </soap:Body>
</soap:Envelope>
```

Put Operation

```
<soap:Envelope>
  <soap:Header>
    <wsa03:MessageID>uuid:111111111</wsa03:MessageID>
    <wsa03:RelatesTo>uuid:123456789</wsa03:RelatesTo>
  </soap:Header>
  <soap:Body>
    <put>
      <key>key</key>
      <value>
        <PO/>
      </value>
    </put>
  </soap:Body>
</soap:Envelope>
<soap:Envelope>
  <soap:Body>
    <putResponse>
      <value/>
    </putResponse>
  </soap:Body>
</soap:Envelope>
```

The <put> operation also has a MessageID, but it is ignored because the RelatesTo has a value that provides the conversation ID.

24.5.1.3 Message Payload Data that Sets the Conversation ID

This example shows how message payload data can be used to set the conversation ID among messages in a conversation.

In these examples, the proxy service maps the ID to the MessageID / RelatesTo SOAP headers.

Figure 24–5 shows the communication pattern.

Create Operation

Client to proxy service

```
<soap:Envelope>
  <soap:Body>
    <create/>
  </soap:Body>
</soap:Envelope>
```

```

<soap:Envelope>
  <soap:Body>
    <createResponse>
      <mapID>uuid:123456789</mapID>
    </createResponse>
  </soap:Body>
</soap:Envelope>

```

Proxy service to SOA composite (via a SOA-DIRECT business service)

```

<soap:Envelope>
  <soap:Header>
    <wsa03:MessageID>uuid:123456789</wsa03:MessageID>
  </soap:Header>
  <soap:Body>
    <create/>
  </soap:Body>
</soap:Envelope>

```

Not shown: The ID was generated in the request of the proxy service pipeline and inserted as a <wsa03:MessageID> before invoking the process. On the process side, the Create operation is one-way, so a SOAP response must be created before replying to the client. The response sends back the ID that was generated by the proxy service.

Put Operation

Client to proxy service

```

<soap:Envelope>
  <soap:Body>
    <put>
      <mapID>uuid:123456789</mapID>
      <key>key</key>
      <value>
        <PO/>
      </value>
    </put>
  </soap:Body>
</soap:Envelope>
<soap:Envelope>
  <soap:Body>
    <putResponse>
      <value/>
    </putResponse>
  </soap:Body>
</soap:Envelope>

```

Proxy service to SOA composite (via a SOA-DIRECT business service)

```

<soap:Envelope>
  <soap:Header>
    <wsa03:RelatesTo>uuid:123456789</wsa03:RelatesTo>
  </soap:Header>
  <soap:Body>
    <put>
      <key>key</key>
      <value>
        <PO/>
      </value>
    </put>
  </soap:Body>
</soap:Envelope>

```

```
<soap:Envelope>
  <soap:Body>
    <putResponse>
      <value/>
    </putResponse>
  </soap:Body>
</soap:Envelope>
```

Dispose Operation

Client to proxy service

```
<soap:Envelope>
  <soap:Body>
    <dispose>
      <mapID>uuid:123456789</mapID>
    </dispose>
  </soap:Body>
</soap:Envelope>
```

Proxy service to SOA composite (via a SOA-DIRECT business service)

```
<soap:Envelope>
  <soap:Header>
    <wsa03:RelatesTo>uuid:123456789</wsa03:RelatesTo>
  </soap:Header>
  <soap:Body>
    <dispose/>
  </soap:Body>
</soap:Envelope>
```

24.5.1.4 Transformation Examples

In these examples, the client uses a more recent version of the WS-Addressing spec (wsa04 prefix). The proxy service is responsible for transforming the SOAP headers to use the wsa03 prefix. The proxy service developer configures the transformation.

[Figure 24–5](#) shows communication pattern.

Create Operation

Client to proxy service

```
<soap:Envelope>
  <soap:Header>
    <wsa04:MessageID>uuid:123456789</wsa04:MessageID>
  </soap:Header>
  <soap:Body>
    <create/>
  </soap:Body>
</soap:Envelope>
```

Proxy service to SOA composite (via a SOA-DIRECT business service)

```
<soap:Envelope>
  <soap:Header>
    <wsa03:MessageID>uuid:123456789</wsa03:MessageID>
  </soap:Header>
  <soap:Body>
    <create/>
  </soap:Body>
</soap:Envelope>
```

Put Operation

Client to proxy service

```

<soap:Envelope>
  <soap:Header>
    <wsa04:MessageID>uuid:111111111</wsa04:MessageID>
    <wsa04:RelatesTo>uuid:123456789</wsa04:RelatesTo>
  </soap:Header>
  <soap:Body>
    <put>
      <key>key</key>
      <value>
        <PO/>
      </value>
    </put>
  </soap:Body>
</soap:Envelope>
<soap:Envelope>
  <soap:Body>
    <putResponse>
      <value/>
    </putResponse>
  </soap:Body>
</soap:Envelope>

```

Proxy service to SOA composite (via a SOA-DIRECT business service)

```

<soap:Envelope>
  <soap:Header>
    <wsa03:MessageID>uuid:111111111</wsa03:MessageID>
    <wsa03:RelatesTo>uuid:123456789</wsa03:RelatesTo>
  </soap:Header>
  <soap:Body>
    <put>
      <key>key</key>
      <value>
        <PO/>
      </value>
    </put>
  </soap:Body>
</soap:Envelope>
<soap:Envelope>
  <soap:Body>
    <putResponse>
      <value/>
    </putResponse>
  </soap:Body>
</soap:Envelope>

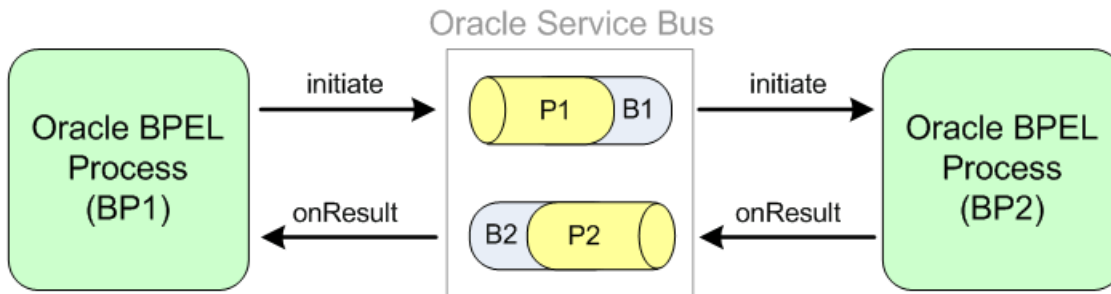
```

24.5.2 Asynchronous Composite to Composite Native Communication Through Oracle Service Bus Example

The following example shows the SOAP headers involved in a SOA composite invoking another SOA composite asynchronously through Oracle Service Bus. The first SOA composite uses a BPEL Process exposed as a direct binding reference to invoke Oracle Service Bus. The second SOA composite uses a BPEL process exposed as a direct binding service to receive requests from Oracle Service Bus.

In [Figure 24–6](#), P1 and P2 are proxy services that pass messages (and perform transformations) to B1 and B2 business services, which are required to make calls to SOA composites using the Oracle Service Bus SOA-DIRECT transport.

Figure 24–6 SOA Composite Invoking an SOA Composite Through Oracle Service Bus



Refer to [Figure 24–6](#) for the following SOAP header examples.

24.5.2.1 Port and Message Definitions

```

<message name="LoanServiceRequestMessage">
  <part name="payload" element="types:loanApplication"/>
</message>
<message name="LoanServiceResultMessage">
  <part name="payload" element="types:loanOffer"/>
</message>
<portType name="LoanService">
  <operation name="initiate">
    <input message="tns:LoanServiceRequestMessage"/>
  </operation>
</portType>
<portType name="LoanServiceCallback">
  <operation name="onResult">
    <input message="tns:LoanServiceResultMessage"/>
  </operation>
</portType>
  
```

24.5.2.2 BP1 to P1 – Initiate operation

```

<soap:Envelope>
  <soap:Header>
    <wsa03:ReplyTo>
      <wsa03:Address>

t3://soaserver:8001/default/AmericanLoanClient/LoanserviceRequester
      </wsa03:Address>
    </wsa03:ReplyTo>
    <MessageID>AmericanLoanClient-1.0/60007</MessageID>
  </soap:Header>
  <soap:Body >
    <loanApplication>
      ...
    </loanApplication>
  </soap:Body>
</soap:Envelope>
  
```

24.5.2.3 P1/B1 to BP2

```

<soap:Envelope>
  
```



```

<soap:Header>
  <wsa03:ReplyTo>
    <wsa03:Address>http://serverB:7001/P2</wsa03:Address>
    <wsa03:referenceParameters>
      <osb:Callback>
        <osb:Address>

t3://soaserver:8001/default/AmericanLoanClient/LoanserviceRequesterRef#Loanservice
RequesterBpel

        </osb:Address>
      </osb:Callback>
    </wsa03:referenceParameters>
  </wsa03:ReplyTo>
  <MessageID>AmericanLoanClient~1.0/60007</MessageID>
</soap:Header>
<soap:Body >
  <loanApplication>
    ...
  </loanApplication>
</soap:Body>
</soap:Envelope>

```

The ReplyTo callback address is set by B1, which gets the value from the Callback Proxy field in the SOA-DIRECT transport configuration, as described in [Section 24.3.2, "SOA-DIRECT Transport Configuration for Business Services."](#) B1's callback proxy is P2.

You must wrap the original replyTo information and send it as reference properties so that it is echoed back in the onResult callback message (to follow).

Note: This sample uses osb:Callback and osb:Address for illustration purpose only. There is no standard or Oracle Service Bus standard elements defined for WS-Addressing support.

24.5.2.4 BP2 to P2 – onResult operation

```

<soap:Envelope>
  <soap:Header>
    <wsa03:RelatesTo>AmericanLoanClient~1.0/60007</wsa03:RelatesTo>
    <osb:Callback>
      <osb:Address>

t3://soaserver:8001/default/AmericanLoanClient/LoanserviceRequesterRef#Loanservice
RequesterBpel

      </osb:Address>
    </osb:Callback>
  </soap:Header>
  <soap:Body >
    <loanOffer>
      ...
    </loanOffer>
  </soap:Body>
</soap:Envelope>

```

The reference property osb:Callback is sent back as a SOAP header by the Oracle BPEL Process Manager engine.

24.5.2.5 P2/B2 to BP1 – onResult operation

```
<soap:Envelope>
  <soap:Header>
    <wsa03:RelatesTo>AmericanLoanClient~1.0/60007</wsa03:RelatesTo>
  </soap:Header>
  <soap:Body >
    <loanOffer>
      ...
    </loanOffer>
  </soap:Body>
</soap:Envelope>
```

Proxy service P2 removes the temporary `osb:Callback` header; but prior to deleting this header, the `replyTo` address value is copied to the `$outbound` variable so that the SOA-DIRECT transport in business service B2 can send the callback message to the correct SOA service component.

JCA Transport

Oracle Service Bus provides a J2EE Connector Architecture (JCA) transport that interacts with Enterprise Information Systems (EIS), such as Enterprise Resource Planning (ERP) systems, letting EIS applications and services participate in the service bus environment. For a list of adapters the JCA transport works with, see [Section 25.2.1, "Adapter Support."](#)

This document, which describes the JCA transport, contains the following sections:

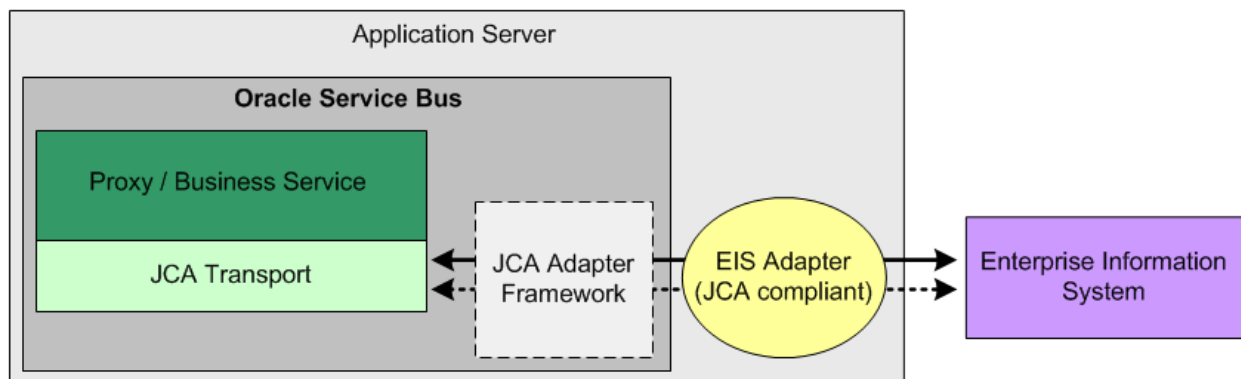
- [Section 25.1, "About the JCA Transport"](#)
- [Section 25.2, "Working with Adapters"](#)
- [Section 25.3, "Invoking an EIS Service Through Oracle Service Bus"](#)
- [Section 25.4, "Invoking an External Service from an EIS"](#)
- [Section 25.5, "Transport Configuration Reference"](#)

25.1 About the JCA Transport

The JCA transport provides native connectivity between Oracle Service Bus and EIS systems, letting those systems interact in the service bus layer and leverage the capabilities and features of Oracle Service Bus.

In JCA proxy or business services, the JCA transport works in conjunction with a built-in JCA adapter framework and JCA-compliant adapters to interact with EIS systems, as shown in [Figure 25-1](#). Solid arrows signify request, dotted arrows signify response.

Figure 25-1 Oracle Service Bus Services Interacting With an EIS



JCA proxy services listen for inbound requests from supported JCA adapters, and JCA business services invoke EIS endpoints through supported adapters.

This section describes features and characteristics of the JCA transport.

25.1.1 Messaging

The JCA transport supports request-only and synchronous request/response messaging patterns using SOAP 1.1.

25.1.1.1 Transactions

The JCA transport is transactional. If a JCA proxy service is invoked in an EIS transaction, or if a JCA business service is invoked in a transaction, the transport propagates the transaction.

25.1.2 Transport Headers

Oracle JCA adapters transmit header information through Normalized Message properties (with the exception of the Oracle JCA Adapter for AQ's payload header). You cannot map these Normalized Message properties to SOAP message headers in Oracle Service Bus, but the Oracle Service Bus transport maps the properties to transport headers. The following subsections list the predefined transport headers in Oracle Service Bus that support the Normalized Message properties in the Oracle Database, AQ, and Applications JCA adapters. Oracle Service Bus maps all other Normalized Message properties to user-defined transport headers.

For more information on Normalized Message properties in JCA adapters, see the *Oracle Fusion Middleware User's Guide for Technology Adapters*.

25.1.2.1 \$inbound and \$outbound Request Headers

Following are predefined Oracle Service Bus transport headers in the \$inbound and \$outbound request variables:

- `jca.aq.Priority` (int)
- `jca.aq.Correlation` (string)
- `jca.aq.HeaderDocument` (string)
- `jca.apps.HeaderDocument` (string)
- `jca.file.FileName`
- `jca.file.Directory`
- `jca.file.Size`
- `jca.file.Batch`
- `jca.file.BatchIndex`
- `jca.file.LastModifiedTime`

25.1.2.2 \$inbound-Only Request Headers

Following are predefined Oracle Service Bus transport headers in the \$inbound request variable:

- `jca.aq.MessageId` (string)
- `jca.aq.Attempts` (int)

- `jca.aq.EnqueueTime` (string)
- `jca.aq.OrigMessageId` (string)

25.1.2.3 \$outbound-Only Request Headers

Following are predefined Oracle Service Bus transport headers in the \$outbound request variable:

- `jca.aq.Delay` (int)
- `jca.aq.Expiration` (int)
- `jca.aq.RecipientList` (string)
- `jca.aq.ExceptionQueue` (string)
- `jca.db.ProxyUserName` (string)
- `jca.db.ProxyPassword` (string)
- `jca.db.ProxyRoles` (string)
- `jca.db.ProxyCertificate` (string)
- `jca.db.ProxyDistinguishedName` (string)
- `jca.db.ProxyIsThickDriver` (boolean)
- `jca.apps.Username`(string)
- `jca.apps.Responsibility`(string)
- `jca.apps.RespApplication`(string)
- `jca.apps.SecurityGroup`(string)
- `jca.apps.NLSLanguage`(string)
- `jca.apps.ORG_ID`(integer)
- `jca.apps.ecx.TransactionType`(string)
- `jca.apps.ecx.TransactionSubtype`(string)
- `jca.apps.ecx.PartySiteId`(string)
- `jca.apps.ecx.MessageType`(string)
- `jca.apps.ecx.MessageStandard`(string)
- `jca.apps.ecx.DocumentNumber`(integer)
- `jca.apps.ecx.ProtocolType`(string)
- `jca.apps.ecx.ProtocolAddress`(string)
- `jca.apps.ecx.Username`(string)
- `jca.apps.ecx.Password`(string)
- `jca.apps.ecx.Attribute1`(string)
- `jca.apps.ecx.Attribute2`(string)
- `jca.apps.ecx.Attribute3`(string)
- `jca.apps.ecx.Attribute4`(string)
- `jca.apps.ecx.Attribute5`(string)
- `jca.apps.ecx.Payload`(string)

25.1.3 Endpoint Properties

The JCA transport supports a number of service endpoint properties you can set in the JCA proxy or business service configuration, as described in [Section 25.5.2, "JCA Transport Configuration for Proxy and Business Services"](#). Following are the supported endpoint properties.

For more information, see "Oracle JCA Adapters Endpoint Properties" in the *Oracle Fusion Middleware Administrator's Guide for Oracle SOA Suite and Oracle Business Process Management Suite* and "Oracle JCA Adapter Properties" in the *Oracle Fusion Middleware User's Guide for Technology Adapters*.

25.1.3.1 Proxy Service Endpoint Properties

Following are the supported endpoint properties in JCA proxy services:

- `jca.retry.count`
- `jca.retry.interval`
- `jca.retry.backoff`
- `jca.retry.maxInterval`

Oracle JCA Adapter for AQ

- `DequeueTimeOut`
- `ConnectionRetryDelay`
- `adapter.aq.dequeue.threads`

Oracle JCA Adapter for Database and Oracle Applications

- `activationInstances`

Oracle JCA Adapter for Files

- `ignoreListingErrors`
- `IgnoreZeroByteFile`
- `jca.message.encoding`
- `notifyEachBatchFailure`
- `oracle.tip.adapter.file.debatching.rejection.quantum`
- `oracle.tip.adapter.file.highavailability.maxRetry`
- `oracle.tip.adapter.file.highavailability.maxRetryInterval`
- `oracle.tip.adapter.file.rejectOriginalContent`
- `oracle.tip.adapter.file.timeout.recoverpicked.minutes`
- `oracle.tip.adapter.file.timeout.recoverunpicked.minutes`
- `recoveryInterval`
- `serializeTranslation`
- `useFileSystem`

25.1.3.2 Business Service Endpoint Properties

Following are the supported endpoint properties in JCA business services:

- `jca.retry.count`

- `jca.retry.interval`
- `jca.retry.backoff`
- `jca.retry.maxInterval`
- `jca.retry.maxPeriod`

Oracle Adapter for Oracle Applications

- Username
- Responsibility

Oracle JCA Adapter for Files

- `inMemoryTranslation`
- `IgnoreZeroByteFile`
- `oracle.tip.adapter.file.mutex`
- `oracle.tip.adapter.file.rejectOriginalContent`
- `serializeTranslation`

25.1.4 Security

This section describes how the JCA transport and JCA adapter framework handle security.

A JNDI service account is used during both endpoint validation and run time.

During endpoint validation, if a static service account is associated with the endpoint, JNDI lookup against the adapter connection factory is performed using the subject in the static service account. If the service does not use a service account, an anonymous subject is used.

At run time, a JCA proxy service supports only static service accounts. If a static service account is configured on the service, the subject in the service account is used to perform the JNDI lookup against the adapter connection factory. Otherwise, an anonymous subject is used.

JCA business services at run time support static, pass through, and mapping service accounts. If a service account is configured on a JCA business service, JNDI lookup against the adapter connection factory and the subsequent outbound invocation is performed using the subject in the service account. If the service does not use a service account, an anonymous subject is used.

Oracle adapters that connect to an EIS database (Oracle Adapters for Database, AQ, and Oracle Applications) connect using the JDBC datasource associated with the adapter-managed connection factory. Other Oracle adapters, such as the Oracle JCA Adapter for Files, support Oracle WebLogic Server container-managed and application-managed sign-on for outbound connections. For more information, see "Securing Enterprise Information System Credentials" in the *Oracle Fusion Middleware User's Guide for Technology Adapters*.

25.1.4.1 Proxy Services

On inbound requests the JCA adapter framework, which invokes a JCA proxy service, always invokes the proxy service as anonymous.

25.1.4.2 Business Services

Depending on which type of JNDI service account is used in a JCA business service, the outbound service endpoint is invoked with a subject in the following ways:

- No service account is used – An anonymous subject is used to invoke outbound JCA endpoint.
- A pass-through service account is used – The subject associated with the inbound request is used to invoke the outbound JCA endpoint.
- A static service account is used – The subject associated with the static user name/password in the service account is used to invoke the outbound JCA endpoint.
- A mapping service account is used – The subject associated with the mapped user name/password in the service account is used to invoke the outbound JCA endpoint.

25.1.5 Logging

The JCA transport logs messages using the Oracle WebLogic Server NonCatalog logger. JCA adapter framework logs are redirected to the Oracle WebLogic Server log.

Debug log records generated by the JCA transport, JCA adapter framework, and JCA adapters are redirected to the Oracle WebLogic Server log only if Oracle Service Bus transport debug logging is enabled. To enable the debug mode, before starting the server, edit the `alsbdebug.xml` file in the domain directory and set the category `alsb-jca-framework-adapter-debug` flag to `true`. JCA framework and adapter messages are logged with the tag `JCA_FRAMEWORK_AND_ADAPTER`.

Note: The `alsb-transport-debug` flag, which is used for all other transports, has no effect on the JCA transport, framework, and adapters.

For more information, see "Debugging Oracle Service Bus" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

25.1.6 URI Rewriting

The JCA transport supports URI rewriting in a proxy service message flow.

25.1.7 Environment Variables

The JCA transport declares the following environment variable values, which can be maintained when moving an Oracle Service Bus configuration among different deployment environments.

For descriptions of these values, see [Section 25.5.2, "JCA Transport Configuration for Proxy and Business Services."](#)

- JCA JNDI Service Account (`JCA_ENV_JNDI_SERVICE_ACCOUNT`)
- Work Manager
- JCA Always Use WSDL Flag (`JCA_ENV_ALWAYS_USE_WSDL`)
- JCA Connection Mode (`JCA_ENV_CONNECTION_MODE`)

- JCA Overwrite Connection Authentication Flag (JCA_ENV_OVERWRITE_CONNECTION_FACTORY_AUTHENTICATION_PROPERTIES)
- JCA Authentication Overwrite Service Account (JCA_ENV_CONNECTION_AUTHENTICATION_OVERWRITE_SERVICE_ACCOUNT)

25.1.8 Encoding

For inbound requests and outbound responses, the JCA adapter framework sends messages to JCA proxy services with UTF-8 encoding.

25.2 Working with Adapters

The JCA transport is compatible with a variety of EIS adapters. This section describes how to use the supported adapters with Oracle Service Bus and the JCA transport.

A JCA file and WSDL are required to create a JCA proxy or business service in Oracle Service Bus. For proxy or business services that interact with supported adapters, you must generate the JCA files and WSDLs in your EIS system environment. After you generate JCA files and WSDLs, import them into your Oracle Service Bus configuration and use the JCA file to create a JCA service.

For example, when creating services that interact with Oracle JCA adapters, generate a JCA file and WSDL in Oracle JDeveloper, import the files into the Oracle Service Bus IDE, and create and configure JCA services based on the JCA file. For more information, see [Section 2.2.2, "Generating a JCA Business Service from an Outbound JCA File"](#) and [Section 2.3.2, "Generating a JCA Proxy Service from an Inbound JCA File."](#)

You can also import the JCA and WSDL files into the Oracle Service Bus Console for JCA service generation, as described in the console online help.

This section contains the following topics:

- [Section 25.2.1, "Adapter Support"](#)
- [Section 25.2.2, "JCA Adapter Framework"](#)
- [Section 25.2.3, "Configuring Adapters – General"](#)
- [Section 25.2.5, "Configuring the Oracle JCA Adapter for Database"](#)
- [Section 25.2.6, "Configuring the Oracle JCA Adapter for AQ"](#)
- [Section 25.2.7, "Rejected Messages"](#)

25.2.1 Adapter Support

The Oracle Service Bus JCA transport lets you interact with the following JCA-compliant adapters:

- Oracle Adapter for Oracle Applications
- Oracle JCA Adapter for AQ
- Oracle JCA Adapter for Database
- Oracle JCA Adapter for Files
- Oracle BAM Adapter (Business Activity Monitoring)
- PeopleSoft (Oracle Application Adapters 10g)
- SAP R/3 (Oracle Application Adapters 10g)

- Siebel (Oracle Application Adapters 10g)
- J.D. Edwards (Oracle Application Adapters 10g)

See the following guides for more information on Oracle adapters:

- *Oracle Fusion Middleware User's Guide for Technology Adapters*
- *Oracle Fusion Middleware User's Guide for Oracle Business Activity Monitoring*
- Oracle Application Adapters 10g documentation at http://download.oracle.com/docs/cd/E13199_01/oracleappsadapter/docs1031/

25.2.1.1 Oracle JCA Adapter Limitations

Following are limitations when using some JCA adapters with Oracle Service Bus:

- Oracle JCA Adapter for Files – Attachments (large payload support), pre- and post-processing of files, using a re-entrant valve for processing ZIP files, content streaming, and file chunked read are not supported with Oracle Service Bus.
- Oracle JCA Adapter for AQ – Streamed payload is not supported with Oracle Service Bus.

25.2.2 JCA Adapter Framework

The JCA Transport uses the Oracle Service Bus JCA adapter framework to interact with JCA-compliant adapters that in turn provide connectivity to external EIS services. The JCA adapter framework abstracts the complexity of interacting with those adapters, letting you focus on proxy and business service development in Oracle Service Bus.

For inbound interactions, the JCA proxy service registers a listener with the associated JCA adapter endpoint. When an event occurs in the EIS system where a message is sent to the JCA proxy, and the JCA adapter framework invokes the proxy service with a request-only or request-response pattern.

On outbound interactions, when a client invokes an EIS service through Oracle Service Bus, the JCA business service invokes the JCA adapter endpoint through the JCA adapter framework.

No configuration of the JCA adapter framework is necessary. It is deployed and functions automatically as you create JCA proxy and business services in Oracle Service Bus and deploy your adapters as required.

25.2.3 Configuring Adapters – General

The Oracle JCA adapters (such as Oracle JCA Adapters for AQ, Database, and Oracle Applications) are deployed automatically in an Oracle Service Bus domain. (You must manually install and deploy other supported third-party resource adapters in the Oracle WebLogic Server Console.)

Create a data source for and configure each adapter in Oracle WebLogic Server. Perform the following general steps to configure an already-deployed Oracle adapter (such as AppsAdapter, AqAdapter, or DbAdapter).

1. Create a data source – In the Oracle WebLogic Server Console, create and configure a data source to connect to the EIS database.
2. Configure the deployed resource adapter – In the Oracle WebLogic Server Console, modify the adapter configuration to use the data source you created.

- a. Go to **Deployments > [adapter] > Configuration > Outbound Connection Pools**.
- b. Create a new connection pool factory based on the new data source. During the process, designate a location for the connection pool factory Plan.xml file. This file persists the connection factory configuration settings.
- c. After you create the connection pool factory, click its name and modify its properties. To do this, click in the **Properties** field for XADataSourceName (for XA data sources) or dataSourceName (for non-XA data sources), enter the name of the data source you created, and press **Enter**.

Note: You *must* press **Enter** after you enter the name of the data source, while the cursor is in the input field. If you do not do this, the connection pool factory properties are not saved.

- d. Activate your changes in the Change Center.

See the following sections for adapter-specific configuration information.

25.2.4 Configuring Adapters that Poll a Database

By default, as a performance best practice, the Oracle adapters that poll a database use one thread to poll the database (NumberOfThreads=1 property in the activation spec). Because the adapter never releases that thread, which is by design, you may see a stuck thread stack trace in the server log. If you set the NumberOfThreads to more than one, you may see stack traces for all of those threads. You can ignore stuck thread stack traces.

To prevent stuck thread stack traces from appearing for the adapter, configure a Work Manager that includes the following setting:

```
<ignore-stuck-threads>true</ignore-stuck-threads>
```

In the JCA proxy service, configure the transport's Dispatch Policy to use the Work Manager you created.

For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in *Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server*.

25.2.5 Configuring the Oracle JCA Adapter for Database

Following are key configuration points between Oracle Service Bus and the Oracle Database Adapter. For more detailed configuration instructions, see the *JCA DB Adapter with OSB* tutorial at http://www.oracle.com/technology/sample_code/products/osb/index.html.

- **Data Source:** For the Oracle JCA Adapter for Database data source, set the following options:

Option	Setting
Initial Capacity	0
Test Connections on Reserve	selected
Test Frequency	5
Test Table Name	SQL SELECT 1 FROM DUAL

Option	Setting
Seconds to Trust an Idle Pool Connection	0
Connection Creation Retry Frequency	10

Note: Only one proxy service can poll a specific Oracle Database table.

25.2.6 Configuring the Oracle JCA Adapter for AQ

For the Oracle JCA Adapter for AQ data source, select the **Test Connections on Reserve** option.

25.2.7 Rejected Messages

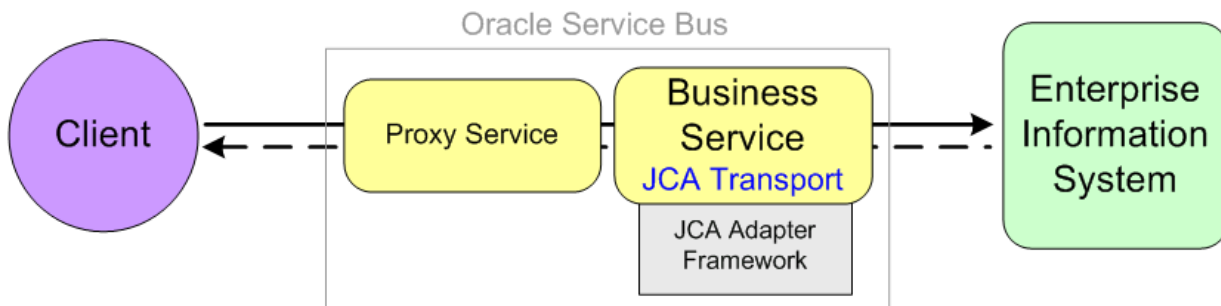
The JCA adapter framework automatically logs rejected messages—messages with data errors—to a `/domain/jca/.../rejectedMessages` directory for each adapter.

25.3 Invoking an EIS Service Through Oracle Service Bus

This section describes the process of invoking an EIS service from a client through Oracle Service Bus.

The JCA transport supports request and synchronous request/response message patterns. This section illustrates a synchronous request/response pattern, as shown in [Figure 25-2](#).

Figure 25-2 A Client Invoking an EIS Service Through Oracle Service Bus



25.3.1 Creating, Configuring, and Invoking the Services

Use the following guidelines to invoke an EIS service from a client.

For configuration issues that apply to specific adapters, see [Section 25.2, "Working with Adapters."](#)

- Create a Business Service in Oracle Service Bus that represents the EIS service you want to invoke.

Create an outbound JCA business service in Oracle Service Bus. For more information, see [Section 2.2.2, "Generating a JCA Business Service from an Outbound JCA File."](#)

For additional business service configuration details, see [Section 2.2, "Working with Business Services."](#)

- Create a Proxy Service in Oracle Service Bus that invokes the business service.
 - In the proxy message flow, perform any transformation necessary to send messages in the JCA transport's supported message format, described in the [Section 25.1.1, "Messaging"](#) section.
 - Invoke the proxy service from the client with the desired payload.

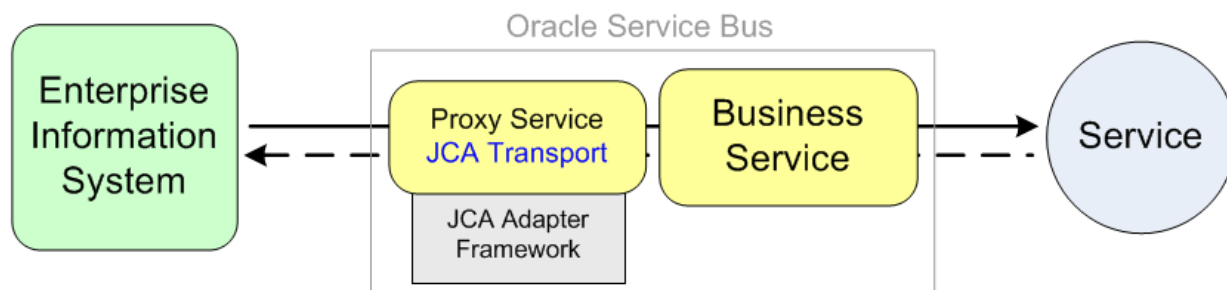
For additional proxy service configuration details, see [Section 2.3, "Working with Proxy Services"](#).

25.4 Invoking an External Service from an EIS

This section describes the process of invoking an external service from an EIS through Oracle Service Bus.

The JCA transport supports request and synchronous request/response message patterns. This section illustrates a synchronous request/response pattern, as shown in [Figure 25–3](#).

Figure 25–3 An EIS Invoking an External Service Through Oracle Service Bus



25.4.1 Creating, Configuring, and Invoking the Services

Use the following guidelines to invoke an external service from an EIS.

For configuration issues that apply to specific adapters, see [Section 25.2, "Working with Adapters."](#)

- Create a Business Service in Oracle Service Bus that represents the external service you want to invoke.

For business service configuration details, see [Section 2.2, "Working with Business Services."](#)
- Create a Proxy Service in Oracle Service Bus that invokes the business service.
 - Create an inbound JCA proxy service in Oracle Service Bus. For more information, see [Section 2.3.2, "Generating a JCA Proxy Service from an Inbound JCA File."](#)
 - In the proxy message flow perform any transformation necessary to send messages in the supported message format of the external service you want to invoke.
 - Invoke the proxy service from the EIS as you normally would.

When you create a JCA proxy service in Oracle Service Bus, the JCA transport registers a listener with the JCA adapter endpoint that listens for service invocations from the EIS. The JCA adapter framework passes service invocations to the proxy service.

For additional proxy service configuration details, see [Section 2.3, "Working with Proxy Services"](#).

25.5 Transport Configuration Reference

This section provides descriptions for JCA transport-specific configuration options. For descriptions of general business and proxy service configuration options, see the following topics:

- *Oracle Fusion Middleware Developer's Guide for Oracle Service Bus*
 - [Section 2.2, "Working with Business Services"](#)
 - [Section 2.3, "Working with Proxy Services"](#)

25.5.1 Endpoint URI

Use the following endpoint URI format for JCA services:

```
jca://<resource_adapter_jndi>
```

where `resource_adapter_jndi` is the value of the "location" attribute in the connection-factory element in the JCA file. This value matches the adapter connection factory JNDI.

25.5.1.1 Endpoint Redeployment

JCA service endpoints are dependent on WSDLs, service accounts, JCA resources, XML schemas, and XML resources (TopLink or EclipseLink mapping files). When any of those resource types is updated and saved for a service, the JCA service endpoint is automatically deleted, recreated, and redeployed. For a JCA proxy service, a new adapter listener is also initialized to listen for inbound requests.

JCA endpoint redeployment has a potential impact on services at run time, depending on whether or not you select the "Always use configuration from JCA file" option for a service as described in [Table 25–1](#). For example:

- If "Always use configuration from JCA file" is selected, after a JCA endpoint is redeployed, the updated JCA file is used to connect and interact with the resource adapter at run time.
- If "Always use configuration from JCA file" is not selected, existing service configuration overrides are used to interact with the resource adapter at run time instead of corresponding JCA file properties that may have been updated.

25.5.2 JCA Transport Configuration for Proxy and Business Services

[Table 25–1](#) describes the options available on the JCA transport configuration page in either Eclipse or in the Oracle Service Bus console.

Table 25–1 JCA Transport Configuration Options

Option	Description
JCA File	<p>Click Browse to select a JCA resource. The JCA resource defines different aspects of the service, such as details about the adapter used, a binding to the WSDL and TopLink or EclipseLink mapping file, and the activation/interaction spec properties required by the service.</p> <p>Once you select a valid JCA resource, the remaining transport configuration fields become available.</p>
Adapter Name	A read-only value showing the name of the adapter that the JCA service will use.
Adapter Type	A read-only value showing the adapter type.
Dispatch Policy	<p>Select the instance of Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists.</p> <p>For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i>.</p>
JNDI Service Account	<p>JNDI Service Account is for JNDI context security, used to access the EIS adapter managed connection factory. Click Browse and select a service account. If no service account is specified, an anonymous subject is used.</p> <p>For JCA business services, there is no restriction on the type of JNDI service account that can be configured, such as static or pass-through, but the run time must be able to access a user name and password. JCA proxy services can use only static JNDI service accounts.</p> <p>For more information on JNDI service accounts, see Section 25.1.4, "Security."</p>
EndPoint Properties	<p>This field lets you assign values to endpoint properties such as retries for the type of adapter the service uses.</p> <p>For a list of supported endpoint properties, see Section 25.1.3, "Endpoint Properties."</p>
Dynamic EndPoint Properties	<p>This option lets you pass request parameters to JCA-compliant services. For example, you can use a dynamic endpoint property to pass database query parameters to the Oracle JCA Adapter for Database.</p> <p>For more information on querying with parameters, see "Oracle JCA Adapter for Database" in the <i>Oracle Fusion Middleware User's Guide for Technology Adapters</i>.</p> <p>Enter a name/value pair for each dynamic endpoint property you want to provide. The endpoint property key matches the query parameter name.</p>
Always use configuration from JCA file	<p>This option determines whether or not Activation Spec Properties (proxy services) and Interaction Spec Properties (business services) are always used from the JCA file.</p> <p>If this option is selected (default), the JCA transport interacts with the JCA framework using the activation/interaction spec properties in the JCA file.</p> <p>If this option is deselected, you can override the Activation/Interaction Spec Properties.</p> <p>For the redeployment impact of using this option, see Section 25.5.1.1, "Endpoint Redeployment."</p>

Table 25–1 (Cont.) JCA Transport Configuration Options

Option	Description
Operation Name	Displays a read-only name of the selected WSDL operation. An operation can have its own activation/interaction spec properties, shown in the Activation/Interaction Spec Properties field.
Activation/Interaction Spec Properties	<p>Activation Spec Properties is the field name for proxy services; Interaction Spec Properties is the field name for business services.</p> <p>If this service is an inbound service invoked by an EIS application, this field displays the activation spec properties for the JCA inbound operation shown in Operation Name field.</p> <p>You can override the activation/interaction spec properties if you deselect Always use configuration from JCA file.</p> <p>Note: For Oracle Adapter Suite adapters, activation/interaction spec properties are displayed as read-only. The Oracle Adapter Suite adapters store their own configurations, which you must change in the Oracle Adapter Suite management tools.</p>
Connection properties (legacy)	<p>For legacy JCA services that use non-managed mode connection properties (deprecated in this release), see the connection configuration options at http://download.oracle.com/docs/cd/E13159_01/osb/docs10gr3/jcatransport/transport.html#wp1105451.</p>

For more information on endpoint and activation/interaction spec properties, see the *Oracle Fusion Middleware User's Guide for Technology Adapters*.

25.5.3 Proxy Service Operation Configuration

JCA WSDLs support document literal binding only. The only algorithm for Operation configuration in JCA proxy services is SOAPAction Header. Oracle Service Bus effective WSDLs always contain SOAPAction. The value of the SOAPAction field is the operation name.

25.5.4 Proxy Service Message Handling

The JCA transport does not support message content streaming or attachments. Enabling content streaming has no effect on the JCA transport at run time. Since content streaming is a service configuration option over which the transport has no control, the transport does not raise validation errors if you enable content streaming.

25.5.5 Policies

Setting policies on JCA services is not allowed. If you set a policy on a JCA service, Oracle Service Bus shows a conflict in the service configuration.

25.5.6 Business Service Retry Application Errors

Application errors cannot be retried with the JCA transport, and the transport has no control of this option. Set this option to "No."

HTTP and Poller Transports

You can use different types of transports to configure proxy services or business services in Oracle Service Bus. The transport protocol you select depends on the service type, the type of authentication required, the service type of the invoking service, and so on.

Poll-based transports are transports with transport pollers pinned to a managed server. These transports use the JMS framework to ensure that the processing of a message is done at least once. E-mail, File, FTP, and SFTP are poll-based transports. This section describes the poll-based transports and the HTTP transport.

This document includes the following sections:

- [Section 26.1, "HTTP Transport"](#)
- [Section 26.2, "E-mail Transport"](#)
- [Section 26.3, "File Transport"](#)
- [Section 26.4, "FTP Transport"](#)
- [Section 26.5, "SFTP Transport"](#)

26.1 HTTP Transport

The HTTP transport lets you send messages between clients and service providers through Oracle Service Bus using the http/s protocol. The HTTP transport also provides support for working with REST (Representational State Transfer) environments, as described in [Section 26.1.3, "REST Support."](#)

You can select the HTTP transport protocol when you configure any type of proxy or business service. Use the following endpoint URI formats:

- **Proxy Services:** /someService
- **Business Services:** http://host:port/someService

where `someService` is the name of proxy service or a business service

26.1.1 Configuring Proxy Services using the HTTP Transport

[Table 26–1](#) describes the parameters you can specify to configure the HTTP transport for a proxy service.

Table 26–1 Parameters for Configuring HTTP Transport for Proxy Service

Parameter	Description
HTTPS Required	<p>Select this check box for inbound HTTPS endpoints.</p> <p>To learn more, see Chapter 49, "Configuring Transport-Level Security."</p>
Authentication	<p>You must configure one of the following authentication methods:</p> <ul style="list-style-type: none"> ■ None - Specifies that authentication is not required. ■ Basic - Specifies that basic authentication is required to access this service. <ul style="list-style-type: none"> Basic authentication instructs Oracle WebLogic Server to authenticate the client using a user name and password against the authentication providers configured in the security realm, such as a Lightweight Directory Access Protocol (LDAP) directory service and Windows Active Directory. The client must send its user name and password on the HTTP request header. Basic authentication is strongly discouraged over HTTP because the password is sent in clear text. However, it is safe to send passwords over HTTPS because HTTPS provides an encrypted channel. Caution: By default, all users (authorized and anonymous) can access a proxy service. To limit the users who can access a proxy service, create a transport-level authorization policy. See "Editing Transport-Level Access Policies" in the <i>Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus</i>. ■ Client Certificate - Specifies encrypted communication and strong client authentication (two-way SSL). To learn more, see Chapter 49, "Configuring Transport-Level Security." ■ Custom Authentication - Specifies that an authentication token is contained in an HTTP header. The client's identity is established through the use of this client-supplied token. You must configure an Identity Assertion provider that maps the token to an Oracle Service Bus user. <p>The custom authentication token can be of any active token type supported by a configured Oracle WebLogic Server Identity Assertion provider.</p>
Dispatch Policy	<p>Select a dispatch policy for this endpoint. Leave blank to use the default dispatch policy.</p> <p>Dispatch policy refers to the instance of WLS Work Manager that you want to use for the service endpoint.</p> <p>For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i>.</p>

Table 26–1 (Cont.) Parameters for Configuring HTTP Transport for Proxy Service

Parameter	Description
Request Encoding	<p>The following describe request encoding for inbound and outbound transports:</p> <ul style="list-style-type: none"> For HTTP inbound transports: <p>If the character set encoding parameter of the <code>Content-Type</code> header is not specified in Client Request, enter a character set encoding parameter. If you do not enter a value, the field defaults to <code>ISO-8859-1</code>.</p> For HTTP outbound transports: <p>If you have not configured a request encoding, the Oracle Service Bus run time decides the most appropriate encoding while it makes a request to the business service. In the case of a non-pass-through scenario, the default character encoding is <code>UTF-8</code> at run time. However if it is a pass-through scenario, the run time will pass through the encoding received with the outbound response.</p>
Response Encoding	Accept the default <code>ISO-8859-1</code> as the character set encoding for responses in HTTP transports, or enter a different character set encoding.
Authentication Header	<p>Enter the HTTP header (any except Authorization) from which Oracle Service Bus is to extract the token. This field is available only if you selected the Custom Authentication check box.</p> <p>For example, <code>client-xyz-token</code>.</p>
Authentication Token Type	Select an authentication token type. Only the active token types configured for an Identity Assertion provider are available. This field is available only if you selected the Custom Authentication check box.

For more information on how to configure HTTP transport based proxy services and advanced settings, see [Section 2.3, "Working with Proxy Services."](#)

26.1.2 Configuring Business Services using the HTTP Transport

You must select HTTP as the transport protocol when you configure any type of business service based on HTTP and the endpoint URI is of the form:

```
http://<host:port/someService>
```

where:

- `host`: is the name of the system that hosts the service.
- `port`: is the port number at which the connection is made.
- `someService`: is a target service.

Note: You must specify the following endpoint URI when you configure a business service based on HTTPS.

```
https://<host:port/someService>
```

[Table 26–2](#) describes all the parameters you can specify to configure HTTP transport for a business service.

Table 26–2 Parameters for Configuring HTTP Transport for Business Service

Parameter	Description
Read Timeout	Enter the read timeout interval in seconds. A zero (0) value indicates no timeout.
Connection Timeout	Enter the connection timeout interval in seconds. If the timeout expires before the connection can be established, Oracle Service Bus raises a connection error. A zero (0) value indicates no timeout.
HTTP Request Method	<p>This parameter lets you to use one of the following HTTP methods in a request:</p> <ul style="list-style-type: none"> ■ POST – Passes all its data, of unlimited length, directly over the socket connection as part of its HTTP request body. The exchange is invisible to the client, and the URL does not change. For REST-based requests, POST tells the transport to perform a create/replace operation or perform an action with the request. ■ GET – You can include as part of the request some of its own information that better describes what to get. This information is passed as a sequence of characters appended to the request URL in a query string. You can use GET in a business service with a Service Type of "Any XML Service," or with a Service Type of "Messaging Service" when the Request Message Type is set to "None." For REST-based requests, GET retrieves a representation of a remote resource. ■ PUT – You can use PUT in a business service with a Service Type of "Any XML Service" or "Messaging Service." PUT tells the transport to perform a create/replace operation with a REST-based request, such as uploading a file to a known location. ■ HEAD – You can use HEAD in a business service with a Service Type of "Any XML Service," or with a Service Type of "Messaging Service" when the Response Message Type is set to "None." HEAD tells the transport to get header information for a remote resource rather than getting a full representation of the resource in a REST-based request. ■ DELETE – You can use PUT in a business service with a Service Type of "Any XML Service" or "Messaging Service." DELETE tells the transport to perform a delete operation with a REST-based request. <p>Note: If a method is already set in the \$outbound/transport/request/http:http-method variable, that value takes precedence over any method you select for HTTP Request Method.</p>

Table 26–2 (Cont.) Parameters for Configuring HTTP Transport for Business Service

Parameter	Description
Authentication	<p>Select one of the following:</p> <ul style="list-style-type: none"> ■ None - Specifies that authentication is not required to access this service. ■ Basic - Specifies that basic authentication is required to access this service. <p>Basic authentication instructs Oracle WebLogic Server to authenticate the client using a user name and password against the authentication providers configured in the security realm, such as a Lightweight Directory Access Protocol (LDAP) directory service and Windows Active Directory. The client must send its user name and password on the HTTP request header.</p> <p>Basic authentication is strongly discouraged over HTTP because the password is sent in clear text. However, it is safe to send passwords over HTTPS because HTTPS provides an encrypted channel.</p> <p>Caution: By default, all users (authorized and anonymous) can access a business service. To limit the users who can access a business service, create a transport-level authorization policy. See "Editing Transport-Level Access Policies" in the <i>Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus</i>.</p> <ul style="list-style-type: none"> ■ Client Certificate - Specifies encrypted communication and strong client authentication (two-way SSL). To learn more, see Chapter 49, "Configuring Transport-Level Security."
Service Account	<p>A service account is an alias resource for a user name and password. This is a required field if you selected the Basic Authentication Required field.</p> <p>For more information, see Section 2.1.16, "Creating Service Account Resources."</p>
Dispatch Policy	<p>Select the instance of Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists.</p> <p>For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i>.</p>
Request Encoding	<p>Accept the default <code>iso-8859-1</code> as the character set encoding for requests in HTTP transports, or enter a different character set encoding.</p>
Response Encoding	<p>Accept the default <code>iso-8859-1</code> as the character set encoding for responses in HTTP transports, or enter a different character set encoding.</p>
Proxy Server	<p>Enter a proxy server resource or click Browse to choose an entry from the list of configured proxy server resources.</p>

Table 26–2 (Cont.) Parameters for Configuring HTTP Transport for Business Service

Parameter	Description
Follow HTTP redirects	Select this check box to specify that HTTP redirects (which are requests with a response code 3xx) should be automatically followed. A redirect occurs when you send an outbound request to the URL of a business service, and that service returns a response code (for example, 302) that says the URL is no longer valid and this request needs to be sent to another URL. If you select the Follow HTTP Redirects check box, Oracle Service Bus automatically re-sends the request to the new URL without any action on your part. Deselect this check box if you do not want the HTTP redirects to be automatically followed.
Use Chunked Streaming Mode	Select this option if you want to use HTTP chunked transfer encoding to send messages. Note: Do not use chunked streaming with if you use the Follow HTTP Redirects option. Redirection and authentication cannot be handled automatically in chunked mode.

For more information on how to configure this transport, see [Section 2.2, "Working with Business Services."](#)

26.1.3 REST Support

The HTTP transport provides support for working with REST (Representational State Transfer) environments through Oracle Service Bus, whether you have REST clients that need to interact with non-REST service providers, non-REST clients that need to interact with REST-based service providers, or REST-to-REST services you want to expose through Oracle Service Bus.

In a REST pattern, you invoke HTTP methods (such as GET, PUT, HEAD, POST, and DELETE) on resources that are located at specific URLs. For example, when a user updates his own profile information in a Web application that uses REST, a POST action updates the user information in the database through the service's REST API.

Oracle Service Bus provides the following placeholder variables for handling REST-based requests for inbound and outbound communication:

- **\$inbound or \$outbound/transport/request/http:http-method** – For handling HTTP methods such as GET, PUT, HEAD, POST, and DELETE.
- **\$inbound or \$outbound/transport/request/http:query-string** – For handling query strings in a URL. For example, in the URL `http://localhost:7021/myproxy/weather?operation=temperature&pincode=80439/`, the query string is `operation=temperature&pincode=80439/`.
- **\$inbound or \$outbound/transport/request/http:relative-URI** – For handling relative portions of a REST resource URL (relative to the proxy service URI). For example, in the URL `http://localhost:7021/myproxy/weather/`, `/weather` is a relative URL to `http://localhost:7021/myproxy/`.

26.1.3.1 REST in Proxy Services

With an Oracle Service Bus proxy service, you have the flexibility to interact with REST patterns, whether you are receiving REST-based requests or generating REST-based actions.

For example, if your team wants to develop REST-based applications and invoke services in a non-REST service provider, you can send REST operations through a proxy service and transform those operations into a format the service provider

understands; or you could transform a non-REST request into a resource URL and invoke an operation in a REST-based service provider.

26.1.3.1.1 XQuery Examples Following are XQuery examples of URI parsing using HTTP variables in a proxy server.

Relative-URI

A proxy service has a URI `http://localhost:7001/weather`, and you want to capture the relative URI parts of a request. You create the following XQuery:

```
<relative-URI>
{
  for $c in
  fn:tokenize($inbound/ctx:transport/ctx:request/http:relative-URI, "/")
  where fn:string-length($c) != 0
  return
  <part>
  {$c}
  </part>
}
</relative-URI>
```

If a request comes with the URI of `http://localhost:7001/weather/temperature/35457`, the relative-URI will be `/temperature/35457`, and the XQuery output will be:

```
<relative-URI>
  <part>temperature</part>
  <part>35457</part>
</relative-URI>
```

Query-String

A proxy service has a URI `http://localhost:7001/weather`, and you want to capture the URL query string. You create the following XQuery:

```
<query-params>
{
  for $c in
  fn:tokenize($inbound/ctx:transport/ctx:request/http:query-string, "&")
  return
  <param name="{fn:substring-before($c, "=")}"
  value="{fn:substring-after($c, "=")}"></param>
}
</query-params>
```

If a request comes with a URI of `http://server:7001/weather?operation=temperature&pincode=35457`, the query-string will be `operation=temperature&pincode=35457`, and the XQuery output will be:

```
<query-params>
  <param name='operation' value='temperature' />
  <param name='pincode' value='35457' />
</query-params>
```

26.1.3.1.2 Headers If your service requires specific headers to handle HTTP/REST methods, create user-defined HTTP header variables in your proxy service message flow.

26.1.3.2 REST in Business Services

With an Oracle Service Bus business service, you can invoke REST-based services.

For REST operations, the HTTP transport uses the value in the \$outbound/transport/request/http:http-method variable. If that variable does not supply an HTTP method, the HTTP transport lets you select one of the following **HTTP Request Methods** in the transport configuration: POST, PUT, HEAD, GET, AND DELETE.

Note: If the business service uses a Service Type of WSDL Web Service, only the POST method is available.

Using the \$outbound/transport/request-http/http-method variable, you can also supply your own methods. For example, you can use COPY, MOVE, and LOCK for WebDAV environments (Web-based Distributed Authoring and Versioning).

Use the following guidelines for setting \$outbound variables:

- The transport does not provide run-time validation for custom methods or for manually set supported methods that do not comply with the constraints described in this section.
- Since \$outbound is only available in a Routing node, you cannot specify a method name at run time for publish and service callout actions.
- If the \$outbound query-string is set, the business service passes the query string as part of the URI while invoking an external service.
- If the \$outbound relative-URI is set, the business service uses that value to generate the URI, which is relative to the business service URI.

For example, in a business service with a URI of

http://service.com/purchaseOrder

and the following HTTP variables

\$outbound/transport/request-http/relative-URI: "/PO12367" and

\$outbound/transport/request-http/query-string: "item=NO1&color=black"

The final resolved URI is

http://service.com/purchaseOrder/PO12367?item=NO1&color=black

26.1.3.2.1 Response Codes for HTTP Business Services The HTTP transport provides the following response codes for HTTP methods:

Table 26–3 Response codes for HTTP business services

Method	Response Codes
POST	200 (OK)
	201 (Created)
	204 (No Content)
PUT	200 (OK)
	201 (Created)
	204 (No Content)
	301 (Moved Permanently) – The server sends the response code. The business service handles this response by re-sending the original request.
	200 (OK)
HEAD, GET	200 (OK)

Table 26–3 (Cont.) Response codes for HTTP business services

Method	Response Codes
DELETE	200 (OK) 202 (Accepted) 204 (No Content)

26.2 E-mail Transport

You can select the e-mail transport protocol when you configure a Messaging Type or Any XML Service type of proxy service or business service. The following topics describe how to configure proxy services and business service using the E-mail transport.

Note: E-mail transport supports one-way messaging for services of Messaging Services type.

When you create a messaging type proxy service or a messaging type business service using e-mail transport you must set the response type to none in the Message Type configuration page.

26.2.1 Configuring Proxy Services Using the E-mail Transport

When you configure a proxy service using the e-mail transport, you must specify an endpoint URI in the following format:

```
mailfrom:<mailserver-host:port>
```

where `mailserver-host` is the name of the host mail server port: is the port used by the mail server host.

[Table 26–4](#) describes the parameters you can configure for an e-mail transport based proxy service.

Table 26–4 Parameters for Configuring E-mail Transport for Proxy Services

Parameter	Description
Service Account	This is a mandatory parameter. This is the service account resource. The service account consists of a user name/password combination required to access the e-mail account.
Polling Interval	This is a mandatory parameter. This parameter specifies the polling interval in milliseconds. The default value is 60 ms.
E-mail Protocol	This is a mandatory parameter. There are two types of protocol from which you can select, <code>imap</code> and <code>pop3</code> . The default protocol is <code>pop3</code> .
Read Limit	This is a mandatory parameter. This specifies the number of files to be read in each poll. The default value is 10.
Pass By Reference	If this parameter is enabled, the file is staged in the archive directory and passed as a reference in the message headers.
Post Read Action	This is a mandatory parameter. This specifies whether the files should be deleted, moved, or archived after being read by the service. By default the files are deleted after reading.

Table 26–4 (Cont.) Parameters for Configuring E-mail Transport for Proxy Services

Parameter	Description
Attachments	This is a mandatory parameter. This parameter specifies if the attachments are to be archived or ignored. By default this parameter is set to ignore. Note: If attachments are archived, the attachment files are passed as a reference in the message headers irrespective of the settings for the Pass By Reference parameter.
IMAP Move Folder	This is the destination of the messages if the <code>Post Read Action</code> is set to move. You must configure this field only if <code>Post Read Action</code> is set to move.
Download Directory	This is a mandatory parameter. It specifies the file system directory path to download the message.
Archive Directory	This is a mandatory parameter. A file URI that points to the directory where the files are archived. This field is active only when <code>Post Read Action</code> parameter is set to archive.
Error Directory	This is an optional parameter. This parameter specifies the type of encoding to read the request message. The default encoding is <code>iso-8859-1</code> .

For more information on how to configure e-mail services, see [Section 2.3, "Working with Proxy Services."](#)

26.2.2 Configuring Business Services Using the E-mail Transport

When you configure a business service using the e-mail transport, you can specify one or more endpoint URIs in the following formats, which lets you send e-mail messages to multiple recipients in multiple domains:

- `mailto:name@domain_name.com`
- `mailto:name@domain_name.com?smtp=smtp_server_resource`
- `mailto:name@domain_name.com?mailsession=jndi_mail_session`

For example:

- `mailto:user1@example1.com`
- `mailto:user2@example2.com?smtp=exampleSMTP`
- `mailto:user3@example3.com?mailsession=my.mail.Session`

[Table 26–5](#) describes the parameters you can configure for an e-mail transport based proxy service.

Table 26–5 Parameters for Configuring E-mail Transport for Business Services

Parameter	Description
SMTP Server	Select the default SMTP Server to use for endpoint URI entries of <code>name@domain_name.com</code> . If you provide SMTP server parameters in the endpoint URI, those server resources are used instead of this SMTP Server setting. Do not select an SMTP server if you use the Mail Session option. You must first create the SMTP Server resource. For more information, see Section 2.1.18, "Creating SMTP Server Resources."

Table 26–5 (Cont.) Parameters for Configuring E-mail Transport for Business Services

Parameter	Description
Mail Session	Enter the JNDI name of the configured mail session to use for endpoint URI entries of name@domain_name.com. If you provide JNDI mail session parameters in the endpoint URI, those mail sessions are used instead of this Mail Session setting. Do not enter a Mail Session if you use the SMTP Server option.
From Name	You must first configure mail sessions in the Oracle WebLogic Server Console.
From Address	Create a Mail Session in Oracle WebLogic Server Administration Console. You must set the Mail Session parameter or the SMTP Server parameter.
Reply To Name	This is an optional parameter. This parameter specifies the name from which the reply should be sent.
Reply To Address	This is an optional parameter. This parameter specifies the e-mail address from which the e-mail message should be sent.
Connection Timeout	This is an optional parameter. You can use this parameter to specify time in milliseconds after which the connection to the SMTP server times out.
Request Encoding	This is an optional parameter. This parameter specifies the type of encoding to read the request message. The default encoding is iso-8859-1.

For more information on how to configure business services, see [Section 4.2, "Business Service Configuration."](#)

26.3 File Transport

You can select the File transport protocol when you configure a Messaging Type or Any XML Service type of proxy service and the endpoint URI is of the form:

file:///<root-dir/dir1>

where root-dir/dir1 is the absolute path to the destination directory.

Note: File transport supports one-way messaging only for services of Messaging Service type.

When you create a messaging type proxy service or a messaging type business service using file transport you must set the response type to none in the Message Type configuration page.

26.3.1 Configuring Proxy Services using the File Transport

[Table 26–6](#) describes the parameters you can specify to configure the file transport for a proxy service.

Table 26–6 Parameters for Configuring File Transport for Proxy Services

Parameter	Description
File Mask	Specify the files that the proxy service should poll. If the URI is a directory and you specify * . *, the service polls for all the files in the directory. Only the wildcard characters * and ? are allowed in the File Mask. Regular expressions are not supported.

Table 26–6 (Cont.) Parameters for Configuring File Transport for Proxy Services

Parameter	Description
Polling Interval	This is a mandatory parameter. This specifies the value for the polling interval in milliseconds. The default value is 60 ms.
Read Limit	This is a mandatory parameter. This specifies the number of files to be read in each poll. The default value is 10. If 0 is specified, all the files are read.
Sort By Arrival	This is an optional parameter. This parameter indicates the sequence of events raised in the order of the arrival of files. The default value for this parameter is <code>False</code> .
Scan Subdirectories	This is optional. If enabled, the sub-directories are also scanned.
Pass By Reference	If this parameter is enabled, the file is staged in the archive directory and passed as a reference in the headers.
Post Read Action	This parameter is mandatory. This specifies whether the files should be deleted or archived after being read by the service. By default the files are to be deleted after reading.
Stage Directory	This is a mandatory parameter. This file URI points to the staging directory. Note: You must not put the stage directory inside the polling directory.
Archive Directory	This is a mandatory parameter. This file URI points to the directory where the files are archived. This field is active only when <code>Post Read Action</code> parameter is set to archive. Note: You must not put the archive directory inside the polling directory
Error Directory	This is a mandatory parameter. This URI points to a directory, in which the contents of the file will be stored in case of a error. Note: You must not put the error directory inside the polling directory
Request Encoding	This is an optional parameter. This parameter specifies the type of encoding to read the request message. The default encoding is <code>utf-8</code> .

For more information on how to configure file transport based proxy services, see [Section 2.3, "Working with Proxy Services."](#)

26.3.2 Configuring Business Services using the File Transport

When you configure a business service using the file transport you must specify the endpoint URI in the following format:

```
file:///<root-dir/dir1>
```

where `root-dir/dir1` is the absolute path to the destination directory.

[Table 26–7](#) describes the parameters you can specify to configure the file transport for a proxy service.

Table 26–7 Parameters for Configuring File Transport for Business Services

Parameter	Description
Prefix	This is an optional parameter. This parameter specifies the prefix to be attached to the filename.

Table 26–7 (Cont.) Parameters for Configuring File Transport for Business Services

Parameter	Description
Suffix	This is an optional parameter. This parameter specifies the suffix to be attached to the filename.
Request Encoding	This is an optional parameter. This specifies the type of encoding to read the message. The default encoding which will be used is utf-8.

For more information on how to configure this transport, see [Section 2.2, "Working with Business Services."](#)

26.4 FTP Transport

You can select the FTP transport protocol when you configure a Messaging Type or Any XML Service type of proxy service and the endpoint URI is of the form:

```
ftp://<hostname:port/directory>
```

where

- `hostname` – The name of the host on which the destination directory is stored.
- `port` – The port number at which the FTP connection is made.
- `directory` – The destination directory.

The directory is relative to the FTP session's working directory. For example, if you want the FTP service to copy a file to `home/my_ftp/documents`, and the working directory is `home/my_ftp/`, the URL would be `ftp://example:21/documents`.

Note: File transport supports one-way messaging for services of Messaging Services type.

When you create a messaging type proxy service or a messaging type business service using FTP transport you must set the response type to none in the Message Type configuration page.

26.4.1 Configuring Proxy Services using the FTP Transport

[Table 26–8](#) describes the parameters you can specify the parameters to configure the FTP transport for a proxy service.

Table 26–8 Parameters for Configuring FTP Transport for Business Services

Parameter	Description
User Authentication	You must select one of the following types of user authentication: <ul style="list-style-type: none"> ■ <code>anonymous</code>—If you select anonymous, you do not require any login credentials to login to the FTP server, but you optionally supply your e-mail ID for identification. ■ <code>external user</code>—If you select external user, you have to reference a Service Account resource, which contains your user name/password for the FTP server.
Pass By Reference	This is an optional parameter. If this parameter is enabled, the file is staged in the archive directory and passed as a reference in the headers.

Table 26–8 (Cont.) Parameters for Configuring FTP Transport for Business Services

Parameter	Description
Remote Streaming	This is an optional parameter. Setting this parameter to <code>True</code> will poll FTP files directly from the remote server at processing time.
File Mask	This is a mandatory parameter. This specifies the files that should be polled by the proxy service. If the URI is a directory and <code>*.*</code> is specified, then the service will poll all the files in the directory.
Polling Interval	This is a mandatory parameter. This specifies the value for the polling interval in milliseconds. The default value is 60 ms.
Read Limit	This is a mandatory parameter. This specifies the number of files to be read in each poll. The default value is 10.
Post Read Actions	This is a mandatory parameter. This specifies whether the files should be deleted or archived after being read by the service. By default, the files are deleted after reading.
Transfer Mode	This parameter specifies whether the mode of file transfer is binary or ASCII. By default the transfer is binary.
Archive Directory	This is a mandatory parameter. This file URI points to the directory where the files are archived. This field is active only when <code>Post Read Action</code> parameter is set to archive.
Download Directory	This is a mandatory parameter. Enter the directory on your local machine where files are downloaded during the file transfer. Note: The Archive, Download, and Error directories are absolute paths, and they are automatically created. If you specify a relative path, the files are created relative to the Java process that starts the Oracle WebLogic Server.
Error Directory	This is a mandatory parameter. This URI points to a directory location, where the contents of the file will be stored in case of a error.
Request Encoding	This is an optional parameter. This parameter specifies the type of encoding to read the request message. The default encoding is <code>utf-8</code> .
Advanced Settings	Click the icon to expand the <code>Advanced Settings</code> section. Configuring parameters in this section is optional. The parameters you can set in this section are: <ul style="list-style-type: none"> ■ Scan Subdirectories—This is optional. If enabled the sub-directories are also scanned. ■ Sort By Arrival ■ Timeout ■ Retry

For more information on how to configure file transport based proxy services, see [Section 2.2, "Working with Business Services."](#)

26.4.2 Configuring Business Services using the FTP Transport

You can select the FTP transport protocol when you configure a Messaging Type or Any XML Service type of business service and the endpoint URI is of the form:

```
ftp://<hostname:port/directory>
```

where

- `hostname` – The name of the host on which the destination directory is stored.

- `port` – The port number at which the FTP connection is made.
- `directory` – The destination directory.

The directory is relative to the FTP session's working directory. For example, if you want the FTP service to copy a file to `home/my_ftp/documents`, and the working directory is `home/my_ftp/`, the URL would be `ftp://example:21/documents`.

[Table 26–9](#) describes the parameters you must specify to configure the FTP transport for a business service.

Table 26–9 Parameters for Configuring FTP Transport for Business Service

Parameter	Description
User Authentication	You must select one of the following types of User Authentication: <ul style="list-style-type: none"> ■ anonymous—If you select anonymous, you do not require any login credentials to login to the FTP server. But you optionally supply your e-mail ID for identification. ■ external user—If you select external user, you have to reference a Service Account resource, which contains your user name/password for the FTP server.
Prefix for destination filename	This is a mandatory parameter. This parameter specifies the prefix to be attached to the filename.
Suffix for destination filename	This is a mandatory parameter. This parameter specifies the suffix to be attached to the filename.
Request Encoding	This is an optional parameter. This parameter specifies the encoding for the request message.

For more information on how to configure this transport, see [Section 2.2, "Working with Business Services."](#)

26.5 SFTP Transport

The SFTP transport is a poll-based transport that allows you to transfer files securely over the SSH File Transfer Protocol (SFTP) using SSH version 2. It polls a specified directory at regular intervals based on a predefined polling interval. After authentication, a connection is established between Oracle Service Bus services and the SFTP server, enabling file transfer. The SFTP transport supports one-way inbound and outbound connectivity.

The following are the key features of SFTP transport:

- The SFTP transport is available for the following service types:
 - Messaging service (with request message type specified)
 - Any XML service

For more information about configuring service types, see [Section 2.2, "Working with Business Services"](#) and [Section 2.3, "Working with Proxy Services."](#)

- The SFTP transport supports processing of large messages. When you configure a proxy service, you can enable content streaming and specify whether large messages must be buffered in memory or in a disk file. For more information, see "Streaming Body Content" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

- For inbound message transfer, the QoS is set to **exactly-once**, which ensures that every message is processed at least once. For outbound message processing, the QoS is **best-effort**.

Note: For messages that are not transferred, you must create the error-handling logic (including any retry logic) in the pipeline error handler.

For more information about QoS in Oracle Service Bus messaging, see "Modeling Message Flow in Oracle Service Bus" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

26.5.1 Environment Values

Environment values are predefined fields in the configuration data and are likely to change when you move the configuration from one domain to another (for example, from test to production). The following table lists the environment values associated with the SFTP transport.

Table 26–10 Environment Values

Environment Value	Description
Archive Directory	The directory to which the files are moved from either the download directory or the remote location.
Download Directory	The directory on your local machine to which files are downloaded during the file transfer.
Error Directory	The location where messages are posted if there is a problem.
Managed Server for Polling	The managed server that is used for polling (in a cluster scenario).

For more information, see:

- "Customization" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*
- [Section 26.5.4.2, "Configuring Proxy Services"](#)
- [Section 26.5.4.3, "Configuring Business Services"](#)

26.5.2 General Principles of SFTP Authentication

The following principles are applicable to the SFTP authentication process for both proxy and business services:

- **Connection:** The Oracle Service Bus service (proxy and business) always acts as the SFTP client and connects to the SFTP server.
- **Authentication by the SFTP server**
 - For public key and host-based authentication, the SFTP server authenticates the connection with the public key of the Oracle Service Bus service.
 - For username-password authentication, the SFTP server authenticates the connection with the username and password.
- **Authentication by the SFTP client:** The Oracle Service Bus service always authenticates the SFTP server with the public-key/host/IP combination defined in

the **known_hosts** file. For more information, see [Section 26.5.4.1.1, "Creating the Known Hosts File."](#)

- Connection establishment: The connection is established only when both the server and client authentications are successful.
- Transfer
 - If the client is a proxy service, the file (message) is downloaded from the SFTP server.
 - If the client is a business service, the file (message) is uploaded to the SFTP server.

26.5.3 Run-Time Behavior

Transferring files by using the SFTP transport involves the following steps:

1. The proxy service polls the input directory at regular intervals.

Note: A new connection is created for each poll cycle.

2. If the proxy service finds a file in the input directory, it renames the file with a **.stage** extension. This renaming ensures that the service does not pick up the same files during the next polling cycle.

The **.stage** file exists in the input directory until it is delivered.

Note: If the file cannot be retrieved from the input directory (due to network failure, for example), the **.stage** file is processed during a clean-up cycle. The clean-up cycle is performed every 15 minutes or after 15 polling cycles, whichever occurs later. If a **.stage** file exists during two consecutive clean-up cycles, it is processed again.

3. A JMS task is created for the message and added to the domain-wide JMS queue.
4. A domain-wide MDB receives the task and processes the message.

Note: The task uses a pooled connection for processing the message. If a connection is not available in the pool, a new connection is created.

5. The message is delivered to the pipeline and the **.stage** file is deleted.

Note: If an SFTP business service is configured, the service puts the message in the outbound directory through a pooled connection.

If the message is not delivered, the transport attempts to transfer it again and repeats the process up to a predefined number of attempts. If the message cannot be delivered, it is moved to the error directory.

26.5.4 Using the SFTP Transport

You can use the SFTP transport to transfer files securely using SSH File Transfer Protocol (SFTP).

The following sections describe how you can use the SFTP transport to transfer files securely:

- [Section 26.5.4.1, "Enabling SFTP Authentication"](#): This section describes the authentication methods that the SFTP transport supports.
- [Section 26.5.4.2, "Configuring Proxy Services"](#) and [Section 26.5.4.3, "Configuring Business Services"](#): These sections describe how you can configure proxy and business services to use the SFTP transport.
- [Section 26.5.4.4, "Handling Communication Errors"](#) and [Section 26.5.4.5, "Troubleshooting"](#): These sections provide information to help you solve problems that may occur while configuring or using the SFTP transport.
- [Section 26.5.4.6, "Importing Resources"](#): This section describes the SFTP services-specific policy and configuration details that you can preserve when you import resources.
- [Section 26.5.4.7, "Importing and Publishing Services: UDDI Registries"](#): This section lists the properties that are published when SFTP services are published to UDDI registries. It also lists the properties that are imported when SFTP services are imported from UDDI registries.

26.5.4.1 Enabling SFTP Authentication

The SFTP transport supports the following authentication methods:

- Username-password authentication
- Host-based authentication
- Public key authentication

Oracle Service Bus services authenticate the SFTP server based on the server details defined in a **known_hosts** file. So to enable server authentication, you must create a **known-hosts** file on the client machine.

26.5.4.1.1 Creating the Known Hosts File The **known_hosts** file must exist in the server on which the Oracle Service Bus proxy services (inbound requests) or business services (outbound requests) run. The file must contain the host name, IP address, and public key of the remote SFTP servers to which the proxy service or business service can connect.

1. Create a **known_hosts** file and enter details in the following format:

Hostname,IP algorithm publickey, where:

- *Hostname* is the host name of the SFTP server.
- *IP* is the IP address of the SFTP server.

If you use an IPv6 address, do not use a double colon to represent multiple zeros. Write out all zeros. For example, use this format "0:0:0:0" instead of this format "::".

- *algorithm* can be either DSA or RSA, based on the SFTP server configuration. Enter **ssh-rsa** or **ssh-dss** depending on the algorithm that is supported.

- `publickey` is the public key of the SFTP server. It must be in the Open SSH public key format.

Example

```
getafix,172.22.52.130 ssh-rsa
```

```
AAAAB3NzaC1yc2EAAAABIwAAAIEAtR+M3Z9HFxnKZTx66fZdnQqAHQcF1vQe1+EjJ/HWYtg
```

```
Anqsn0hMJzqWMatb/u9yFwUpZBirjm3g2I9Qd8VocmeHwoGPhDGFQ5LQ/PPo3esE+CGwdnC
```

```
OyRCktNHeuKxo4kiCCJ/bph5dRpghCQIvsQvRE3sks+XwQ7Wuswz8pv58=
```

The **known_hosts** file can contain multiple entries, but each entry must be on a separate line.

2. Move the **known_hosts** file to the `DOMAIN_HOME/config/osb/transports/sftp` directory.

The directories `/transports/sftp` are not created automatically. You must create the directories.

26.5.4.1.2 Enabling Username-Password Authentication Username-password authentication is the simplest and quickest method of authentication. It is based on the credentials of the user.

To enable username and password authentication for a service:

1. Create a static service account by using the user credentials on the SFTP server. For more information, see [Section 2.1.16, "Creating Service Account Resources."](#)
2. Create a **known_hosts** file. For more information, see [Section 26.5.4.1.1, "Creating the Known Hosts File."](#)

26.5.4.1.3 Enabling Host-Based Authentication Host-based authentication uses a private host key. This method can be used when all the users share a private host.

To enable host-based authentication for a service:

1. Configure a service key provider with an SSL client authentication key. For more information, see "Service Key Providers" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

Note: You can extract the public key from the key store that was used while creating the service key provider. The public key must be in the Open SSH format.

2. Create a **known_hosts** file. For more information, see [Section 26.5.4.1.1, "Creating the Known Hosts File."](#)
3. Configure the SFTP server to accept requests from Oracle Service Bus, which is a client to the SFTP server.

For example, for an SFTP server on Linux, you must do the following:

- Edit the `/etc/ssh/shosts.equiv` file and add the host name or IP address of the machine on which the Oracle Service Bus domain runs.
- Edit the `/etc/ssh/ssh_known_hosts` file and add the host name or IP address of the machine on which the Oracle Service Bus domain runs, followed by a space and the public key.

Note: For host-based authentication, Open SSH compares the host name provided by the client against a reverse lookup on the client IP address. Because of scenarios where the request comes from a multi-homed machine or through NAT gateway, host names may not match, resulting in authentication failure. To work around such scenarios, turn off the DNS check by setting the `HostbasedUsesNameFromPacketOnly` option to "yes" in `/etc/ssh/sshd_config`. This workaround does not subvert security, because the real security is in the public key matching between the `known_hosts` file and the SFTP server.

26.5.4.1.4 Enabling Public Key Authentication Public key authentication is performed using your own private key. This method can be used when each user has a private key.

To enable public key authentication:

1. Configure a service key provider with SSL client authentication key. For more information, see "Service Key Providers" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.
2. Configure the SFTP server to accept requests from Oracle Service Bus (SFTP client).
For example, for an SFTP server on Linux, you must extract the public key from the key store and enter the key in the `$HOME/.ssh/authorized_keys` file on the SFTP server. Ensure that the path and file exist.
3. Create a `known_hosts` file. For more information, see [Section 26.5.4.1.1, "Creating the Known Hosts File."](#)

26.5.4.2 Configuring Proxy Services

When you create a proxy service in the **Transport Configuration** page of the Oracle Service Bus Console, you must select the transport protocol as `sftp` and specify the endpoint configuration in the following format:

```
sftp://hostname:port/directory
```

Following is a description of each part of the endpoint URI:

- `hostname` is the host name or IP address of the SFTP server.
- `port` is the port on which SFTP server is listening. The default port for SFTP is 22.
- `directory` is the location that is polled for files at regular intervals.

The directory is relative to the SFTP session's working directory. For example, if you want the SFTP service to copy a file to `home/my_sftp/documents`, and the working directory is `home/my_sftp/`, the URL would be `sftp://example:21/documents`.

Note: Since the SFTP transport supports only message and XML service types, you must select **Messaging Service** or **Any XML Service** as the service type in the **General Configuration** page of the Oracle Service Bus Console.

When you select **Messaging Service** as the service type,

- You must specify **Binary**, **Text**, **MFL**, or **XML** as the request message type.
- You must set the response message type to **None** because the SFTP transport supports only one-way messaging.

For more information, see [Section 2.3, "Working with Proxy Services."](#)

Configure the proxy service as described in the following table.

Table 26–11 *Configuring SFTP Proxy Service*

Field	Description
User Authentication	<p>The proxy service is authenticated by the SFTP server based on the specified user authentication method.</p> <p>Select the required authentication method.</p> <ul style="list-style-type: none"> ■ Username Password Authentication: Specifies that a static service account is associated with this authentication method and the client is authenticated using the credentials provided in the service account. ■ Host-Based Authentication: Specifies that a user name and service key provider are required. Any user connecting from a known host is authenticated using the private key of the host. ■ Public Key Authentication: Specifies that a user name and service key provider are required. Users have their own private keys.
Service Account	<p>Enter the service account for the user, or click Browse and select a service account. For information about using service accounts, see Section 2.1.16, "Creating Service Account Resources."</p>
Service Key Provider	<p>This field is available only for the public key and host-based authentication methods.</p> <p>Enter a service key provider, or click Browse and select a service key provider. For more information, see "Service Key Providers" in the <i>Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus</i>.</p>
Username	<p>This value is required only when you select either the host-based or public key authentication method.</p> <ul style="list-style-type: none"> ■ In host-based authentication, the user name is required for polling the home directory of the user on the SFTP server. ■ In public key authentication, the user name is required for polling the home directory of the user and for identifying the location of the public key on the SFTP server. <p>Enter the user name.</p>
Pass By Reference	<p>Select this option to stage the file in the archive directory and pass it as a reference in the headers.</p> <p>Note: This option is available only when remote streaming is disabled.</p>

Table 26–11 (Cont.) Configuring SFTP Proxy Service

Field	Description
Remote Streaming	Select this option to stream the SFTP files directly from the remote server at the time of processing. When you select this option, the archive directory is the remote directory on the SFTP server machine. Therefore, you must specify the archive directory relative to the SFTP user directory.
File Mask	You can use the file mask for transferring files of specific types. Enter a regular expression to select the files that you want to pick from the directory. The default value is <code>*.*</code> .
Polling Interval	Polling interval is the frequency at which the input directory is polled. Polling involves creation of an SFTP connection. Enter the interval (in seconds) at which the file must be polled from the specified location. The default value is 60. Note: Avoid setting a low polling interval because a low interval causes frequent polls on the directory.
Read Limit	If numerous files exist in the poll directory, you can limit the number of concurrent transfers by selecting an appropriate value in this field. If you do not want to specify a limit, enter 0 (zero). The default value is 10 . Note: In some cases, the SFTP server may limit the number of concurrent connections; make sure that the read limit you define is lower than the server-defined limit.
Post Read Action	Select the action that must be performed on the message after the file is transferred. <ul style="list-style-type: none"> ■ Archive: The message is archived in the specified archived directory. ■ Delete: The message is deleted.
Archive Directory	If Post Read Action is set to Archive , then, after the files are transferred, they are moved (from either the download directory or the remote location) to the archive directory. If you selected the Pass By Reference option, you must specify the archive directory. If remote streaming is enabled, the archive directory is with respect to the SFTP server. If remote streaming is disabled, the archive directory is available on the Oracle Service Bus machine. Specify the absolute path of the archive directory. Note: If the directory does not exist, it is created automatically. If you specify a relative path, the directory is created at a path that is relative to the Java process that starts the Oracle WebLogic Server.
Download Directory	During file transfer, the files are downloaded to this directory. If remote streaming is enabled, this option is disabled. Specify the absolute path of the directory on your local machine to which files are downloaded during the file transfer. Note: If the directory does not exist, it is created automatically. If you specify a relative path, the directory is created at a path that is relative to the Java process that starts the Oracle WebLogic Server.

Table 26–11 (Cont.) Configuring SFTP Proxy Service

Field	Description
Error Directory	<p>If a problem occurs during file transfer, the messages are posted to the error directory.</p> <p>If remote streaming is enabled, the error directory is with respect to the SFTP server. If remote streaming is disabled, the error directory is available on the Oracle Service Bus machine.</p> <p>Specify the absolute path of the error directory.</p> <p>Note: If the directory does not exist, it is created automatically. If you specify a relative path, the directory is created at a path that is relative to the Java process that starts the Oracle WebLogic Server.</p>
Request encoding	Accept the default value (UTF-8) as the character set encoding for requests in the SFTP transports.
Scan Subdirectories	<p>Select this option if you want all subdirectories within the directory that is specified in the endpoint URI to be scanned recursively.</p> <p>Note: Scanning subdirectories requires additional processing overhead; so use this option judiciously.</p>
Sort By Arrival	Select this option to deliver events in the order of arrival. This ensures that message delivery is not random, but based on the time at which the file is downloaded to the destination directory.
Timeout	Enter the socket timeout interval, in seconds, after which the connection must be dropped. If you do not want the connection to time out, enter 0 . The default value is 60.
Retry Count	<p>You can use this setting to enable multiple attempts in case of errors such as network failure.</p> <p>Specify the number of retries for SFTP connection failures. The default value is 3.</p>

For more information about configuring proxy services to use the SFTP transport, see [Section 2.3, "Working with Proxy Services."](#)

26.5.4.2.1 Configuring Transport Headers and Metadata When you configure a proxy service, you can use a Transport Header action to set the header values in messages. The following table lists the transport header and metadata related to the SFTP transport.

Table 26–12 Transport Headers and Metadata

Header / Metadata	Description
FileName	This value is used as the file name in the destination directory.
isFilePath	<p>This is a metadata field. The possible values are true and false.</p> <ul style="list-style-type: none"> ■ True: FileName is interpreted as the absolute path of the file. ■ False: FileName is interpreted as the name of the file.
filePath	This is a response metadata field that indicates the absolute path at which the file specified in the FileName header is written.

Configuring Transport Headers in the Oracle Service Bus Message Flow

You can configure the transport headers only for outbound requests in the Oracle Service Bus message flow. In the pipeline, use a transport header action to set the

header values in messages. For more information, see [Section 2.4.32, "Adding and Configuring Transport Headers Actions in Message Flows."](#)

Configuring Transports Headers and Metadata in the Test Console

You can configure the **FileName** transport header and the **isFilePath** metadata values in the Oracle Service Bus test console when you test the SFTP transport-based services during development. For more information, see "Using the Test Console" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

26.5.4.3 Configuring Business Services

When you create a business service in the **Transport Configuration** page of the Oracle Service Bus Console, you must select the transport protocol as **sftp** and specify the endpoint URI (location of the service) in the following format:

```
sftp://hostname:port/directory
```

Following is a description of each part of the endpoint URI:

- *hostname* is the host name or IP address of the SFTP server.
- *port* is the port on which SFTP server is listening. The default port for SFTP is 22.
- *directory* is the location in which the outbound message is stored or written.

The directory is relative to the SFTP session's working directory. For example, if you want the SFTP service to copy a file to `home/my_sftp/documents`, and the working directory is `home/my_sftp/`, the URL would be `sftp://example:21/documents`.

Note: Since the SFTP transport supports only message and XML service types, you must select **Messaging Service** or **Any XML Service** as the service type in the **General Configuration** page of the Oracle Service Bus Console.

When you select **Messaging Service** as the service type,

- You must specify **Binary**, **Text**, **MFL**, or **XML** as the request message type.
- You must set the response message type to **None** because the SFTP transport supports only one-way messaging.

For more information, see [Section 2.2, "Working with Business Services."](#)

Configure the business service as described in the following table.

Table 26–13 Configuring SFTP Business Service

Field	Description
User Authentication	<p>The proxy service is authenticated by the SFTP server based on the specified user authentication method.</p> <p>Select the required authentication method.</p> <ul style="list-style-type: none"> ■ Username Password Authentication: Specifies that a static service account is associated with this authentication method and the client is authenticated using the credentials provided in the service account. ■ Host-Based Authentication: Specifies that a user name and service key provider are required. Any user connecting from a known host is authenticated using the private key of the host. ■ Public Key Authentication: Specifies that a user name and service key provider are required. Users have their own private keys.
Service Account	<p>Enter the service account for the user, or click Browse and select a service account. For information about using service accounts, see Section 2.1.16, "Creating Service Account Resources."</p>
Service Key Provider	<p>This field is available only for the public key and host-based authentication methods.</p> <p>Enter a service key provider, or click Browse and select a service key provider. For more information, see "Service Key Providers" in the <i>Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus</i>.</p>
Username	<p>This value is required only when you select either the host-based or public key authentication method.</p> <ul style="list-style-type: none"> ■ In host-based authentication, the user name is required for polling the home directory of the user on the SFTP server. ■ In public key authentication, the user name is required for polling the home directory of the user and for identifying the location of the public key on the SFTP server. <p>Enter the user name.</p>
Timeout	<p>Enter the socket timeout interval, in seconds, after which the connection must be dropped. If you do not want the connection to time out, enter 0. The default value is 60.</p>
Prefix for destination File Name	<p>Enter the prefix for the name of the file that is stored on the remote server.</p>
Suffix for destination File Name	<p>Enter the suffix for the name of the file that is stored on the remote server.</p>
Request encoding	<p>Accept the default value (UTF-8) as the character set encoding for requests in the SFTP transports.</p>

For more information about configuring business services using the SFTP transport, see [Section 2.2, "Working with Business Services."](#)

26.5.4.4 Handling Communication Errors

You can configure the SFTP transport-based business services to handle communications errors, which can occur when a connection or user authentication fails while connecting to the remote SFTP server. When you configure the business service, you can enable the business service endpoint URIs to be taken offline after a specified retry interval.

For more information, see the following topics in "Monitoring" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*:

- "Configuring Operational Settings for Business Services"
- "Viewing Business Services Endpoint URIs Metrics"

26.5.4.5 Troubleshooting

Most of the errors occur while configuring an SFTP proxy or business service. The following are a few tips to help you understand and solve the errors:

- Make sure that you have an appropriately configured **known_hosts** file in place.
- For public key authentication, you must store the public key file on the server. For more information, see the documentation accompanying your SFTP server.
- The **Connection refused** error message indicates that the SFTP server is not available on the configured host and port.
- The **Authentication failed** error message indicates that the username or password is not valid, or that the public key is not configured correctly.
- The **Connection did not complete** error message is displayed after the actual error that caused the connection failure (example: **Key not found**) is displayed.
- The **Key not found for IP, host** error message indicates that the **known_hosts** file does not contain an entry that corresponds to the specified IP-host combination. If the entry exists, then try with another algorithm key; for example, if the earlier attempt was with an RSA key, try again with a DSA key.

26.5.4.6 Importing Resources

When you import a resource that exists in an Oracle Service Bus domain, you can preserve the existing security and policy configuration details of the resource by selecting the **Preserve Security and Policy Configuration** option. The following SFTP service-specific details are preserved when you import a resource:

- Client authentication method
- Reference to the service account (for services associated with username-password authentication)
- Reference to the service key provider (for services associated with host-based or public key authentication)
- User name (for services associated with host-based or public key authentication)

For more information about importing resources from the Oracle Service Bus Console, see [Section 2.1.14, "Importing Resources."](#)

26.5.4.7 Importing and Publishing Services: UDDI Registries

When an SFTP service is published to the UDDI registry, `Authentication mode`, `Request encoding`, `Sort by arrival`, and `File mask` are the properties that are published.

[Table 26–14](#) lists the properties that are imported from the registry when an SFTP service is imported from the UDDI registry.

Table 26–14 Properties Imported from UDDI Registry

Property	Description
Prefix for destination file name	The prefix for the name of the file that is stored on the remote server. The default value is " " (null).
Suffix for destination file name	The suffix for the name of the file that is stored on the remote server. The default value is " " (null).
Authentication mode	The authentication method that is imported from the registry. When an SFTP business service with user authentication is imported from an UDDI registry to Oracle Service Bus, a conflict is generated. <ul style="list-style-type: none"> ■ For username-password authentication, you must create a service account and associate it with the service. ■ For host-based or public key authentication, you must create a service key provider and associate it with the service.

After the service is import imported, the default value of the load balancing algorithm is **round-robin**.

For more information, see "UDDI" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

SB Transport

The SB transport allows Oracle products to synchronously invoke an Oracle Service Bus proxy service using RMI. The inbound transport allows clients to access SB proxy services using RMI. The outbound transport allows the invocation of SB proxy services using RMI. By default, accessing all services using T3 protocol, IIOP, HTTP, T3s, IIOPS, or HTTPS depends on the configuration of the target server. For more information, see "Configure Default Network Connections" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*.

The SB transport supports:

- Propagation of the transaction context. The transaction originated in the client Oracle Service Bus servers can optionally be propagated to the SB proxy service.

Propagation of the security context. By default, the security context associated with the SB client thread is used to invoke the SB proxy services. This may require enabling domain trust between domains. See "Important Information Regarding Cross-Domain Security Support" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.
- Invocation of SB proxy services, with custom identities, by the outbound endpoint using a service account.
- Specification of time out value for non-transactional invocations. The client request returns when Oracle Service Bus does not respond to the request within the specified interval.
- Association of a dispatch-policy for both request and response connections. For more information, see "Using Work Managers to Optimize Scheduled Work" in *Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server*.
- Optimization of RMI call and call-by-reference when routing to a SB business service without a JNDI provider.
- The following service types:
 - WSDL service
 - Any SOAP service
 - Any XML service
- The following messaging patterns:
 - Request (one-way) and request-response for the inbound transport.

For an Oracle Service Bus client the by default the messaging pattern is inherited from the pipeline of the SB outbound transport.

For a non-Oracle Service Bus client by default messaging pattern is request-response.

- Request and request-response for the outbound transport Environment Values. For more information on the environment values the SB supports, see [Section 27.1, "Environmental Values."](#)
- The following default values for the Quality of Service (QoS):
 - Exactly-Once for non-Oracle Service Bus clients
 - Best-Effort for Oracle Service Bus clients

You can also set the QoS of a service using routing options in the message flow. For more information, see "Quality of Service" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

This document provides information about:

- [Section 27.1, "Environmental Values"](#)
- [Section 27.2, "Configuring Proxy Services to Use the SB Transport"](#)
- [Section 27.3, "Configuring Business Services to Use the SB Transport"](#)
- [Section 27.4, "Handling Errors"](#)
- [Section 27.5, "UDDI"](#)

27.1 Environmental Values

[Table 27–1](#) describes the environment values the SB Transport supports.

Table 27–1 Environment Values

Environment Value	Use this value to ...
Timeout (category: operational)	Override the time out value set for the business service.
Service account (category: security)	Update the user credentials associated with the business service.
Use SSL (category: security)	Enable or disable the option of exposing a service using a secure protocol for a proxy service.

27.2 Configuring Proxy Services to Use the SB Transport

A client Oracle Service Bus server connects with the Oracle Service Bus server using the JNDI context and the proxy service URI. The security context of the client is used to invoke the proxy service. The default QoS is `Exactly-once`. Optionally, the client can change the QoS, set a request time out value and specify a desired messaging pattern. The message is received by the inbound SB transport and processed through the message flow.

To create a proxy service from the Oracle Service Bus Console:

1. Enter the general configuration details for the service in the General Configuration page. For more information, see [Section 4.3.1, "Proxy Service General Configuration Page."](#)
2. In the Transport Configuration page, select the transport protocol as `sb`. Specify the endpoint URI as `<proxy service name>`. This proxy service name is the unique identifier associated with the proxy service that is to be invoked by the

client. For more information, see [Section 4.3.6, "Proxy Service Transport Configuration Page."](#)

3. In the SB Transport Configuration page, specify the values as described in [Table 27-2](#).

Table 27-2 Fields for Configuring SB Transport for Proxy Services

Field	Description
Dispatch Policy	You can use the default dispatch policy or configure a work manager for the inbound request. Dispatch policy refers to the instance of WLS Work Manager that you want to use for the service endpoint. For more information, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> .
Use SSL	To expose the service through a secure protocol, choose the Use SSL option. Although this implies that the client should use the SSL protocol to access the SB proxy service, this does not prevent the client from accessing the service through unsecure protocols. In addition, the endpoint URI associated with the service would be sbs instead of sb: <ul style="list-style-type: none"> ▪ When you export a secure service through UDDI and preserve security and policy configuration details during import. ▪ In the effective WSDL. <p>Note: A proxy service is not bound to any particular protocol. It is the responsibility of the WLS administrator to enable SSL, IIOP, or HTTP tunneling whenever it is necessary.</p> <p>This flag only affects the URI scheme of the service when it is exported or the JNDI provider selection for the business service URI when it is imported from UDDI. It does not prevent a client from accessing the service using a non-secured protocol.</p>

For more information, see [Section 2.3, "Working with Proxy Services."](#)

27.3 Configuring Business Services to Use the SB Transport

The SB business service can send messages only to other SB proxy services. A JNDI provider, which is specified in the endpoint URI of the business service, performs a JNDI lookup on the remote Oracle Service Bus server. The client user credentials or the user credentials defined in the service account associated with the business service are used to invoke the proxy service. Optionally, a time out value and a custom dispatch policy can be associated with the business service. The QoS of the service can also be set by using the routing options.

To create a business service from Oracle Service Bus Console:

1. Enter the general configuration details for the service in the General Configuration page. For more information, see [Section 4.2.1, "Business Service General Configuration Page."](#)
2. In the Transport Configuration page, select the transport protocol as sb and specify the endpoint URI in the following format:

```
sb://<jndi_provider_name>/<service_name>
```

where,

- <jndi_provider_name> is the name of the JNDI provider, which points to the corresponding Oracle Service Bus JNDI provider. Optional. When omitted, the default context is used. This implies that the service and the Oracle Service

Bus server are located on the same machine. When the call is co-located, serialization is skipped during service invocation. For more information, see [Section 27.3.1, "JNDI Provider."](#)

- `<service name>` is a target service and corresponds to the remote proxy service URI.

For more information see, [Section 4.2.4, "Business Service Transport Configuration Page."](#)

3. In the SB transport configuration page, specify the values as described in [Table 27-3.](#)

Table 27-3 Fields for Configuring SB Transport for Business Services

Field	Description
Dispatch Policy	Select the instance of Oracle WebLogic Server Work Manager that you want to use as the dispatch policy for this service endpoint. The default Work Manager is used if no other Work Manager exists. For more information, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> .
Time out	The duration, in seconds, after which the business service times out and business service returns a run time error when a timeout occurs. The specified time out value is <i>not</i> applied when: <ul style="list-style-type: none"> ■ QoS of the service endpoint is <code>Exactly-Once</code>. ■ The specified value is a negative value. ■ The time out value is overridden in the optional Timeout custom header of the outbound request in the message flow. For information, see Section 2.4.32, "Adding and Configuring Transport Headers Actions in Message Flows."
Service Account	Specify the user credentials that should be used for invoking the remote proxy service. If no service account is specified, the user credentials of the inbound proxy service (the inbound client) of this business service are used for security context propagation. For more information, see Section 2.1.16, "Creating Service Account Resources."

For more information, see [Section 2.2, "Working with Business Services."](#)

27.3.1 JNDI Provider

A JNDI provider points to the Oracle Service Bus server where the service is deployed to retrieve the RMI stubs corresponding to the SB proxy service. The JNDI provider has a high performance caching mechanism for remote connections and EJB stubs. T3, IIOP, HTTP, T3s, IIOPS, or HTTPS transport protocols can be used by JNDI provider. The preferred communication protocol from Oracle Service Bus to a WLS domain is T3 or T3S. If messages need to go through a fire wall, you can use HTTP tunneling by using an HTTP provider url in the context and by enabling HTTP tunneling on the WLS server.

Note: It is the responsibility of the administrator to ensure that the protocol supported by the JNDI provider is on the remote Oracle Service Bus server.

When you create a business service, you can associate it with a JNDI provider. For more information, see [Section 27.3, "Configuring Business Services to Use the SB Transport"](#) and [Section 2.1.6, "Creating JNDI Provider Resources."](#)

27.4 Handling Errors

You can configure the SB transport business services to handle communications errors. You can configure business service URIs to be taken offline when communication errors occur. When you configure the operational settings for the business service, you can enable the business service endpoint URIs to be taken offline after the specified retry interval.

For more information, see *Configuring Operational Settings for Business Services and Viewing Business Services Endpoint URIs Metrics in "Monitoring" in the Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus.*

When a connection error occurs while invoking a SB proxy service, the SB transport provider generates the BEA-380002 error code.

A connection error can occur due to any of the following reasons:

- The target proxy service does not exist
- The JNDI provider settings are incorrect
- Any remote or naming exception occurs during RMI invocation

Note: Naming Exception of type `javax.naming.NamingSecurityException` typically occurs when the identity used during the invocation is not recognized by the target server. When this occurs, the request returns a generic runtime error, which is not treated as a connection error.

SOAP faults returned by SB Proxies are treated as application errors.

27.5 UDDI

You can import and publish services to the UDDI registry. For more information, see "UDDI" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus.*

27.5.1 Publishing a Service

When you publish a proxy service to a UDDI register, the URI associated with the published service has the following format:

```
sb://host:port/service_name
```

where, `host:port` refers to the host name and listening port of the Oracle Service Bus server hosting the proxy service that is being published.

If the `Use SSL` option is enabled for the proxy service that is being published, the URI associated with the published service has the following format:

```
sbs://host:port/service_name
```

where, `host:port` refers to the host name and the SSL listening port of the Oracle Service Bus server hosting the proxy service.

If the proxy service that is being published is running on a cluster, `host:port` is the `Cluster Address` setting in the `Cluster` section of the `config.xml` file. This value can either be a single host name and port number that is used to connect to any WLS in the cluster or it can be a comma-separated list of the host name and listener ports of the managed servers in the Oracle Service Bus cluster. For more information, see "WebLogic JNDI" in *Oracle Fusion Middleware Programming JNDI for Oracle WebLogic Server*.

For more information, see "Publishing Proxy Services to a UDDI Registry" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

27.5.2 Importing a Service

When you import a service from the UDDI registry, the SB transport provider matches the `sbscheme` and `host:port` information from the service URI property with a JNDI provider registered on the Oracle Service Bus server using the appropriate protocol based on `sbscheme`. `Sbscheme` is the URI scheme of the SB transport-based service and can be either `sb` or `sbs`.

If `sbscheme` is `sb`, the transport provider looks for the JNDI provider using T3, T3S, IIOP, IIOPS, HTTP, or HTTPS protocol (in this order). If `sbscheme` is `sbs`, the transport provider looks for the JNDI provider using T3S protocol, IIOPS, then HTTPS (in this order). The JNDI provider that matches the service URI property is used to generate the endpoint URI of the business service that is imported to Oracle Service Bus.

If there is no matching JNDI provider, the import fails unless the imported URI is a local URI and the scheme is not `sb`, the default context is used. This implies that there is no JNDI provider specified for the service and it is considered co-located with the server.

For example, if the service URI property value is:

```
sbs://remote_oracle_service_bus_host:7002/myservice
```

the generated URI of the business service imported to Oracle Service Bus would be:

```
sb://my_jndi_provider/myservice
```

where, `my_jndi_provider` is a JNDI provider resource registered on the Oracle Service Bus server with a `t3s://remote_oracle_service_bus_host:7002` URL.

For more information, see "Importing Business Services From a UDDI Registry" in *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

Using the EJB transport, Oracle Service Bus supports native RMI invocation of EJB 2.1 or EJB 3.0 stateless session beans deployed on supported platforms. It allows transactional and secure communications. You can also leverage the EJB transport to expose an EJB as a Web Service through Oracle Service Bus.

This section includes the following topics:

- [Section 28.1, "Introduction"](#)
- [Section 28.2, "Invoking EJBs from Oracle Service Bus"](#)
- [Section 28.3, "Exposing EJBs as Web Services"](#)
- [Section 28.4, "Advanced Topics"](#)
- [Section 28.5, "Troubleshooting"](#)

28.1 Introduction

In Oracle Service Bus, you can configure business services to use the EJB transport. The EJB transport is fully integrated into the Oracle Service Bus configuration, management, monitoring, and test consoles. You can use business services configured with the EJB transport for publish, service callout, and service invocations. You cannot create proxy services that use the EJB transport.

The EJB transport provides the following capabilities:

Transactional Integrity – You can call an EJB business service in the context of a global transaction. The EJB transport can also suspend or start a global transaction before invoking an EJB.

Security Propagation – The security context established at the beginning of a message flow, from an Oracle Service Bus client, is propagated to the other system. For example, an incoming SOAP over HTTP request to Oracle Service Bus that requires authentication is authenticated by Oracle Service Bus and the authenticated subject can then be propagated to the EJB server. See "Important Information Regarding Cross-Domain Security Support" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

HTTP Tunneling and Encrypted Communication – You can access EJBs that are behind a firewall with HTTP tunneling. For additional security, you can use SSL to encrypt all of the communications with the EJB Server.

JNDI Provider – EJB transport leverages the JNDI provider—an Oracle Service Bus resource. The JNDI provider defines communication protocols and security credentials for accessing remote servers. A JNDI provider can be reused by multiple EJB business

services. This provides a centralized way for administrators to manage remote EJB server configurations.

For information about JNDI provider resources, see [Section 2.1.6, "Creating JNDI Provider Resources."](#)

High Performance Caching – The EJB transport is built on high performance cache. This allows the reuse of established connections and minimizes EJB stubs lookups.

Failover and Load Balancing – The EJB transport can take advantage of scenarios in which the same EJB is deployed in multiple domains or on a cluster for load balancing or failover or both.

Advanced XML to Java Binding Capabilities – The EJB transport leverages the Oracle WebLogic Server JAX-RPC stack to perform Java to XML bindings. The JAX-RPC stack is a high performance engine that supports advanced Java objects such as XML Beans. If the Java type is not recognized by the stack, an extension mechanism is provided to facilitate support of these Java types. For information about this extension mechanism (using the converter classes), see [Section 28.4.2, "Supported Types and Converter Class."](#)

Intelligent Retries – The EJB transport makes retry decisions based on the nature of the failure that can occur during the invocation of an EJB.

28.2 Invoking EJBs from Oracle Service Bus

Before you can configure a business service in Oracle Service Bus, you must register a JNDI provider resource and a client JAR resource. This section describes how to design and configure an EJB transport business service in Oracle Service Bus.

28.2.1 Register a JNDI Provider Resource

A JNDI Provider resource allows you to specify the communication protocols and security credentials used to retrieve EJB stubs bound in the JNDI tree of remote Oracle WebLogic Server domains. (For more information on how to set up a JNDI tree, see *Oracle Fusion Middleware Programming JNDI for Oracle WebLogic Server*.)

Typically, the target EJB is not located in the same domain as Oracle Service Bus. In this case, you must register a JNDI Provider resource. When the EJB is located in the same domain, you can define a provider to specify credentials and take advantage of stubs caching, although it is optional in this case.

The JNDI provider has a high performance caching mechanism for remote connections and EJB stubs. The preferred communication protocol from Oracle Service Bus to an Oracle WebLogic Server domain is `t3` or `t3s`. If messages need to go through a firewall, you can use HTTP tunneling.

Note: Although it is possible to use an Oracle WebLogic Server foreign JNDI Provider, Oracle recommends that you do not.

The transport does not support two-way SSL or CLIENT CERT to look-up JNDI tree or access a method on an EJB.

28.2.1.1 Adding a JNDI Provider

For information about registering and configuring a JNDI provider resource in Oracle Service Bus, see [Section 4.8, "JNDI Providers."](#)

28.2.2 Register an EJB Client JAR Resource

A client JAR must be registered as a resource in Oracle Service Bus. It is therefore part of the Oracle Service Bus configuration and can be exported from and imported into a project.

An EJB client JAR file must contain the interfaces and classes needed by Oracle Service Bus to access an EJB. This includes the remote and home interfaces (EJB 2.1) or business interfaces (EJB 3.0) and any dependent types to which the client is exposed, such as method parameter types or application exceptions.

If your business service requires converter classes, you can register a JAR file containing the converter classes as an Oracle Service Bus resource and subsequently use these classes to help map parameter and return value types to Java classes that can be mapped to XML. Alternatively, you can package these converter classes in the EJB client JAR. For information about converter classes, see [Section 28.4.2.1, "Converter Classes."](#)

Consider the following guidelines when using EJB client JARs:

- Adding home and remote interfaces in the system classpath is bad practice and is not supported by Oracle Service Bus.
- Oracle recommends that you keep the client JAR size small, include a single home interface per JAR and not register the entire `ejb-jar` file.
- You can use Eclipse to obtain a client JAR for EJBs deployed on Oracle WebLogic Server.
- Oracle Service Bus supports client-jars compiled with JDK 1.4 or later.

28.2.2.1 Adding a Client or Converter JAR

For information about registering and configuring a JAR resource in Oracle Service Bus, see "Adding JARs" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

28.2.2.2 Create a Service Account (Optional)

If the EJB methods are protected, you can specify the credentials you want to use for the invocations. Those credentials are often different than the credentials used by the JNDI provider. For information about adding and using service accounts, see [Section 2.1.16, "Creating Service Account Resources."](#)

28.2.2.3 Locate an EJB in the JNDI Tree

If you do not know the JNDI name for an EJB, you can browse the EJB Server JNDI tree. For information about browsing the JNDI tree using the Oracle WebLogic Server Administration Console, see "View objects in the JNDI" tree in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*.

28.2.3 Transport Configuration Reference

An EJB business service is a Transport Typed Service, which means the type of the transport is determined by the configuration of the service.

The type of an EJB business service is equivalent to a SOAP XML service—in other words, you can use an EJB business service like any other SOAP XML business service. A WSDL is generated when you save the EJB Transport Configuration.

The WSDL is generated based on the interface of the EJB. The EJB transport configuration page provides configuration options for you to control the interface of the service and the WSDL that is generated.

This section describes the endpoint URL format and configuration options for the EJB transport.

28.2.3.1 EJB Endpoint URI

Following is the URI pattern for the EJB transport:

`ejb:jndi_provider:jndi_name`

If the EJB is deployed locally, you need not provide a JNDI provider name. In this case, the URI format is:

`ejb::jndi_name`

For EJB 3.0 business services on Oracle WebLogic Server, `jndi_name` takes the form of `mappedName#BusinessInterface`.

28.2.3.2 EJB Transport Configuration for Business Services

[Table 28–1](#) describes the transport-specific configuration options for the EJB transport.

Table 28–1 EJB Transport Configuration Options for Business Services

Option	To create or edit...
Pass Caller's Subject	Select this check box to have Oracle Service Bus pass the authenticated subject from the proxy service when invoking the EJB and no service accounts are configured. Note that the Service Account field is disabled when this option is selected.
Service Account	Click Browse and select a service account from the list displayed. If no service account is specified, an anonymous subject is used. This option is not available if you use the Pass Caller's Subject option.
Supports Transaction	Select this check box to specify transaction support.
Client Jar	Click Browse and select an EJB client JAR resource from the list displayed.
Converter Jar	Click Browse and select an EJB converter class JAR resource from the list displayed. For more information, see Section 28.2.2.1, "Adding a Client or Converter JAR" and Section 28.4.2.1, "Converter Classes."
Home Interface	EJB 2.1 only – Select the required EJBHome interface from the options populated by the JAR. The JNDI name in this URI sample must be associated with the EJBHome interface you select here. If the EJB is not of the required type or an EJBHome interface is not specified in the client-jar, Oracle Service Bus displays a warning.
Remote Interface	EJB 2.1 only – This field is automatically populated depending on the configuration of the Home Interface.
Business Interface	EJB 3.0 only – Select the business interface in the client JAR that you want to invoke.
Target Namespace	This field is populated by information picked up from the JAR.

Table 28–1 (Cont.) EJB Transport Configuration Options for Business Services

Option	To create or edit...
Style	<p>Select Document Wrapped or RPC according to your requirements. If two or more methods of your stateless session EJB have the same number and data type of parameters, and you want the operations to be document-oriented, you must specify that they be document-wrapped.</p> <p>The style is important because when routing or publishing to the EJB, \$body must have content that matches the style. Also when calling out to an EJB, the style affects the parameter contents, especially for document wrapped. Secondly one usage pattern is to define an EJB business service and then create a proxy service with the same WSDL that routes to the EJB. In this case care must be taken on the style of the WSDL because the client tool used to invoke the proxy might have limitations on the style of the WSDL.</p>
Encoding	Select Encoded or Literal .
Methods	<p>Select the required methods. Click + to expand the method, which lets you edit the default parameter values and select a converter if provided or required.</p> <p>You must exclude the methods with parameters or return types that are not supported by the JAX-RPC stack, or you must associate such arguments with converter classes.</p> <p>You can change the default operation name for a given method. By default, the operation name is the method name. If an EJB contains methods with same name, you must change the operation names so that they are unique. WSDLs require unique operation names.</p> <p>Note: If the credentials or transaction settings are different between the methods for a given EJB, you can customize the methods for a given business service, and create a business service per method. This gives you fine-grained control over transactions and credentials.</p>
Exceptions	<p>This field appears if a method throws a business exception. If an EJB method throws an exception that has data types not supported by Java Web Services (JWS), such as an ArrayList, use the Exceptions field to select a converter class that converts the exception to a type supported by JWS.</p> <p>Your converter class must implement com.bea.wli.sb.transports.ejb.ITypeConverter. Converter classes can only be configured for checked exceptions and not for run-time exceptions.</p> <p>Package the converter class and the converted exception class in the client or converter JAR.</p>

28.2.4 Invoking EJB Business Services

An EJB business service can be used as a SOAP XML business service. You can publish to, route to, or callout to an EJB business service. If you need transaction support, set the quality of service to *Exactly-Once*. See [Section 28.4.1, "Transaction Processing, Retries, and Error Handling."](#)

You can also use the test console to validate your configuration and to help you to determine the shape of the XML request.

28.3 Exposing EJBs as Web Services

You can leverage the EJB transport to easily expose EJBs as Web Services.

Note: You cannot create a proxy service from an existing EJB business service—you must first get the WSDL generated from the EJB business service, and then create the proxy service based on that WSDL. To do so, complete the following steps.

1. Create an EJB business service pointing to the EJB you want to expose, as described in [Section 28.2.3, "Transport Configuration Reference."](#)
2. From the service details page, get the WSDL for the EJB business service.
The WSDL is contained in a JAR file. You can obtain the WSDL only if there is no pending session.
3. Extract the WSDL from the JAR and register it as a WSDL resource.
If the configuration of the business service changes, a new WSDL is generated. If that happens, you must get the new WSDL and re-register it as a WSDL resource.
4. Create a SOAP XML proxy service based on the WSDL.
5. Edit the proxy service pipeline and route to the EJB business service.

You can now invoke the EJB as a Web Service with no need for purchasing an expensive Web Service toolkit or carrying out intrusive actions on the EJB server.

28.4 Advanced Topics

This section includes information about EJB transport that will help you understand how EJB business services behave at run time depending on how they are configured at design time.

28.4.1 Transaction Processing, Retries, and Error Handling

This section describes the EJB transport's transaction processing, retries and failover, and error handling.

28.4.1.1 Transactions

The EJB transport can create, suspend, and propagate transactions. The transaction between Oracle Service Bus and the EJB server are XA transactions. If you use transactions with HTTP tunneling or have a dedicated communication channel, you must set the *security interoperability* mode for the transaction manager to *performance*. For information about setting the security interoperability mode and other transaction configurations, see "Configuring Transactions" in *Oracle Fusion Middleware Programming JTA for Oracle WebLogic Server*.

For the deployment descriptors to be set appropriately for XA-capable resources, you must set the XA attribute on the referenced connection factory before creating a proxy service.

To determine the behavior of the EJB business service, considerations include whether the proxy service pipeline has a transactional context, and what quality of service (QoS) settings are specified in the pipeline when invoking the service:

QoS Best-Effort – If *Best-Effort* QoS is specified in the pipeline, no transaction is propagated to the EJB—any ongoing transaction is suspended before invocation, and resumed after invocation.

QoS Exactly-Once – If *Exactly-Once* QoS is specified in the pipeline, and If the EJB does not support transactions (that is, if the Supports Transaction option on the EJB transport configuration page is unchecked), no transaction is propagated to the EJB. As in the case of *Best-Effort*, any ongoing transaction is suspended before invocation and resumed afterwards.

or

If the EJB supports transactions (that is, if the Supports Transaction option on the EJB transport configuration page is checked), the EJB is invoked in the context of a transaction—any ongoing transaction is propagated to the EJB. If no transaction is present, a transaction is created before invocation and committed afterwards.

For more information about QoS in Oracle Service Bus services, see "Quality of Service" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

28.4.1.2 Retries and Failover

Assuming that the EJB business service is configured for retries or failovers, the EJB transport distinguishes the following types of exceptions:

- Runtime Exceptions or Remote Exceptions—typically unexpected fatal errors or communication exceptions
- Exception raised by the JAX-RPC engine—exceptions that occur during the XML to Java conversion
- EJB Checked Exceptions—exceptions declared in the EJB method signature specific to the EJB implementation; also called Business Exceptions

Retries and failover are based on the type of errors and also in the QoS:

QoS Best-Effort – If a run-time or remote exception is thrown, the EJB transport attempts retries or failovers.

If an exception occurs in the JAX-RPC engine, an error is raised to the pipeline and no retries or failover attempts are made.

If an EJB Checked Exception is thrown, an error is raised to the pipeline and no retries or failover attempts are made.

QoS Exactly-Once – If a run-time or remote exception is thrown and the ongoing transaction has been set as *rollback only* (likely by the EJB container), it means the EJB container has been reached and a fatal error either occurred within the EJB container or the EJB. In this case, no retries or failover attempts are made and an error is raised to the pipeline.

If a runtime or remote exception is thrown but the ongoing transaction has not been set as *rollback only*, it means an error occurred before the invocation of the EJB container and the EJB transport will attempt retries or failovers. Note that in this case, the EJB transport still respects the *exactly-once* semantic.

If an exception occurs in the JAX-RPC engine, the EJB transport sets the ongoing transaction to *rollback only* and an error is raised to the pipeline; no retries or failover attempts are made.

If an EJB Checked Exception is thrown, an error is raised to the pipeline and no retries or failover attempts are made.

See [Section 28.4.1.1, "Transactions"](#) for other repercussions of QoS specifications for an EJB business service.

28.4.1.3 Error Handling

When throwing a checked exception, according to the EJB specifications, the ongoing transaction can be specified as *rollback only*.

If the ongoing transaction is set as *rollback only* by the EJB developer, the transaction is eventually rolled back by its creator (most likely the proxy service).

If the ongoing transaction is not set to *rollback only*, and a checked exception is raised, it is important to catch EJB checked exceptions in the pipeline with an error handler. If those exceptions are not caught, the pipeline errors are propagated back to the proxy service. The proxy service, in turn, is likely to rollback the ongoing transaction (depending of the transport implementation)—this may not be the intended result.

For example, assume you have an EJB with the following method:

```
public void withdrawFunds(float amount) throws RemoteException,
InsufficientFundsException {...}
```

Also assume that when an `InsufficientFundsException` exception is thrown, the EJB does not set the current transaction as *rollback only*. In most scenarios, it is wrong to allow the proxy service to roll back the transaction—you may need to configure an error handler in the pipeline to catch the error and avoid this scenario.

28.4.2 Supported Types and Converter Class

The EJB transport is responsible for the conversion between XML and Java. The conversion is performed by the Oracle WebLogic Server JAX-RPC engine.

The EJB transport natively supports the following types:

- Primitive types
- `XmlObject` (both Apache and Oracle versions)
- Schema generated `XMLBeans` (both Apache and Oracle versions)
- `JavaBean` classes

For the full list of natively supported types, see "Using JAXB Data Binding" in *Oracle Fusion Middleware Getting Started With JAX-WS Web Services for Oracle WebLogic Server*.

An EJB method can use parameters/return types that are either not supported by the JAX-RPC engine (an error is reported at design time) or that do not map directly to XML (errors occur at run time). The most commonly used unsupported types are:

- "Object", "Object[]"
- Java Collections, as they are not strongly typed (for example, List, Set)
- Java classes that do not follow the `JavaBean` pattern (for example, Map)

You can write a custom converter class that converts those types into types more suitable for conversion between XML and Java. The EJB transport supports custom converter classes.

28.4.2.1 Converter Classes

A Converter class is a Java class that implements and conforms to the contract defined by the `com.bea.wli.sb.transports.ejb.ITypeConverter` Java interface of the Oracle Service Bus public API.

To use a converter class for an EJB business service, you must:

1. Create a converter class by implementing and compiling the interface.

2. Add the converter class to the client JAR or to a converter class JAR file (See [Section 28.2.2.1, "Adding a Client or Converter JAR"](#)).
3. When customizing the method configuration during the creation of an EJB business service, navigate to one of the parameter/return types and select the desired converter. See [Section 28.2.3, "Transport Configuration Reference"](#)—the service configuration user interface displays a list of the converters available that can be applied to a particular parameter/return type.

28.5 Troubleshooting

The information in this section is provided to help you troubleshoot problems when designing or running an EJB business service.

28.5.1 Enabling Debug Mode

The EJB transport uses the same logger as other Oracle Service Bus transports. To enable the debug mode, before starting the server, edit the `alsbdebug.xml` file in the domain directory and set the category `alsb-transport-debug` to `true`. For more information about the `alsbdebug.xml` file and the debug flags, see "Debugging Oracle Service Bus" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

28.5.2 Temp Directories

During design time, the EJB transport generates files in a subfolder in the `temp` directory. It is safe to delete those folders and files, and sometimes may be useful to check them to determine what went wrong during activation.

28.5.3 Deployed Application

When an EJB business service is created an application is deployed on the Oracle Service Bus Server. You can use the Oracle WebLogic Server Administration Console to monitor and tune this application.

28.5.4 Errors

The following items may help in the event that you need to troubleshoot a problem with an EJB business service:

- The following error when creating a business service is due to a Windows operating system limitation—paths containing more than 255 characters are not supported:

```
The system cannot find the path specified):Probably the string length of the
path of the file being extracted was too long
```

You can try to reduce the path length by creating a shorter path to the Oracle Service Bus domain, or you can use the following option to override the Oracle WebLogic Server `temp` directory when starting the server:

```
-Dweblogic.j2ee.application.tmpDir=$desired_short_dir
```

- If you get an XML marshalling error when invoking an EJB business service and you believe the request to be valid against the service WSDL, you probably need to write a converter class. For information, see [Section 28.4.2.1, "Converter Classes."](#)

- If the EJB interfaces and stubs are changed on the remote server, the first time you try to invoke the new EJB, an error is thrown. Those changes on the remote server are not visible to Oracle Service Bus—it tries to invoke the cached EJB stubs, which are no longer valid. However, when the invocation error occurs, the transport assumes that those stubs are now invalid, and remove them from the cache—in this way, the error is prevented on subsequent attempts to invoke the EJB. To avoid this first-time error, you can reset the JNDI Provider in the Oracle Service Bus Console.

JEJB Transport

This document, which describes the JEJB transport and provides instructions on creating and using JEJB proxy and business services, contains the following sections:

- [Section 29.1, "About the JEJB Transport"](#)
- [Section 29.2, "Creating and Configuring JEJB Services"](#)
- [Section 29.3, "Use Cases"](#)
- [Section 29.4, "Transport Configuration Reference"](#)
- [Section 29.5, "Testing JEJB Services"](#)
- [Section 29.6, "UDDI Integration"](#)

29.1 About the JEJB Transport

The JEJB transport lets you pass Plain Old Java Objects (POJOs) through Oracle Service Bus. For example, you can use an EJB to invoke a remote EJB operation or a non-EJB service, or you can invoke an EJB operation with a non-EJB request. Use case details are described in [Section 29.3, "Use Cases."](#)

To a J2EE client, a JEJB proxy service pipeline looks like a stateless session bean. A JEJB proxy service, on receiving the method arguments, passes their XML representation in the pipeline \$body variable. POJO arguments are represented as the XML fragment. This XML fragment contains the location of the actual POJO stored in the object repository within the pipeline. XML arguments can either be passed by value or by reference (referencing the actual object stored in the object repository). Primitive types are always passed by value.

For more detailed information on POJOs in message flows, see "Java Content in the body Variable" and "Extensibility Using Java Callouts and POJOs" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

The JEJB transport is always synchronous, so the pattern is always request-response.

For deployment, Oracle Service Bus automatically packages JEJB proxy services as enterprise archives (EARs).

29.1.1 Difference Between the JEJB Transport and the EJB Transport

The EJB transport, available only for business services, invokes remote EJBs through the Java Web Services (JWS) framework. The JEJB transport, which lets you invoke remote EJBs and external services with POJOs, passes POJOs directly through Oracle Service Bus to the target EJB methods using an RMI serialization/deserialization cycle.

The EJB transport provides a "Support Transaction" flag, but all Oracle Service Bus proxy services provide transactional support, making the transaction option unnecessary for JEJB business services.

29.1.2 Environment Values

The JEJB transport stores the following environment values for JEJB services:

- Service URI
- Work Manager
- UDDI Auto Publish (Proxy Services)
- Service Account (Business Services)

29.1.3 WSDL Generation

For proxy and business services, the JEJB transport generates a Document-style WSDL with Literal encoding that is used solely for describing the message passed to the pipeline. The WSDL format lets you leverage Oracle Service Bus WSDL features such as per-operation monitoring.

The message structure defined in the WSDL may differ from the actual pipeline message at run time if you inline your POJO arguments in the message using the "Pass XMLBeans by value" option, described in [Table 29-1](#).

Following is the behavior of the pipeline message format for XMLBeans type of parameters:

Proxy Services

- Request Parameters: Request parameters in the pipeline message refer to an inline Xml Object if "Pass XMLBeans by value" is true; otherwise the reference is to java-content ref.
- Response Parameter: The response may refer to an inline XML Object or the java-content ref, as the response may come in either form from the business service.

Business Services

- Response Parameter: Return parameters in the pipeline message refer to an inline Xml Object if "Pass XMLBeans by value" is true; otherwise the reference is to java-content ref.
- Request Parameters: Request method parameters in the pipeline message may refer to an inline XML Object or the java-content ref, as the request may come in either form from the proxy service.

29.1.4 Error Handling

This section describes how the JEJB transport handles errors.

29.1.4.1 Exception Propagation in the Response

The JEJB transport stores request exceptions in the object repository and propagates them to the JEJB proxy service through the \$fault variable. \$fault would contain the location of the exception instance within the <java-exception> <java-content ref="jcid"/> </java-exception> element, where jcid is the reference to the exception instance stored in the object repository.

To propagate the user exception to the client, the JEJB proxy service would expect the response in one of these formats:

- env:Envelope/env:Fault/detail/mc:java-exception

Format, where `jcId` is a reference to the error in the object repository:

```
<detail>
  <mc:java-exception>
    <mc:java-content ref="jcId"/>
  </mc:java-exception>
  ...
</detail>
```

- env:Envelope/env:Fault/detail/mc:fault/mc:java-exception

Format, where `jcId` is a reference to the error in the object repository:

```
<detail>
  <mc:fault xmlns:mc="http://www.bea.com/wli/sb/context">
    <mc:java-exception>
      <mc:java-content ref="jcId"/>
    </mc:java-exception>
    ...
  </mc:fault>
</detail>
```

- env:Envelope/
env:Fault/detail/mc:fault/mc:details/con1:ReceivedFaultDetail/con1:detail/mc:java-exception

Format, where `jcId` is a reference to the error in the object repository:

```
<con:details>
  <con1:ReceivedFaultDetail
xmlns:con1="http://www.bea.com/wli/sb/stages/transform/config">
  <con1:faultcode>soapenv:Server</con1:faultcode>
  <con1:faultstring>checkExceptionConversion</con1:faultstring>
  <con1:detail>
    <mc:java-exception>
      <mc:java-content ref="jcId"/>
    </mc:java-exception>
  </con1:detail>
  </con1:ReceivedFaultDetail>
</con:details>
```

If you want to raise your own exceptions to return to the caller, raise them in a Java Callout in the proxy service pipeline.

29.1.4.1.1 Java Callout and Service Callout Exceptions If you configure a Java Callout or Service Callout in an error handler for "Reply with Failure," you must format the `$body` so that it conforms to one of the previously described fault structures.

29.1.4.2 Application and Connection Errors

This section describes the conditions under which the JEJB transport throws application and communication errors, which are subject to the retry configuration on a service.

29.1.4.2.1 Connection Errors The JEJB transport throws connection errors in the following situations:

- NamingExceptions looking up the EJBs raised during the remote call.
- A run-time or remote exception is thrown, but the ongoing transaction has *not* been set as rollback-only, signifying that the error occurred before the invocation of the EJB container.

29.1.4.2.2 Application Errors The JEJB transport throws application errors in the following situations:

- A run-time or remote exception is thrown and the ongoing transaction has been set as rollback-only (likely by the EJB container), signifying the EJB container has been reached and a fatal error either occurred within the container or within the EJB itself.
- Business exceptions defined in the EJB business interface.
- An exception caused by a faulty encoding of the parameters in XML.

29.2 Creating and Configuring JEJB Services

This section provides instructions for creating and configuring JEJB proxy and business services.

29.2.1 Creating and Packaging Your Client EJB JAR

This section provides guidelines on creating and packaging POJOs to represent EJB invocations and operations for JEJB proxy and business services.

- Define an interface of type `java.io.Serializable` and include any necessary helper classes, such as business exceptions. The interface does not need to extend any class as long as the interface is valid for one of the RMI protocols described in [Section 29.4.1, "JEJB Endpoint URI,"](#) or is valid for JMS messages if you are using JMS to invoke EJBs.

Note: Though not required, you can:

- Make the interface a Remote interface as defined by the EJB 2.1 specification
- or
- Annotate methods with the `javax.ejb.Remote` annotation to designate it as an EJB 3.0 business interface

For a simple POJO interface (no EJB Remote interface) or an interface annotated with `javax.ejb.Remote`, the JEJB transport provider generates the 3.0 EJB interface out of the JEJB proxy service.

For a Remote interface, the JEJB transport provider generates the 2.1 EJB interface out of the JEJB proxy service.

- The objects received as arguments must be passable to any required classes in a Java Callout archive resource.
- An array of any type is considered a POJO.
- Avoid unnecessary serialization/deserialization cycles by not duplicating the JARs uploaded as Archive Resources to support Java Callouts. Package all archive resource classes in a single archive JAR so that multiple Java Callouts do not serialize/deserialize the objects.

- Package your interface and dependent classes in a single "client" JAR and import it into Oracle Service Bus. While this is the client JAR you will select when configuring a service, it is not technically a fully expanded EJB client JAR, because it contains no stubs. The actual bean (hence Oracle WebLogic Server stub generation) does not exist until a JEJB proxy service is created and activated.

29.2.2 Register a JNDI Provider Resource (Business Services)

A JNDI Provider resource allows you to specify the communication protocols and security credentials used to retrieve EJB stubs bound in the JNDI tree of remote Oracle WebLogic Server domains. (For more information on how to set up a JNDI tree, see *Oracle Fusion Middleware Programming JNDI for Oracle WebLogic Server*.)

Typically, the target EJB is not located in the same domain as Oracle Service Bus. In this case, you must register a JNDI Provider resource. When the EJB is located in the same domain, you can define a provider to specify credentials and take advantage of stubs caching, though doing so is optional.

The JNDI provider has a high performance caching mechanism for remote connections and EJB stubs. The preferred communication protocol from Oracle Service Bus to an Oracle WebLogic Server domain is `t3` or `t3s`. If messages need to go through a firewall, you can use HTTP tunneling.

Note: Although it is possible to use an Oracle WebLogic Server Foreign JNDI Provider, Oracle recommends that you do not.

The transport does not support two-way SSL or CLIENT CERT to look-up JNDI tree or access a method on an EJB.

29.2.2.1 Adding a JNDI Provider

For information about registering and configuring a JNDI provider resource in Oracle Service Bus, see [Section 4.8, "JNDI Providers."](#)

29.2.3 Configuring a JEJB Proxy or Business Service

This section describes the high-level steps for configuring a JEJB proxy or business service. The scenarios described in [Section 29.3, "Use Cases"](#) illustrate when you need to create JEJB proxy and business services. Each use case provides general implementation guidelines.

1. On the General page of the service, select the **Transport Typed** option.
2. On the Transport page:
 - a. Select the **jejb** protocol.
 - b. Enter the Endpoint URI information as described in [Section 29.4.1, "JEJB Endpoint URI."](#)
 - c. Set other global transport options as described in [Section 4.3.6, "Proxy Service Transport Configuration Page"](#) and [Section 4.2.4, "Business Service Transport Configuration Page."](#)
3. On the JEJB Transport page, configure the transport options for the service, described in [Section 29.4.2, "JEJB Transport Configuration for Proxy Services"](#) and [Section 29.4.3, "JEJB Transport Configuration for Business Services."](#)
4. Set the remaining service options as described in [Section 4.3, "Proxy Service Configuration"](#) and [Section 4.2, "Business Service Configuration."](#)

- See your particular use case in [Section 29.3, "Use Cases"](#) for implementation guidelines.

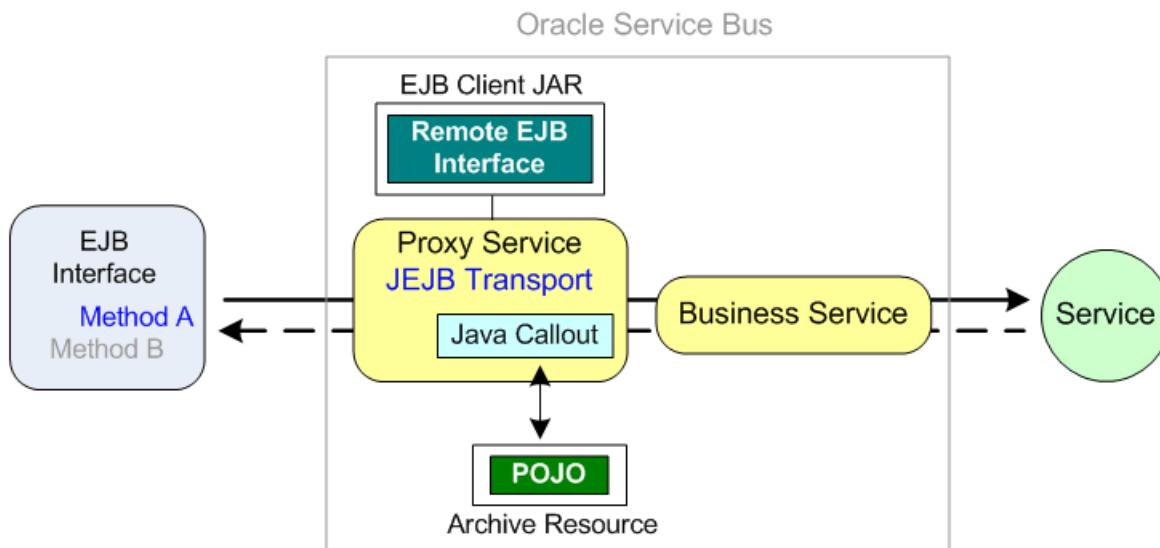
29.3 Use Cases

Following are the supported use cases for using the JEJB transport in proxy and business services. Each use case provides implementation guidelines for you to use in conjunction with the general service configuration, as described in [Section 29.2.3, "Configuring a JEJB Proxy or Business Service."](#)

29.3.1 EJB Invokes an External Service

You can invoke an external service with an EJB through Oracle Service Bus, illustrated by [Figure 29-1](#).

Figure 29-1 An EJB Invokes an External Service



In [Figure 29-1](#), the JEJB proxy service serves as a stateless session bean to the EJB client interface. The JEJB transport provider for the proxy service generates a stateless session EJB from the remote/business interface in the client JAR and the pipeline, then deploys it as an EAR at the JNDI address specified in the endpoint URI.

Caution: Be sure to install policies that protect the JNDI entries from being modified.

The EJB makes a call to a remote interface provided by the proxy service EJB client JAR, passing transaction and security details to the proxy service as well.

The EJB client interface is a POJO with method arguments that the JEJB transport provider represents as a WSDL and passes into the proxy service \$body variable as XML. You can introspect the \$body content to transform the message into the required format to pass to the business service and invoke the external service. The actual POJO is stored in the object repository, and the XML in the \$body references it with a <java-content ref=""> element.

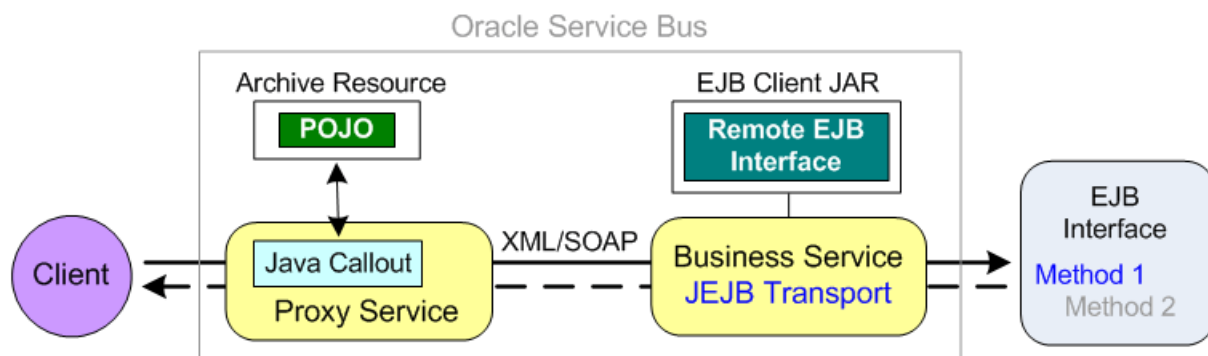
In the response, provide a Java Callout that converts the response to an EJB return format that gets passed to the calling EJB method. View the proxy service's generated WSDL to see the expected message format. For information on viewing the generated WSDL, see "Viewing Resource Details" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

Note: In the proxy service pipeline, you can pass POJO arguments to Java Callouts, to another proxy service using, for example, a Service Callout or a Publish action, or to a business service.

29.3.2 Non-EJB Client Invokes an EJB

You can invoke an EJB with a non-EJB client through Oracle Service Bus, as illustrated in [Figure 29-2](#).

Figure 29-2 A Non-EJB Client Invokes an EJB



In [Figure 29-2](#), a non-EJB client makes a call to a proxy service configured with a transport that matches the request; for example, a JMS proxy service making an invocation with a JMS topic or queue.

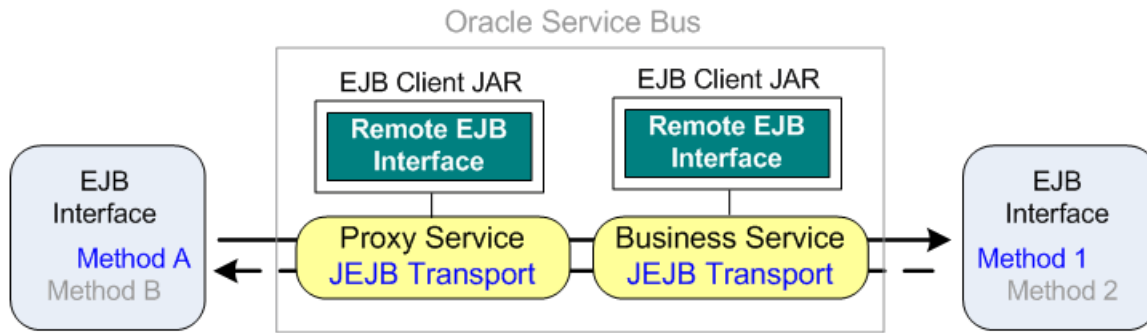
You configure a Java Callout in the request, which converts the request into an XML representation of an EJB call in the \$body variable. Put operations in the \$operation variable. View the business service's generated WSDL to see the expected message format. For information on viewing the generated WSDL, see "Viewing Resource Details" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

The JEJB business service uses its generated WSDL to map the incoming message to the EJB remote interface and invoke the remote EJB method directly.

29.3.3 EJB Invokes EJB

You can invoke an EJB with an EJB through Oracle Service Bus, as illustrated in [Figure 29-3](#).

Figure 29–3 An EJB Invokes an EJB



In Figure 29–3, the EJB call is passed through the proxy and business services to invoke another EJB method. Rather than making a direct RMI call outside of Oracle Service Bus, this architecture lets you leverage Oracle Service Bus features such as message routing, UDDI integration, alerts, monitoring, reporting, and result caching.

The JEJB transport provider for the proxy service generates a stateless session EJB from the remote/business interface in the client JAR and the pipeline, and then deploys it as an EAR at the JNDI address specified in the endpoint URI.

At run time the JEJB proxy service receives a POJO as method argument, stores it in the object repository, and generates an XML representation of the POJO in the proxy service \$body variable according to the generated proxy service WSDL. The proxy service passes the message to the business service, and the business service uses its generated WSDL to map the message to the remote interface and invoke the remote method directly.

If you want to keep your pipeline logic independent of the JEJB transport, you can route messages from the JEJB proxy service to a local proxy service. The local proxy service can then perform the pipeline logic before forwarding the messages to the JEJB business service. If you use this pattern, make sure the local proxy is of type WSDL (with support for the business service operations) so that it has access to the Java objects.

29.4 Transport Configuration Reference

This section provides details on constructing endpoint URIs and configuring the JEJB transport in proxy and business services.

29.4.1 JEJB Endpoint URI

Following are the endpoint URI formats for JEJB proxy and business services.

Note: JEJB services do not support co-located calls.

29.4.1.1 Proxy Service JEJB Endpoint URI

The URL format is *jndi_name*. The URI configured for a JEJB proxy service becomes the global JNDI name for locating the stateless session bean generated by the JEJB transport from the remote/business interface in the client JAR.

Note: For EJB 3.0, *jndi_name* is the mappedName attribute of the @javax.ejb.Stateless annotation in the generated bean. The lookup JNDI name for the generated EJB service is suffixed with *#interface_class*, which is the fully qualified name of the business interface.

You can access the JEJB proxy service as:

- **EJB 2.1** – *protocol://host:port/jndi_name*
- **EJB 3.0** – *protocol://host:port/jndi_name#interface_class*

The *protocol* can be one of the following RMI protocols:

- *iiop* / *iiops* – For generic, server-agnostic use.
- *t3* / *t3s* – For use with Oracle WebLogic Server.
- *http* / *https* – For tunneling and use with Oracle WebLogic Server.

For example:

- EJB 2.1 – *t3://localhost:7001/osb.jejb.myJejbProxy*
- EJB 3.0 – *t3://localhost:7001/osb.jejb.myJejbProxy#com.example.MyEjb3*

29.4.1.2 Business Service JEJB Endpoint URI

Following is the endpoint URI format for a JEJB business service:

jejb:jndi_provider:jndi_ejb_name

The *jndi_provider* is the remote JNDI context.

The *jndi_ejb_name* is the remote EJB's JNDI name.

For example:

- EJB 2.1 – *jejb:myProvider:osb.jejb.myJejbBiz21*
- EJB 3.0 – *jejb:myProvider:myBiz31#osb.jejb.myJejbBiz*

where *#osb.jejb.myJejbBiz* is the fully qualified business interface.

29.4.2 JEJB Transport Configuration for Proxy Services

Use this page to configure transport settings for proxy services using the JEJB transport protocol.

[Table 29–1](#) describes the transport-specific configuration options for business services that use the JEJB transport.

Table 29–1 JEJB Transport Configuration for Proxy Services

Option	Description
Dispatch Policy	Select the instance of the Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists. For information about Work Managers, see “Using Work Managers to Optimize Scheduled Work” in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> .
EJB Spec Version	Select the EJB version of the remote EJB interface.

Table 29–1 (Cont.) JEJB Transport Configuration for Proxy Services

Option	Description
Pass XMLBeans by value	<p>Select this option if you want the transport to generate an "inlined" XML representation of POJO arguments (an XMLObject) whose parameters you can access and manipulate with XQuery expressions.</p> <p>Note: Type information is not available inline for XMLObjects passed by value. If you use this option, you cannot pass the typed XMLObject as the argument in a Java Callout in a proxy service pipeline.</p> <p>Do not select this option if you want to pass the POJO by reference, which also results in better performance.</p> <p>For more information, see Section 29.1.3, "WSDL Generation."</p>
Transaction Attribute	<p>Select one of the following options for handling transactions:</p> <ul style="list-style-type: none"> ▪ Supports – The transport accepts an incoming transaction. Quality of service is exactly-once if the operation is invoked in a transaction and best-effort if the operation is invoked outside of a transaction. ▪ Required – The transport accepts an incoming transaction. If no ongoing transaction exists, the transport starts one. Quality of service is exactly-once. ▪ RequiresNew – The transport always starts a new transaction, suspending an ongoing transaction. Quality of service is exactly-once. ▪ Mandatory – The transport invokes the method in the existing transaction. Quality of service is exactly-once. ▪ NotSupported – The transport suspends an existing transaction and resumes it on invocation. Quality of service is best-effort. ▪ Never – The transport does not invoke the method in a transaction. Quality of service is best-effort.
Client JAR	<p>Click Browse and select an EJB client JAR resource from the list displayed. The client JAR contains the remote or business interface for the remote EJB. The Client JAR is registered as a generic Archive Resource.</p>
Home Interface	<p>EJB 2.1 only – Select the required EJBHome interface from the options populated by the client JAR.</p>
Remote Interface	<p>EJB 2.1 only – This field is automatically populated based on the configuration of the Home Interface.</p>
Business Interface	<p>EJB 3.0 only – Select the business interface from the client JAR that you want to invoke.</p>
Target Namespace	<p>This field is populated by information picked up from the JAR.</p>
Methods	<p>Select the required methods. Click + to expand the method, which lets you edit the default parameter values.</p> <p>You can change the default operation name for a given method. By default, the operation name is the method name. If an EJB contains methods with same name (overloaded), you must change the operation names so that they are unique. WSDLs require unique operation names.</p>

29.4.3 JEJB Transport Configuration for Business Services

Use this page to configure transport settings for business services using the JEJB transport protocol.

Table 29–2 describes the transport-specific configuration options for business services that use the JEJB transport.

Table 29–2 JEJB Transport Configuration for Business Services

Option	Description
Dispatch Policy	<p>Select the instance of the Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists.</p> <p>For information about Work Managers, see “Using Work Managers to Optimize Scheduled Work” in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i>.</p>
EJB Spec Version	Select the EJB version of the remote EJB interface.
Pass XMLBeans by value	<p>Select this option if you want the transport to generate an "inlined" XML representation of POJO arguments (an XMLObject) whose parameters you can access and manipulate with XQuery expressions.</p> <p>Note: Type information is not available inline for XMLObjects passed by value. If you use this option, you cannot pass the typed XMLObject as the argument in a Java Callout in a proxy service pipeline.</p> <p>Do not select this option if you want to pass the POJO by reference, which also results in better performance.</p> <p>For more information, see Section 29.1.3, "WSDL Generation."</p>
Pass Caller’s Subject	As an alternative to selecting a Service Account, select this option to have Oracle Service Bus pass the authenticated subject from the proxy service when invoking the EJB.
Service Account	<p>Click Browse and select a JNDI service account from the list displayed. If no service account is specified, an anonymous subject is used.</p> <p>For more information, see Section 4.19, "Service Accounts."</p>
Client JAR	Click Browse and select an EJB client JAR resource from the list displayed. The client JAR contains the remote or business interface for the remote service. The Client JAR is registered as a generic Archive Resource.
Home Interface	EJB 2.1 only – Select the required EJBHome interface from the options populated by the JAR.
Remote Interface	EJB 2.1 only – This field is automatically populated based on the configuration of the Home Interface.
Business Interface	EJB 3.0 only – Select the business interface from the client JAR that you want to invoke.
Target Namespace	This field is populated by information picked up from the JAR.
Methods	<p>Select the required methods. Click + to expand the method, which lets you edit the default parameter values.</p> <p>You can change the default operation name for a given method. By default, the operation name is the method name. If an EJB contains methods with same name (overloaded), you must change the operation names so that they are unique. WSDLs require unique operation names.</p>

29.5 Testing JEJB Services

In the Oracle Service Bus test console you can test JEJB services that pass POJOs by value ("Pass XMLBeans by value" option), but not by reference. The test console supports only primitive, String, and XML arguments. The transport passes supported POJO arguments and their values as XML specifying the invocation point (the EJB method).

29.6 UDDI Integration

This section describes the UDDI publish and import details for JEJB proxy services:

29.6.1 UDDI Publish

JEJB proxy services publish the following properties to a UDDI registry:

- URI
- EJB Spec Version
- Client JAR
- Home Interface (for EJB 2.1 only)
- Remote Interface (the Business Interface for EJB 3.0)
- Method Names (Not included are operation aliases, parameters, or return details. Method names are passed in one property with all the method signatures appended. Method signatures are separated by the # character.

29.6.2 UDDI Import

This section describes how the JEJB transport handles service import from a UDDI registry.

- URI – The JEJB transport provider attempts to match the host:port information from the URI property in the UDDI registry with a JNDI provider resource registered on the server.

If the transport provider cannot find a JNDI provider, the import fails. However, if the no JNDI provider is found, but the host:port matches the localhost IP and listen port, the resulting business service will be local (no JNDI provider).
- Client JAR – The transport provider downloads the client JARs and, if the manifest classpath exists in the JARs, creates the corresponding JAR resources in the matching directory structure. The first URI in the list is the root client JAR. If no manifest classpath exists in the JARs, you must manually add the resource JARs as dependencies to the root JAR. If a resource in the imported client JAR has the same name as another resource in the domain, the imported resource overwrites the existing resource.

Make sure that the client JAR you are importing does not already exist in your domain.
- Method Names – Methods included in the corresponding property are automatically selected in the endpoint configuration. All the other methods are marked as excluded (deselected).

The following sections describe features and concepts related to interoperability between Oracle Service Bus and WebLogic JMS, and between Oracle Service Bus and WebSphere MQ:

- [Section 30.1, "Overview of JMS Interoperability"](#)
- [Section 30.2, "Asynchronous Request-Response Messaging"](#)
- [Section 30.3, "Using SOAP-JMS Transport"](#)
- [Section 30.4, "Naming Guidelines for Domains, WebLogic, and JMS Servers"](#)
- [Section 30.5, "Specifying the JMS Type for Services"](#)
- [Section 30.6, "WSDL-Defined SOAP Fault Messages"](#)
- [Section 30.7, "Interoperability with WebSphere MQ"](#)
- [Section 30.8, "Message ID and Correlation ID Patterns for JMS Request/Response"](#)
- [Section 30.9, "Using the JMS Transport"](#)

30.1 Overview of JMS Interoperability

Java API for XML-Remote Procedure Call (JAX-RPC) is considered the core Java API to build and deploy Web services using J2EE. JAX-RPC provides a simple, robust platform for building Web services applications by abstracting the complexity of mapping between XML types and Java types and the lower-level details of handling XML SOAP messages from the developer. JAX-RPC introduces a method call paradigm by providing two programming models:

- A server-side model for developing Web services endpoints using Java classes or stateless EJB components
- A client-side model for building Java clients that access Web services as local objects.

JAX-RPC mandates the use of SOAP and interoperability with other Web services built with other technologies. If you already have a stateless session EJB or a Java class that performs your business logic, J2EE 1.4 lets you expose it as a service in a standard manner using JAX-RPC.

All Oracle Service Bus service types support JMS transport. Proxy services and business services must be configured to use JMS transport as described in the [Section 2.3, "Working with Proxy Services"](#) and [Section 2.2, "Working with Business Services."](#)

For information about Oracle WebLogic Server JMS, see:

- "Managing Your Applications" in *Oracle Fusion Middleware Programming JMS for Oracle WebLogic Server*
- "Configure JMS Servers" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*

Note: Oracle Service Bus supports the MQ Extended Transactional Client which is vital for remote transactional support configuration.

30.2 Asynchronous Request-Response Messaging

Messaging can be categorized as follows:

- One-way
- Synchronous request-response
- Asynchronous request-response

However, messaging over JMS is only one-way or via asynchronous request-response. Asynchronous request-response messaging using JMS is an alternative to messaging using HTTP or HTTP(S).

Using asynchronous request-response messaging has the following advantages:

- The request thread does not get blocked while waiting for the response. This removes a thread management issue that can occur when numerous blocking request-response invocations are made. However, HTTP and HTTP(S) support a non blocking mode of operation.
- The messaging is more reliable than HTTP because it can be:
 - Persisted on disk
 - Queued when the service is not available
 - Re-delivered if the server has an error or fails when the message is being processed

If you are using IBM WebSphere MQ, asynchronous request-response messages may be the best approach for interacting with some mainframes. The asynchronous service must echo the correlation ID or the message ID depending on the JMS request-response pattern that you use. The format of either ID used by Oracle Service Bus is compatible with IBM WebSphere MQ and with target services that use MQ native interfaces. For more information, see [Section 30.8, "Message ID and Correlation ID Patterns for JMS Request/Response."](#)

Asynchronous request-response messages are handled by the outbound and inbound transports. That is, the message flow, except for the `$outbound` and `$inbound` transport specific data, does not distinguish between JMS request-response and HTTP request-response.

Oracle Service Bus supports bridging between synchronous and asynchronous request and response. For example, a proxy service can be invoked using HTTP, and the proxy service routes to a JMS request-response business service. This is called synchronous-to-asynchronous service switching.

30.3 Using SOAP-JMS Transport

You can use JMS for SOAP transport instead of HTTP, as SOAP is transport-agnostic. SOAP JMS transport is necessary especially for enterprise customers. Oracle Service

Bus supports SOAP messages with JMS request-response. Oracle Service Bus supports interoperability with Oracle WebLogic Server SOAP-based clients and services.

JMS is also an approach for reliable messaging.

30.3.1 Interoperating with Oracle WebLogic Server

When you use JMS binding to configure a service in Oracle WebLogic Server 9.x and 10.0, use the following SOAP-JMS URL format with Oracle WebLogic Server:

```
jms://host:port/contextURI/serviceName?URI=destJndiName
```

When you configure the service in Oracle Service Bus, the URL must have the following format:

```
jms://host:port/connection_factory/jndi_destination
```

Both formats use the same `jndi_destination`. The `jndi_destination` must be the JNDI name of an existing `QueueConnectionFactory` in the target Oracle WebLogic Server.

When you invoke Oracle WebLogic Server services from Oracle Service Bus, you must set the URI as a JMS property with the value as `/contextURI/serviceName`, inside the message flow on the outbound variable (`$outbound`) before a request is sent to the business service. Use the Transport Headers action to set this property.

For information about setting `$outbound`, see "Inbound and Outbound Variables" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

When an Oracle WebLogic Server Web services client invokes an Oracle Service Bus proxy service, the URI property is ignored. However, it can be passed through to an invoked service using the pass through options of the Transport Headers action. For more information, see [Section 2.4.32, "Adding and Configuring Transport Headers Actions in Message Flows."](#)

Oracle Service Bus can only invoke Oracle WebLogic request-response services running on version 9.2 or later. However, it can also invoke one-way JMS services.

30.3.1.1 Configuring the Response Queues for Cross-domain JMS Calls

When you configure the response queue for cross-domain JMS calls, make sure that there is a separate response queue corresponding to each requesting managed server. For example:

If there are two Oracle Service Bus clustered domains (domain A and domain B) talking to a WLS domain. Let's say the WLS domain has 2 managed servers. If domain A has 3 managed servers and domain B has 4 managed servers, you need to configure 7 distinct queues to serve as response queues ($3 + 4 = 7$) on the WLS domain for sending responses back to domain A and domain B. These queues could be distributed queues (with members on both the managed servers of the WLS domain).

Note: When the JMS requests come from multiple proxy services hosted by different remote domains, you must configure the back-end domain hosting the JMS business service with the separate sets of response queues corresponding to each requesting domain.

30.4 Naming Guidelines for Domains, WebLogic, and JMS Servers

If you are working with more than one domain, make sure that your configuration conforms to the following requirements:

- Oracle WebLogic Server instances and domain names are unique.
- WebLogic JMS server names are uniquely named across domains.
- If a JMS file store is being used for persistent messages, the JMS file store name must be unique across domains.

Note the following rules for JMS Server names:

- You cannot have duplicate JMS server names within the same domain. If you do, when messages are sent to a destination at a particular JMS server, Oracle Service Bus cannot determine to which server the message should be sent.
- If you are using Store and Forward (SAF), duplicate JMS Server names in different domains do not pose a problem.
- In the case of cross-domain communication, duplicate JMS server names can be a problem when you use the `ReplyTo` function. The `ReplyTo` message sent from a given domain is returned to the JMS server on the same domain that received the message instead of being returned to the domain that sent the original message.

For more information on how to configure and manage WebLogic JMS:

- "Managing Your Applications" in *Oracle Fusion Middleware Programming JMS for Oracle WebLogic Server*
- "Configure JMS Servers" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*

30.5 Specifying the JMS Type for Services

To support interoperability with heterogeneous endpoints, Oracle Service Bus allows you to control the content type used, the JMS type used, and the encoding used when configuring message flows. The JMS type can be byte or text. For more information, see "Content Types, JMS Type, and Encoding" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

30.6 WSDL-Defined SOAP Fault Messages

When consuming a WSDL that explicitly defines a fault, the WebLogic clientgen tool generates a subclass of `java.lang.Exception` for the XML fault type. When the Oracle WebLogic Server JAX-RPC stack inspects a SOAP response message and determines that the response message contains a SOAP fault, it tries to map the fault to a clientgen-generated exception Java class.

For example, if a WSDL contains the definitions shown in the following listing, the clientgen tool generates a Java class `com.bea.test.TheFaultType` that extends `java.lang.Exception`. A JAX-RPC client can catch `com.bea.test.TheFaultType` when invoking the related method of the service stub.

Example 30-1 Sample WSDL Definitions

```
<definitions ... xmlns:s0="http://www.oracle.com/test/">
  ...
  <types>
```

```

<xsd:schema targetNamespace="http://www.oracle.com/test/">
  ...
  <xsd:complexType name="theFaultType">
    <xsd:sequence>
      <xsd:element name="ID" type="xsd:int" />
      <xsd:element name="message" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="theFault" type="theFaultType" />
</xsd:schema>
</types>
...
<message name="theFaultMessage">
  <part element="s0:theFaultPart" name="theFault" />
</message>
...
<binding ...>
  <operation ...>
    <soap:operation soapAction="..." style="document" />
    <input ...>
      ...
    </input>
    <output ...>
      ...
    </output>
    <fault ...>
      <soap:fault name="theFaultPart" use="literal" />
    </fault>
  </operation>
</binding>
...
</definitions>

```

The SOAP message must contain a fault of the correct format so that the JAX-RPC stack throws the correct exception. If the fault is constructed from inside an Oracle Service Bus message flow, you must:

1. Replace the node for the `$body` variable with the following sample SOAP:

Example 30–2 Sample SOAP

```

<soap-env:Body>
  <soap-env:Fault>
    <faultcode>
      xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">soap:Server</faultcode>
    <faultstring>Some literal string</faultstring>
    <detail>
      <test:theFault>
        <test:ID>Any user defined code</test:Id>
        <test:message>A specific literal message</test:message>
      </test:theFault>
    </detail>
  </soap-env:Fault>
</soap-env:Body>

```

where:

- `soap-env` is the system prefix for the namespace `http://schemas.xmlsoap.org/soap/envelope/`
- `test` is the prefix for the namespace `http://www.oracle.com/test/`

If the prefix `test` is not already known to Oracle Service Bus, you must declare it.

2. Configure a reply action with failure.

For information about configuring Reply Actions in the Oracle Service Bus Console, see [Section 2.4.22, "Adding and Configuring Reply Actions in Message Flows."](#)

The `clientgen` tool is used to generate the client-side artifacts, such as the JAX-RPC stubs, needed to invoke a Web Service. See "Ant Task Reference" in *Oracle Fusion Middleware WebLogic Web Services Reference for Oracle WebLogic Server*.

30.7 Interoperability with WebSphere MQ

Oracle Service Bus connects to WebSphere MQ via the WebSphere MQ JMS interface. That is, Oracle Service Bus is a WebSphere MQ JMS client.

WebSphere MQ can interface with Oracle Service Bus in two ways:

- Oracle Service Bus acts as the front-end of WebSphere MQ to accept service requests from other applications and converts them to WebSphere MQ requests.
- WebSphere MQ sends messages to other applications through Oracle Service Bus.

For more information, see [Section 33.9.1, "Using the WebSphere MQ JMS Interface."](#)

30.8 Message ID and Correlation ID Patterns for JMS Request/Response

JMS is a standard API for accessing enterprise messaging systems. Oracle WebLogic JMS:

- Enables Java applications sharing a messaging system to exchange messages.
- Simplifies application development by providing a standard interface for creating, sending, and receiving messages.

For an overview and features of JMS, see *Oracle Fusion Middleware Configuring and Managing JMS for Oracle WebLogic Server*.

This section describes the Message ID and Correlation ID patterns supported in Oracle Service Bus for JMS request-response and describe how Oracle Service Bus uses these patterns to interoperate with Java API for Remote Procedure Call (JAX-RPC) Web services. Examples are also provided.

The following topics are included:

- [Section 30.8.1, "Overview of JMS Request-Response and Design Patterns"](#)
- [Section 30.8.2, "JMS Message ID Pattern"](#)
- [Section 30.8.3, "JMS Correlation ID Pattern"](#)
- [Section 30.8.4, "Comparison of Message ID and Correlation ID Patterns"](#)
- [Section 30.8.5, "Interoperating with JAX-RPC Over JMS"](#)
- [Section 30.8.6, "JMS Message ID Pattern Examples"](#)

30.8.1 Overview of JMS Request-Response and Design Patterns

Messaging provides high-speed, asynchronous, program-to-program communication with guaranteed delivery and is often implemented as a layer of software called Message Oriented Middleware (MOM).

In enterprise computing, messaging makes communication between processes reliable, even when the processes and the connection between them are not so reliable. Processes may need to communicate for the following reasons:

- One process has data that needs to be transmitted to another process.
- One process needs to remotely invoke a procedure in another process.

Asynchronous request-response messages are the best approach to interacting with some mainframes, if you are using IBM WebSphere MQ.

30.8.1.1 Patterns for Messaging

Messaging patterns describe the format of messages that flow between parts of a system built with a MOM. There are several types of messages as follows:

- A Command Message enables procedure call semantics to be executed in a messaging system.
- A Document Message enables a messaging system to transport information, such as the information that should be returned to a sender as a result of a command message.
- An Event Message uses messaging to perform event notification.
- A Reply Message handles the semantics of remote procedure call results, that require the ability to handle both successful and unsuccessful outcomes.
- A Reply Specifier enables a program making a request to identify the channel through which a reply should be sent.
- A Correlation Identifier enables a requesting program to associate a specific response with its request. When the data to be conveyed spans several messages, a Sequence Identifier enables the receiver to accurately reconstruct the original data.
- Message Expiration enables a sender to set a deadline by which the message should either be delivered or ignored.
- Message Throttle enables a receiver to control the rate at which it receives messages.

In the case of Oracle Service Bus, each reply message should contain a unique identifier called the correlation ID, which correlates the request message and its reply.

When the caller creates a request message, it assigns a unique identifier to the request that is different from those for all other currently outstanding requests (for example, requests that do not yet have replies). When the receiver processes the request, it saves the identifier and adds the request's identifier to the reply.

When the caller processes the reply, it uses the request identifier to correlate the request with the reply. This is called a correlation identifier because of the way the caller uses the identifier to correlate each reply with the request.

A correlation ID is usually put in the header of a message. The ID is not a part of the command or data the caller is trying to communicate to the receiver.

The receiver saves the ID from the request and adds it to the reply for the caller's benefit. Since the message body is the content being transmitted between the two systems and the ID is not a part of that, the ID is added to the header.

In the request message, the ID can be stored as a correlation ID property or simply a message ID property. When used as a correlation ID, this can cause confusion about which message is the request and which is the reply. If a request has a message ID but

no correlation ID, then a reply has a correlation ID that is the same as the request's message ID.

The correlation ID format used internally by Oracle Service Bus is compatible with WebSphere MQ and works with target services that are using MQ native interfaces.

The outbound transport handles asynchronous request-response messages. That is, the message flow, except for the `$outbound` transport specific data, does not distinguish between JMS request-response and HTTP request-response.

When you define a JMS request-response business or proxy service, you must first choose a design pattern. To do this, select the **Is Response Required** option for a JMS proxy service or a **Response Queues** option for a JMS business service, then select one of the following correlation patterns on the JMS Transport Configuration page:

- JMS Correlation ID—the default pattern
- JMS Message ID

For information about creating JMS proxy and business services, see [Section 2.3, "Working with Proxy Services"](#) and [Section 2.2, "Working with Business Services."](#)

The following sections discuss these patterns. To compare the two patterns, see [Section 30.8.4, "Comparison of Message ID and Correlation ID Patterns."](#)

30.8.2 JMS Message ID Pattern

When you create a business service using the JMS Message ID pattern, you can configure the responses to go to a single URI, or, for failover support, you can configure a response queue for each request URI on each Managed Server in the Oracle Service Bus cluster. These queues must be collocated with the request queues for the service. The proxy service uses this information to set the `JMSReplyTo` property when invoking the business service so that the response is processed by the Managed Server that issued the request.

When you define a proxy service using the JMS Message ID pattern, you need not define the `ResponseURI` because the proxy service replies to the queue specified in the `JMSReplyTo` property. However, you can enter the JNDI name of the JMS connection factory for the response message.

Note: By default, the connection factory of the request message is used. This is useful when you use a non-XA connection factory for JMS responses, but have an XA connection factory for the request.

For the deployment descriptors to be set appropriately for XA-capable resources, you must set the XA attribute on the referenced connection factory before creating a proxy service.

The invoked service must copy the message ID from the request (the value of the JMS header field `JMSMessageID`) to the correlation ID of the response (setting the JMS header field `JMSCorrelationID`). In addition, the invoked service must reply to the queue specified in the `JMSReplyTo` header field.

If you choose the JMS Message ID Pattern, the response arrives at the appropriate managed node.

A JMS proxy service using this pattern can be used in a cluster without further configuration. A JMS business service is available in a cluster. However, when a Managed Server is added to the cluster, all the business services become invalid. To

correct this, ensure that the number of response queues equals the number of Managed Servers that specify the JMS Message ID correlation pattern in the Oracle Service Bus cluster.

Note: The JMS Message ID correlation pattern is not supported when a proxy service invokes another proxy service.

30.8.3 JMS Correlation ID Pattern

When you design a business service in Java, make sure that you set the value of JMS Correlation ID on the response to the value of JMS Correlation ID on the request before sending the JMS response to a queue.

You can obtain the JMS Correlation ID when you receive a message using:

```
String getJMSCorrelationID()
```

The above method returns correlation ID values that provide specific message IDs or application-specific string values.

To set the JMS Correlation ID when you send a message:

```
void setJMSCorrelationID(String correlationID)
```

30.8.4 Comparison of Message ID and Correlation ID Patterns

The JMS request-response patterns differ in the following ways:

- The method by which the response is correlated with the request
- The choice of the response queue

The differences between these two patterns are summarized in [Table 30–1](#).

Table 30–1 Differences Between Message ID and Correlation ID Patterns

JMS Pattern Name	Response Queue	CorrelationID
Correlation ID Pattern	All responses go to the same fixed queue(s).	The server copies the request Correlation ID to the response Correlation ID.
Message ID Pattern	The responses dynamically go to the queue indicated by the <code>JMSReplyTo</code> property.	The server copies the request Message ID to the response Correlation ID.

When the Correlation ID pattern is used, the service that is invoked replies to the queue that corresponds to the request URI. The response always arrives on the same queue and the client has no control over the queue to which the response arrives. For example, if 10 clients send a message to request URI "A", they all get the response to the queue that corresponds with request URI "A". Therefore, clients must filter the messages in the response queue to select the ones that pertain to them. Filtering criteria are configured in the request message Correlation ID property, and the server is configured to echo this to the response Correlation ID property.

In the case of Message ID pattern, the client's `JMSReplyTo` property tells the server where the response should be sent. This queue is specific to the client's server and hence responses to different clients will go to different queues. The server sets the JMS Correlation ID of the response to the JMS Message ID of the request.

Correlation by MessageID is commonly used by many IBM MQ applications as well as JMS applications and is the standard method to correlate request and response.

If you have multiple Oracle WebLogic client domains invoking a target Oracle WebLogic domain using JMS request-response with the Message ID pattern, you can set up both the request and response queues as SAF queues. However, this is not possible with the Correlation ID pattern that uses a single queue for all the responses for a given request URI.

The Correlation ID pattern has two major advantages:

- The response queue configuration is simple and it need not change every time a new Managed Server is added to the cluster.
- Correlation ID can also be used in cases where a proxy service in the domain needs to invoke another proxy service in the same domain.

30.8.5 Interoperating with JAX-RPC Over JMS

The Oracle Service Bus development environment lets you create JAX-RPC Web services that use the JMS transport, in addition to HTTP-HTTPS. These JMS transport JAX-RPC Web services use a JMS queue as the mechanism for retrieving and returning values associated with operations. You can use the JMS Message ID pattern to invoke a JMS transport JAX-RPC Web service.

You can also invoke a JMS Request-Response Oracle Service Bus proxy service from a JAX-RPC static stub, which the Oracle WebLogic clientgen Ant task generates.

This section includes the following topics:

- [Section 30.8.5.1, "Invoking a JAX-RPC Web Service Using the JMS Message ID Pattern"](#)
- [Section 30.8.5.2, "Invoking a JMS Request-Response Proxy Service from a JAX-RPC Client"](#)

30.8.5.1 Invoking a JAX-RPC Web Service Using the JMS Message ID Pattern

To invoke a JMS transport JAX-RPC Web service using the JMS Message ID pattern, complete the following steps:

1. Create a JMS Request-Response Oracle Service Bus business service that uses the JMS Message ID pattern to invoke the JMS transport JAX-RPC Web service. For more information, see [Section 3.1.15, "JMS Transport Configuration Page \(Business Services\)."](#)

This business service uses JMS transport. The JMS queue JNDI name portion of the end point URI must be the same as the queue attribute specified in the `@WLJmsTransport` annotation of the JMS transport JAX-RPC Web service. For example:

```
jms://localhost:7001/AJMSConnectionFactoryJNDIName/JmsTransportServiceRequestQueue
```

The JNDI name of the JMS queue (or queues) assigned to the Destination field, in the Target - Destinations area, must be associated with a JMS server targeted at the Oracle WebLogic Server name that is displayed in the Target field.

2. Create an Oracle Service Bus proxy service that contains a Routing (or Service Callout) action to the JMS Request/Response business service that you created in step 1.

The Request Actions area of the Routing action must contain a Set Transport Headers for the Outbound Request action. When you configure the Transport Headers action, you must add two JMS headers for the Outbound Request action. For detailed instructions about how to configure a Transport Headers action, see [Section 2.4.32, "Adding and Configuring Transport Headers Actions in Message Flows."](#)

In brief:

- a. Configure a Transport Headers Action by selecting Other in the Add Header field and entering a URI in the field provided.
- b. Select Set Header to <Expression> and create the expression by entering a concatenation of the values specified for the `contextPath` and `serviceUri` attributes (in the `@WLJmsTransport` annotation of the JMS transport JAX-RPC Web Service), preceded by a forward-slash. For example, you have the following `@WLJmsTransport` annotation:

```
@WLJmsTransport (
  contextPath="transports",
  serviceUri="JmsTransportService",
  portName="JmsTransportPort",
  queue="JmsTransportServiceRequestQueue"
)
```

You would enter the following expression in the XQuery Text input area when you configure the Transport Headers:

```
/transports/JmsTransportService
```

- c. To specify the second JMS Header, select Other in the Add Header field again, and enter `_wls_mimehdrContent_Type` in the associated field.
- d. Select Set Header to <Expression> and enter `text/xml; charset=UTF-8` in the XQuery Text input area.

30.8.5.2 Invoking a JMS Request-Response Proxy Service from a JAX-RPC Client

For a scenario in which a JAX-RPC Oracle WebLogic Server client invokes a proxy service, you must set the `_wls_mimehdrContent_Type` JMS header for the proxy service's inbound response.

You must specify the header when you issue the response to the incoming JMS Message ID Pattern request.

For example, for the scenario in which you have a JAX-RPC client calling an Oracle Service Bus proxy service, which subsequently calls an Oracle WebLogic Server Web service, the route node configuration is as follows:

For the Request Pipeline

1. Set the transport header for Web service context 'URI' (for example: `interop/AllocJmsDocLit`).
2. Set the transport header for `_wls_mimehdrContent_Type` with `text/xml; charset=UTF-8`.
3. Select Outbound request from the Set Transport headers menu items.
4. Enable `Pass all Headers through Pipeline`.

For the Response Pipeline

1. Add an empty transport header and select Inbound response from the Set Transport headers menu.
2. Enable `Pass all Headers through Pipeline`.

Note: For detailed instructions about how to configure a Transport Headers action, see [Section 2.4.32, "Adding and Configuring Transport Headers Actions in Message Flows."](#)

30.8.6 JMS Message ID Pattern Examples

The following examples describe the different methods by which the JMS Message ID pattern can be used.

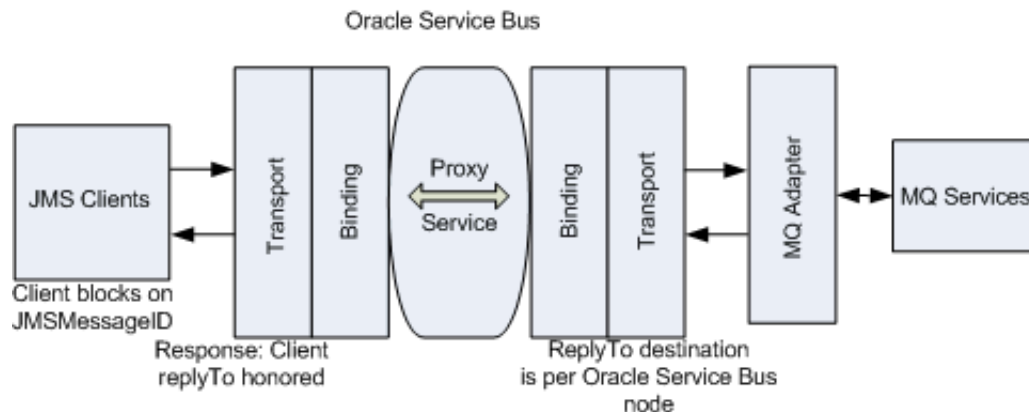
- [Section 30.8.6.1, "Example 1: An MQ Service Uses a JMS Message ID to Correlate the Request-Response Message"](#)
- [Section 30.8.6.2, "Example 2: A JAX-RPC Client with Oracle Service Bus Proxy Service"](#)
- [Section 30.8.6.3, "Example 3: Oracle Service Bus as a Client of an Oracle WebLogic Server JAX-RPC Request/Response Service"](#)

30.8.6.1 Example 1: An MQ Service Uses a JMS Message ID to Correlate the Request-Response Message

In [Figure 30–1](#), the server that hosts the MQ service in the request-response communication echoes the request message ID to the response correlation ID, and sends the response to the `replyTo` queue. The response travels back and is correlated using the JMS MessageID. The Oracle Service Bus `replyTo` destination is set, one per Oracle Service Bus node in a cluster, when the business service is configured. A JMS or MQ native client can also invoke a JMS request-reply proxy service using the JMS Message ID pattern. The client needs to set the `replyTo` property to the queue where it expects the response.

The key to supporting this use case is that JMS Message ID is expected to correlate the request-response message. You also need to create as many MQ series outbound response queues as there are cluster servers.

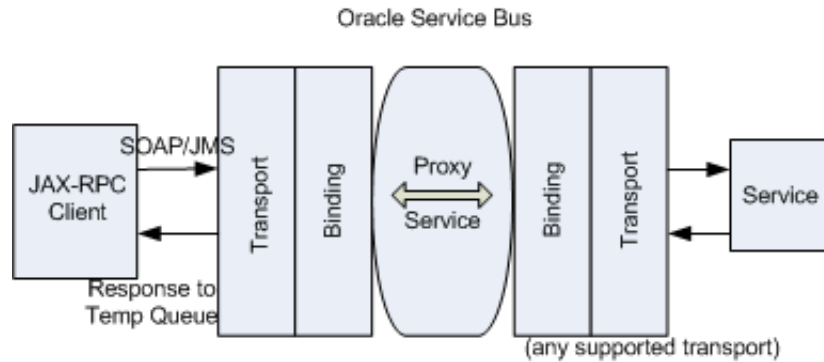
Figure 30–1 MQ Service Uses a JMS Message ID to Correlate the Request/Response Message



30.8.6.2 Example 2: A JAX-RPC Client with Oracle Service Bus Proxy Service

Figure 30–2 represents a JAX-RPC client sending a message to an Oracle Service Bus proxy service, that is, the JAX-RPC inbound case. The JAX-RPC stack employs a temporary queue to receive the response. The Oracle Service Bus JMS transport honors this temporary queue during run time.

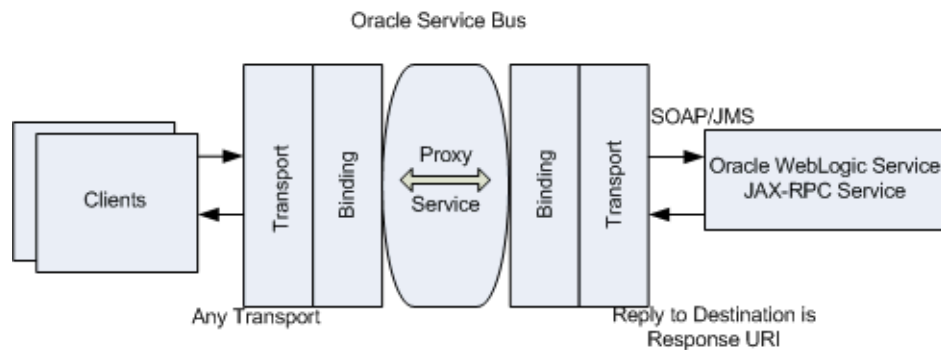
Figure 30–2 JAX-RPC Client with Oracle Service Bus Proxy Service



30.8.6.3 Example 3: Oracle Service Bus as a Client of an Oracle WebLogic Server JAX-RPC Request/Response Service

Figure 30–3 represents the JAX-RPC outbound case or the interoperability of an Oracle WebLogic Server JAX-RPC request/response service with an Oracle Service Bus proxy service.

Figure 30–3 Oracle Service Bus as a Client of an Oracle WebLogic Server JAX-RPC Request/Response Service



Note: When a proxy service in one Oracle WebLogic Server domain needs to send a message to a proxy service in a second domain, the message must first be routed to a pass-through business service in domain 1. JMS Store and Forward between domain 1 and domain 2 forwards the inbound request message to the proxy service in domain 2. When you use JMS request/response, you can choose to forward the inbound response message from domain 2 to domain 1 using JMS Store and Forward as well. In the latter case, exported inbound request and imported inbound response queues must be configured in domain 2 for the proxy service in domain 2. Pay close attention to the JMS Store and Forward configuration.

30.9 Using the JMS Transport

You can select JMS as the transport protocol for proxy and business services of any service type (except Transport Typed Service type for business service). The proxy services and business services can be configured to use the JMS transport as described in:

- [Section 30.9.1, "Configuring Proxy Services using JMS Transport Protocol"](#)
- [Section 30.9.3, "Configuring Business Services using JMS Transport Protocol"](#)

For more information, see [Section 2.2, "Working with Business Services"](#) and [Section 2.3, "Working with Proxy Services."](#) For information about error handling for business services, see [Section 30.9.4, "Error Handling."](#)

When you configure a proxy service, you can use a Transport Header action to set the header values in messages. For more information, see [Section 30.9.2, "Transport Headers."](#)

For information about interoperability of Oracle Service Bus with WebSphere MQ, see [Chapter 33, "MQ Transport."](#)

In Oracle Service Bus, you can use the default dispatch policy when you configure a service or, optionally, configure custom work managers in WLS.

30.9.1 Configuring Proxy Services using JMS Transport Protocol

You can select the JMS transport protocol when you configure any type of proxy service and the endpoint URI is of the form:

```
jms://<host:port[,host:port]*/connection_factory/jndi_destination>
```

where

- `host` is the name of the system that hosts the service.
- `port` is the port number at which the connection is made.
- `[, host:port]*` indicates that you can configure multiple hosts with corresponding ports.
- `connection_factory`: The name of the JNDI Connection Factory. For more information on how to define a connection factory queue, see "Configure resources for JMS system modules" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*.
- `jndi_destination`: The name of the JNDI destination.

To target a JMS destination to multiple servers, use the following format of the URI:

```
jms://host1:port,host2:port/connection_factory/jndi_destination
```

where `connection_factory` is the name of the connection factory queue. For more information on how to define a connection factory queue, see "Configure resources for JMS system modules" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*.

To configure a proxy service using JMS transport protocol, see [Section 3.1.16, "JMS Transport Configuration Page \(Proxy Services\)."](#)

30.9.2 Transport Headers

The various headers related to the JMS transport are listed in [Table 30–2](#). All the headers except the Unit of Order header are common to both outbound requests and inbound response.

Table 30–2 JMS Transport Headers

Header	Description
JMSMessageID	The JMSMessageID header field contains a value that uniquely identifies the message sent by a provider.
JMSDeliveryMode	The JMSDeliveryMode header field contains the delivery mode specified when the message was sent.
JMSExpiration	This header contains the expiration time of the message that is calculated as the sum of the time-to-live value specified on the send method and the current GMT value.
JMSPriority	The JMSPriority header field contains the priority of the message. When a message is sent, this field is ignored. After the send operation is complete, the field holds the value that is specified by the method sending the message.
JMSType	The JMSType header field contains the message type identifier that is specified by a client when a message is sent.
JMSCorrelation ID	This is used to link one message with another. For example to link a request message with a response message.
JMSXAppID	This is one of the JMS defined properties that specifies the identity of the application that sends the message. This is set by the JMS provider
JMSXGroupID	This one of the properties defined by JMS that specifies the group id of the message group to which the message belongs. This is set by the client
JMSXGroupSeq	This one of the properties defined by JMS that specifies the sequence of the message inside the message group.
JMS_IBM_Format	This contains the nature of the application data. For more information, see http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.xms.doc/props/prp_jms_ibm_format.html .
JMS_IBM_Report_Exception	Request exception reports. This also specifies how much of the application data from the message must be retained in the report message. For more information, see http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.xms.doc/props/prp_jms_ibm_rep_excpt.html .
JMS_IBM_Report_Expiration	Request expiration reports. This also specifies how much of the application data from the message must be retained in the report message. For more information, see http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rlmx/index.jsp?topic=/com.ibm.websphere.xms.610.doc/props/prp_jms_ibm_rep_exprn.html .
JMS_IBM_Report_COA	Request a confirm on arrival reports. This also specifies how much of the application data from the message must be retained in the report message. For more information, see http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.xms.doc/props/prp_jms_ibm_rep_coa.html .
JMS_IBM_Report_COD	Request a confirm on delivery reports. This also specifies how much of the application data from the message must be retained in the report message. For more information, see http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rlmx/index.jsp?topic=/com.ibm.websphere.xms.610.doc/props/prp_jms_ibm_rep_cod.html .

Table 30–2 (Cont.) JMS Transport Headers

Header	Description
JMS_IBM_Report_PAN	Request a positive action notification reports. For more information, see http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.xml.doc/props/prp_jms_ibm_rep_pan.html .
JMS_IBM_Report_NAN	Request a positive action notification reports. For more information, see http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.xml.doc/props/prp_jms_ibm_rep_nan.html .
JMS_IBM_Report_Pass_Msg_ID	Request that the message identifier of any report or reply message is the same as that of the original message. For more information, see http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.xml.doc/props/prp_jms_ibm_rep_pmid.html .
JMS_IBM_Report_Pass_Correl_ID	Request that the correlation identifier of any report or reply message is the same as that of the original message. For more information, see http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/topic/com.ibm.websphere.xml.doc/props/prp_jms_ibm_rep_pcid.html .
JMS_IBM_Report_Discard_Msg	Request that the message is discarded if it cannot be delivered to its intended destination. For more information, see http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.xml.doc/props/prp_jms_ibm_rep_dis.html .
JMS_IBM_MsgType	The type of a message. For more information, see http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.xml.doc/props/prp_jms_ibm_msgtype.html .
JMS_IBM_Feedback	This indicates the nature of a report message. For more information, see http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.xml.doc/props/prp_jms_ibm_feedback.html .
JMS_IBM_Last_Msg_In_Group	Indicates whether the message is the last message in a message group. For more information, see http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.websphere.xml.doc/props/prp_jms_ibm_last_mig.html .
JMS_BEA_UnitOfOrder	This header is valid only for the Inbound Response.

30.9.2.1 Configuring Transport Headers

You can configure the transport headers for both inbound and outbound requests in the Message Flow. For information about the transport headers related to the JMS transport, see [Section 30.9.2, "Transport Headers."](#)

In the pipeline, use a Transport Header action to set the header values in messages. For information about adding transport header actions, see "Adding Transport Header Actions" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

Note: For information about the limitations for JMS transport headers, see "Understanding How the Run Time Uses the Transport Settings in the Test Console" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* and "Limitations to Transport Header Values You Specify in Transport Header Actions" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

30.9.3 Configuring Business Services using JMS Transport Protocol

You can select the JMS transport protocol when you configure any type of business service and the endpoint URI is of the form:

```
jms://<host:port[,host:port]*/connection_factory/jndi_destination >
```

where

- `host` is the name of the system that hosts the service.
- `port` is the port number at which the connection is made.
- `[,host:port]*` indicates that you can configure multiple hosts with corresponding ports.
- `connection_factory`: The name of the JNDI Connection Factory. For more information on how to define a connection factory queue, see "Configure resources for JMS system modules" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*.
- `jndi_destination`: is the name of the JNDI destination.

To target a target a JMS destination to multiple servers, use the following format of the URI: `jms://host1:port,host2:port/connection_factory/jndi_destination`

where `connection_factory` is name of the connection factory queue. For more information on how to define a connection factory queue, see "Configure resources for JMS system modules" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*.

When you register a JMS business service, you must manually edit the URI from the WSDL file when adding it to the service definition. The URI format is as follows:

```
jms://<host>:<port>/connection_factory/jndi_destination
```

To configure a business service using the JMS transport protocol, see [Section 3.1.15, "JMS Transport Configuration Page \(Business Services\)."](#)

Note: Configuring a JMS request-response application with response correlation by JMS ID for a business service you must:

- Creating UDQs and local queues targeted to one chosen JMS server.
 - Disable the default targeting for the UDQs that deploys the UDQ on the cluster and creates the member queues on all JMS.
-
-

30.9.4 Error Handling

You can configure JMS-transport business services to handle application and communications errors as follows:

- Application errors

You can specify whether or not to retry business service endpoint URIs when application errors occur. For more information, see [Retry Application Errors](#) option in [Section 4.2.4, "Business Service Transport Configuration Page."](#)

An application error occurs when for a JMS business service configured with request/response, the `System.getProperty("com.bea.wli.sb.transports.jms.ErrorPropertyName", "SERVER_ERROR")` property is `true` in the response message. In this scenario, the JMS transport provider calls the error method with the `TRANSPORT_ERROR_APPLICATION` error code.

- Communication errors

You can configure business service URIs to be taken offline when such communication errors occur. When you configure the operational settings for the business service, you can enable the business service endpoint URIs to be taken offline after the specified retry interval. For more information, see "Configuring Operational Settings for Business Services" and "Viewing Business Services Endpoint URIs Metrics" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

Connection errors occur when any JMS exceptions or naming exceptions occur during any of the following activities:

- Looking up a JMS connection factory.
- Creating a JMS connection from a JMS connection factory.
- Creating a JMS session using a connection object.
- Looking up of a JMS destination.
- Creating a sender from the session object.
- Sending a JMS message using the sender and destination object.

This chapter provides information about the Oracle Service Bus local transport. It includes the following topics:

- [Section 31.1, "Introduction"](#)
- [Section 31.2, "Features and Characteristics of Local Transport Proxy Services"](#)
- [Section 31.3, "Message Handling for Local Transport Proxy Services"](#)
- [Section 31.4, "Usage of Local Transport Proxy Services"](#)
- [Section 31.5, "Limitations"](#)

31.1 Introduction

Commonly, service bus architectures include complex message flows in which messages are routed through multiple proxy services that are organized into larger multiple proxy service flows. Individual proxy services in these multiple proxy service scenarios route, publish, or call out to the next proxy service in the flow.

The reason for a multiple proxy services design is to support modularity and compartmentalization of the various components of the end-to-end message flow.

The individual proxy services in a multiple proxy service flows need to:

- Communicate efficiently and securely.
- Allow transactions and transactional behavior to be propagated
- Allow security context to be propagated so that the identity can be propagated end-to-end. The security context propagation also allows the client of the first proxy service in a multiple service flow to be authorized by the proxy services that are subsequently invoked in the flow—thus supporting fine-grained access control generic headers in the local transport.
- Using the local transport for proxy services ensures support for these capabilities.

31.2 Features and Characteristics of Local Transport Proxy Services

Local transport-based proxy services can only be invoked by other proxy service, not by other clients. The invocation is optimized by Oracle Service Bus. Local proxy services do not have an URI. However, there are no constraints on the service and interface types supported by local transport proxy services. The one exception is that SAML is only supported in a pass through scenario.

If the quality of service (QoS) for an invoking proxy service is defined as Exactly Once, the transaction of that service is propagated to the local transport proxy service.

In other words, the invoked local transport proxy service inherits the transactional behavior of the invoking proxy service. A proxy service can authenticate at the transport level or the message level. If it is enabled, the effective client is the message-level authenticated client. If the message-level authenticated client is not enabled, then the transport-level authenticated client is the effective client (if that is enabled). If neither the message-level nor the transport-level authenticated client is enabled, the anonymous client becomes the effective client.

When a proxy service invokes a local transport proxy service, the effective client of the invoking proxy service becomes the transport-level client of the invoked local proxy service. A local transport proxy service can authorize this client for access with an access control policy. In this way, it is possible to propagate the client of the first proxy service to all the subsequent proxy services in the overall end-to-end message flow.

Local transport proxy services support user-defined transport headers. Consider a scenario in which a proxy service uses the HTTP transport; it routes to a local proxy service and the HTTP proxy service passes headers to the local proxy service using the Transport Header action. In this scenario, if the HTTP proxy service received the Content-Type header, this header is available as a user header in the local transport and is therefore accessible through the standard user header, instead of as a typed transport header.

You can invoke a local transport proxy service from the Oracle Service Bus test browser. Metrics are collected for a local transport proxy service in the same way as they are any other service.

31.3 Message Handling for Local Transport Proxy Services

You can specify message handling options for local transport proxy services.

Specifically, you can enable local transport proxy services to decode and parse inbound messages in MTOM/XOP format and to send responses using the MTOM/XOP format, when appropriate. SOAP Message Transmission Optimization Mechanism (MTOM) is a method of sending binary data to and from Web services. MTOM uses XML-binary Optimized Packaging (XOP) to transfer the binary data.

Similarly, when local transport proxy services are chained through an HTTP transport proxy service, the local transport proxy services inherit the setting of whether attachments are paged to disk and processed in a streaming fashion (without buffering the attachment contents in memory).

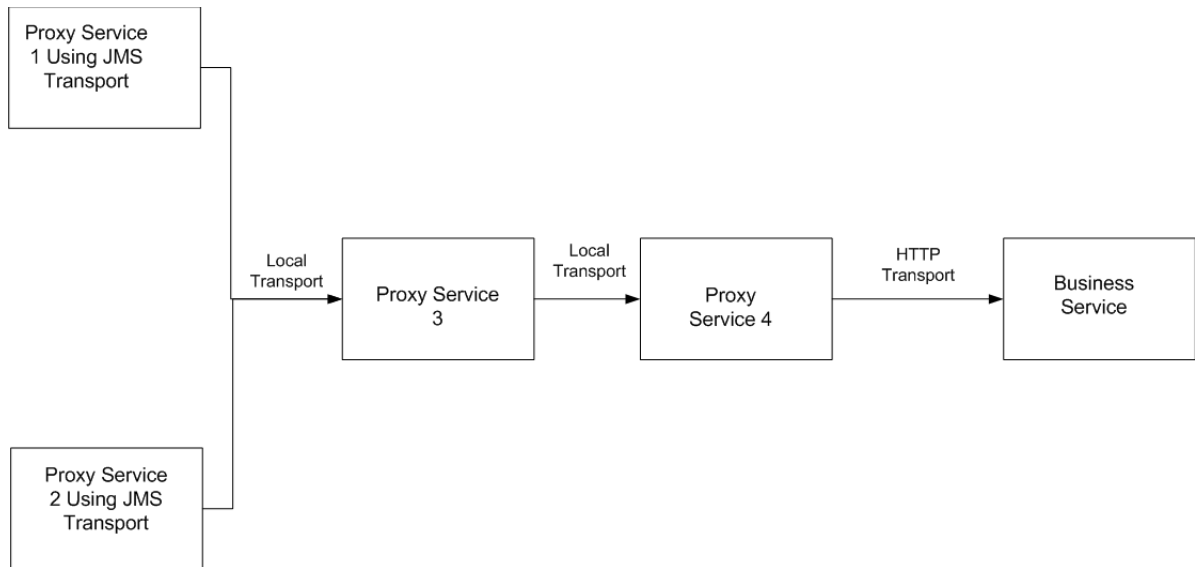
That is if an outer HTTP transport proxy service is configured to page attachments to disk, the chained local transport proxy services also process attachments in this fashion. This enables the proxy service to process large attachments robustly and efficiently.

For information about configuring message handling for proxy services, see [Section 4.3.4, "Proxy Service Message Handling Configuration Page."](#)

31.4 Usage of Local Transport Proxy Services

A common scenario that can be supported using local transport proxy services is one in which a proxy service needs to be invoked using different transports. This can be achieved by putting a set of front-end proxy services (one service per transport) in front of a local transport proxy service in the path of the message flow.

These front-end proxy services simply route messages to the local transport proxy service. The following figure illustrates this scenario.

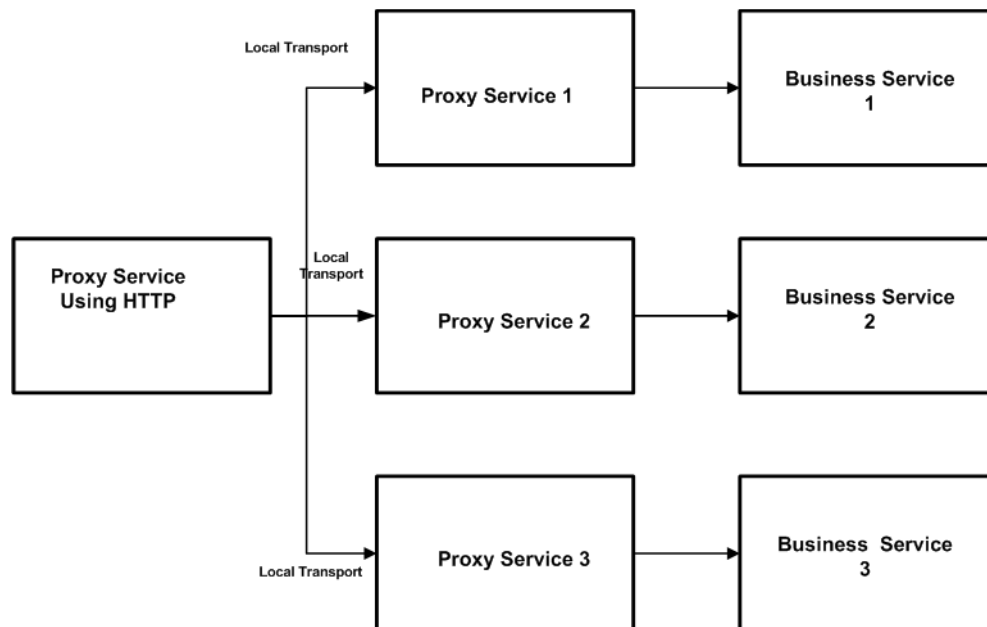
Figure 31-1 Using Local Transport to Implement Convergence

Another common scenario is one in which an Any SOAP or XML type proxy service acts as a front-end to different enterprise systems. This front-end proxy service can receive messages in a variety of formats and uses a technique common to all these messages (for example, a WS-Addressing SOAP header) to route the messages to an appropriate local transport proxy service.

In this scenario, the front-end proxy service is acting as a generic router with little knowledge of the enterprise systems or the message formats and semantics. To further abstract the knowledge of the routing rules at design time, the front-end proxy service can use dynamic routing to route messages to the local transport proxy services. For an example of how dynamic routing is used in the proxy services, see "Using Dynamic Routing" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

Each of the local transport proxy services to which messages are routed from the front-end service in turn acts as a front-end proxy service for a specific business service. The local transport proxy services are aware of the message format required by the business services to which they route.

In this scenario, these local transport proxy services act as functional proxy services. The roles of a functional proxy services are to enforce access control for invoking a particular business service, and to perform any transformation of the messages required to invoke the target business service correctly. The following figure illustrates this scenario.

Figure 31–2 Using Local Transport to Access Multiple Business Services

31.5 Limitations

The limitations of the local transport are:

- You can invoke the proxy service using the local transport only from another proxy service.
- The proxy services using the local transport cannot be published to the UDDI.
- A local transport proxy service cannot process inbound WS-Security SAML tokens.

WS Transport

The Web Services Reliable Messaging (WSRM) specification describes a protocol that allows messages to be delivered reliably between distributed applications even if a software, system, or network failure occurs.

WS-ReliableMessaging is a specification co-developed by IBM, Oracle, Microsoft and TIBCO Systems. This specification is not the same as the WS-Reliability (WSR), which is a competing specification developed by OASIS.

WSRM functionality is available in Oracle Service Bus as the WS transport. Oracle Service Bus supports the specification submitted in February 2005. For more information about the specification, see Web Services Reliable Messaging Protocol (WS-ReliableMessaging) at <http://schemas.xmlsoap.org/ws/2005/02/rm/>.

The WS transport implements both inbound and outbound requests for services derived from SOAP 1.1 and SOAP 1.2 based WSDLs with WSRM policy. However, the WSRM policy can be a part of the WSDL or can be attached to the service. In addition, security policies can also be declared in the WSDL or can be associated with a WSDL-based service. When you configure WSDL-based services with WSRM policies using the Oracle Service Bus Console, you must choose the WS transport for the service. Oracle Service Bus checks for the WSRM policy when you save the service configuration and throws a validation error if WSRM policies are not declared for the WSDL associated with the service.

The following are the key features of the WS Transport:

- One-way and request/response message patterns. For more information, see [Section 32.1.1, "Messaging Patterns."](#)
- Exactly-once transfer between WS transport and other transports (JMS, SB, and Tuxedo transports) that support XA transactions.
- HTTPS with basic authentication, and with client-certificate authentication (two-way SSL) but without client authentication,. For more information, see [Section 32.2, "Authentication and Authorization of Services."](#)
- Retaining WSRM security configuration while importing resources. For more information, see [Section 32.5, "Importing and Exporting Resources."](#)
- Assignment of transport-level access control policy to a WS proxy service in Oracle Service Bus Console. Only an administrator can assign this policy. For more information, see [Section 32.3.4, "Assigning Transport Access Control to Proxy Services."](#)
- WS-Addressing specification submitted in August 2004. For more information, see Web Services Addressing (WS-Addressing) at <http://www.w3.org/Submission/ws-addressing/>.

- WS-I Basic Profile compliance. For more information, see [Section 32.1.4, "Web Services Interoperability."](#)
- Quality of Service (QoS) in Oracle Service Bus for WS proxy service is always set as Exactly Once. For more information, see "Quality of Service" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

You can set the QoS only in the RM policy file using the `<beapolicy:QoS>` element. This element has one attribute, `QoS`, which can take any of the following values:

- `AtMostOnce`
- `AtLeastOnce`
- `ExactlyOnce`
- `OnOrder`

Note: QoS for WS transport is different from QoS for Oracle Service Bus.

- You can associate only SOAP 1.1 and SOAP 1.2 based WSDLs with WSRM policy with a proxy or business service. For more information, see [Section 32.3.3, "Configuring Proxy Services to Use the WS Transport"](#) and [Section 32.3.6, "Configuring Business Services to Use the WS Transport."](#)

32.1 Supported Functionality

This section provides detailed information about functionality supported by the WS transport.

32.1.1 Messaging Patterns

WSRM supports both one-way and request/response messaging patterns. The WS transport does not support reliable response. While the request is always reliable, the response is not sent reliably.

For business services, sending a request to an external web service is asynchronous. Successful invocation implies that the message is given to the RM layer successfully and it will be delivered reliably. However, successful invocation does not mean that the message is sent to the endpoint and has successfully invoked the web service.

For the request/response messaging pattern, the response is received from the external web service for a request. In this case, the request and response paths have two different transactions and run in two different threads. The response pipeline is executed evenly for one-way messaging message pattern. For the one-way pattern, response pipeline invocation means that the message reliably reached the target destination and the web service invocation is complete.

32.1.2 Policies

A proxy service or business service that uses the WS transport must have a WS-Policy with RM assertions. This also implies that services that use any other transport must not have any WS-Policy with RM assertions. WS-Policy with RM assertions and WSSP v1.2 transport-level security assertions are supported for the WS transport.

However, WSSP v1.2 message-level security assertions and 9.X Oracle proprietary security assertions are not supported. RM assertions should only be bound at the service level and not at the operation or operation request/response levels.

Note: You must use only one RM assertion for a WS-Policy.

32.1.2.1 WS-Policy Configurations

WS-Policies can be configured in any one of the following two ways:

- WS-Policy configuration is specified as part of the WSDL associated with the service. The policies specified in the WSDL may be included in the WSDL or referred in the WSDL.
- WS-Policy is assigned to the service from the Oracle Service Bus Console.

Note: You can use only one of these methods to associate a security policy with the service. So, if you configure a policy using the Oracle Service Bus Console, any policies defined in the WSDL are ignored.

32.1.3 Streaming Content for Large Messages

The WS transport does not have streaming support for large messages because the underlying infrastructure (WLS JAX-RPC stack) uses a fully materialized payload. However, when you configure a proxy service for large message processing, the message is fully materialized into a Java object by the WS transport using the streaming optimization in Oracle Service Bus. During the proxy service configuration, you can specify if you want to stream content for large message processing by buffering content either in memory or to disk. For more information, see "Streaming body Content" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

32.1.4 Web Services Interoperability

The WS transport supports web services interoperability through WS-I Basic Profile. Currently, Oracle Service Bus proxy services do not follow all the WS-I Basic Profile restrictions. However, any services configured to use this transport strictly follow the WS-I Basic Profile specification. WS proxy services do not have a WS-I Compliance check in the service configuration and always follow WS-I Basic Profile. This is valid for SOAP1.1 WSDL bindings as WS-I Basic Profile applies only to SOAP 1.1.

32.2 Authentication and Authorization of Services

This section provides information about how proxy and business services are authenticated and authorized.

32.2.1 Proxy Service Authentication

WS proxy services support both basic and client-certificate (two-way SSL) authentication. When basic authentication is specified in the WS-Policy, all HTTP requests, including RM protocol messages to the WS proxy service must have a valid username and password.

Proxy service authentication is supported as follows:

- Outbound client-certificate authentication using SSL key-pair assigned to the service key provider referenced by the proxy service.
- Username-password identity propagation through a WS proxy service (with basic authentication) to any other outbound transport, or outbound WSS username token.
- Credential mapping between WS proxy service (with basic or two-way SSL authentication) and any other transport.
- Sending asynchronous responses from WS proxy service to a RM client through HTTP or HTTPS. The default protocol used by proxy and business services is HTTP.
- Asynchronous responses from a WS proxy service to an RM client connect to the `AcksTo` or `ReplyTo` endpoint references specified by the RM client. The RM client can use either HTTP or HTTPS URL. If the RM client uses HTTPS, the RM client can request a client certificate during the SSL handshake. The WS transport uses the SSL key-pair of the service key provider upon request.

32.2.2 Proxy Service Authorization

Administrators can assign a transport-level access control policy to a WS proxy service in Oracle Service Bus Console. As with all transports, this policy is enforced after the inbound transport provider passes the request message to the Oracle Service Bus binding layer before invoking the request pipeline. For more information, see [Section 32.3.4, "Assigning Transport Access Control to Proxy Services."](#)

32.2.3 Business Service Authentication

WS business services support basic authentication and client-certificate authentication. Outbound basic authentication is supported by means of a service account. Username/password identity propagation and credential mapping are provided by the service account. However, a static account can also be used for authentication. The service account can be static, pass-through or mapped. Pass-through authentication allows passing a username/password from the client request to the back-end RM service. Mapped service accounts allow credential mapping. Static service accounts are used when fixed credentials are required.

WS business services also support SSL client-certificate authentication (two-way SSL). The key-pair (private key and certificate) used for outbound two-way SSL is not configured on the WS business service, but on the service key provider referenced by the proxy service.

Routing a single message to a WS business service may involve multiple HTTP/S requests from the Oracle Service Bus server and back-end service. All such messages are subject to the authentication method configured in the WS business service. In other words, if the service is configured for basic authentication, all messages sent from Oracle Service Bus include the HTTP Authorization header with the username/password and if the message is configured for client-certificate authentication, Oracle Service Bus uses the key-pair to authenticate all messages.

32.3 Using the WS Transport

You can use the WS transport to reliably deliver messages in a distributed network.

The WSRM functionality is available as a transport only for SOAP 1.1 and SOAP 1.2 based WSDLs with WSRM policy. Ensure that the services are associated with a SOAP

1.1 or 1.2 WSDLs with RM-policy or that a RM-policy is attached to the services. You can configure the WS-Policy in a WSDL or assign it to a service. For more information, see [Section 32.3.1.1, "Configuring WS Policies."](#)

Prior to configuring proxy and business services to use the WS transport, ensure that the required WSDLs or WS-Policy files are available in your Oracle Service Bus domain. For more information, see [Section 32.3.1, "Adding Resources to an Oracle Service Bus Domain,"](#) [Section 32.3.3, "Configuring Proxy Services to Use the WS Transport,"](#) and [Section 32.3.6, "Configuring Business Services to Use the WS Transport."](#)

You can optionally configure an error queue for services and Oracle Service Bus delivers failed messages into the queue. The queue can be a distributed queue. Because this queue is not created automatically, you must create it prior to configuring the services. For more information, see [Section 32.3.2, "Configuring an Error Queue."](#)

In addition, you can also import and export resources using the Oracle Service Bus Console. For more information, see [Section 32.5, "Importing and Exporting Resources"](#) and [Section 32.6, "Importing and Publishing Services Using UDDI Registries."](#)

32.3.1 Adding Resources to an Oracle Service Bus Domain

You can add WSDLs, and custom WS-Policy files to the domain using the Oracle Service Bus Console.

32.3.1.1 Configuring WS Policies

The WS transport can be used only with SOAP WSDLs that have a WSRM policy. You can configure a WS-Policy in a WSDL or assign a WS-Policy to a Service from the Oracle Service Bus Console. For more information, see [Section 32.1.2, "Policies."](#)

When no RM police assertions are specified for the WSDL associated with a service (you configure a service using a WSDL with no policy), a validation message appears when you activate the session.

Figure 32–1 Conflicts – When no RM police assertions are specified for the WSDL

The screenshot shows a 'Conflicts' window with a 'Diagnostic Messages' section. It contains a table with one row of diagnostic information.

Name	Path	Resource Type	Messages
test_ps	Test	Proxy Service	<ul style="list-style-type: none"> [ALSB Kernel:398130]WSRM service Test/test_ps must have RM policy assertions in its web service policy

To resolve this conflict, you need to update the WSDL or attach the policy to the service. For more information, see [Section 32.3.1.2, "Attaching WS Policies to a Service"](#) and [Chapter 51, "Using WS-Policy in Oracle Service Bus Proxy and Business Services."](#)

32.3.1.2 Attaching WS Policies to a Service

To attach a WS-Policy file to a service:

1. Locate the proxy or business service and click on the name of the service.
The View a Proxy Service or Business Service page appears.
2. Click on the **Policies** tab. You can view the service policy configuration details.
3. Click on the **Custom Policy Bindings** radio button.
4. Expand the proxy service folder and click **Add**.
The Select WS-Policy page appears.
5. You can search for the required policy and select the policy from the list of predefined policies or custom policy resources and click **Submit**.
6. Click **Update**.

The selected policy is now attached to the proxy service or business service.

Note: When you attach a WS-Policy to a service, any policies defined in the WSDL associated with the service are ignored.

32.3.2 Configuring an Error Queue

By default, undelivered messages are discarded after the specified number of retries. However, you can optionally configure error queues for business services and Oracle Service Bus delivers messages that fail in the message flow into these queues.

You must configure a JMS queue for errors. Oracle recommends that you configure a error queue locally instead of a remote queue.

For business services, when response timeout occurs, the response pipeline is invoked with an error. If sequence expiration interval is reached, the message is placed in an error queue configured for the business service and the response pipeline is invoked with an error. However, if the response timeout has already occurred, the message is placed in the error queue, but the response pipeline is not invoked.

Note: For both one-way and request-response services, putting failed messages in the error queue is only a best effort.

32.3.3 Configuring Proxy Services to Use the WS Transport

Proxy services using the WS transport must be associated with WS-Policy with RM assertions. For more information, see [Section 32.1.2, "Policies."](#)

A proxy service receives the requests from clients and passes it to the message flow after the processing related to WSRM is done. The proxy service could also send the response back to the client after receiving it from the response pipeline. A proxy service using the WS transport can be invoked from any other proxy service and it follows the same behavior as it is invoked by an external client.

When an HTTP proxy server is configured (per WLS wsee stack), WS proxy services send asynchronous messages using the HTTP proxy server.

Proxy services based on WSDL with SOAP 1.2 binding support SOAP 1.2 messages only and throw a fault with version mismatch error for SOAP 1.1 messages. Similarly, proxy services based on WSDL with SOAP 1.1 binding support SOAP 1.1 messages only and throw a fault with version mismatch error for SOAP 1.2 messages.

When you create a proxy service from the Oracle Service Bus Console, select the transport protocol as ws in the Transport Configuration page.

Note: For more information about configuring proxy services, see [Chapter 2.3, "Working with Proxy Services."](#)

[Table 32–1](#) describes the fields you can specify to configure a proxy service to use the WS transport:

Table 32–1 *Fields Required to Configure a Proxy Service to Use the WS Transport*

Field	Description
Protocol	Select <code>ws</code> from the list of available protocols.
Endpoint URI	Endpoint configuration for a proxy service that uses the WS transport is similar to that of <code>http/s</code> proxy service configuration. Specify the URI in <code>/contextPath</code> format. Note: Make sure that the context path is unique for proxy services that use either HTTP or the WS transport.

Now, you must specify configuration details specific to the WS transport.

[Table 32–2](#) describes the dispatch policy and advanced options like the retry count and retry delay values you can specify to configure the WS transport for a proxy service.

Table 32–2 *Fields Required to Configure WS Transport for a Proxy Service*

Field	Description
Dispatch Policy	Select a dispatch policy for this endpoint. Dispatch policy refers to the instance of WLS Work Manager that you want to use for the service endpoint. For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> .
Retry Count	The number of times, the WSRM layer tries to deliver a message to the Oracle Service Bus message flow. The default is 3. If an unhandled exception occurs in the request flow of a proxy, the incoming WS Transport message is redelivered to the message flow up to the number of times specified by this value. This is important for reliably processing the WS transport messages. Note: When the message delivery fails, the current transaction is rolled back, but the message is not removed from the queue. The server tries to send the message until the message is successfully delivered or the retry limit is reached. When the retry limit is reached, that message is removed from the queue or moved to an error queue. The error queue can be a distributed queue and can be created from the Oracle WebLogic Server Administration Console. For more information, see Section 32.3.2, "Configuring an Error Queue."
Retry Delay	The duration that the server should wait before retrying to deliver the message. The default is 5 seconds.

For more information about configuring proxy services using the WS transport, see WS Transport Configuration Page in [Chapter 2.3, "Working with Proxy Services."](#)

32.3.4 Assigning Transport Access Control to Proxy Services

Administrators can assign a transport-level access control policy to a WS Proxy Service in the Oracle Service Bus Console. As with all transports, this policy is enforced after the inbound transport provider passes the request message to the Oracle Service Bus binding layer before invoking the request pipeline.

Transport-level access control policies are managed within Oracle Service Bus sessions. When the session is activated, the access policy is stored in an Authorization Provider. At runtime, the binding layer calls the security framework authorization APIs, which in turn call the authorization provider.

To determine the access control of the proxy service resources at runtime, administrators can add one or more policy conditions. For example, a basic policy might simply name the Operator user. At runtime, the security framework interprets this policy as "only an Operator can access the proxy service resources." For more information, see [Section 32.3.4.1, "Adding Policy Conditions."](#)

Caution: Proxy services configured in the Oracle Service Bus Console to use the WS transport can also be viewed in the Oracle WebLogic Server Administration Console. Administrators can assign an access control policy from the Oracle WebLogic Server Administration Console and the Oracle Service Bus Console. However, policies assigned from the Oracle WebLogic Server Administration Console will have no effect and are not evaluated at runtime. Only access control policies assigned in the Oracle Service Bus Console are enforced.

To assign transport access control to a proxy service:

1. Locate the proxy service and click the proxy service name.
2. In the View a Proxy Service page, click the **Security** tab.

Figure 32–2 Security tab for a Proxy Service

Configuration Details	Operational Settings	SLA Alert Rules	Policies	Security
General Configuration				
Service Key Provider	<input type="text"/>	<input type="button" value="Browse..."/>		
Access Control				
Transport Access Control	✖ test_ps			
Custom Authentication				
<input type="button" value="Back"/>		<input type="button" value="Update"/>		<input type="button" value="Reset"/>

3. Click the name of the proxy service.

You can set the transport-level policy of the proxy service in this page.

Figure 32–3 Transport-Level Policy

Proxy Service Name	Test/test_ps
<input type="button" value="Save"/>	
<p>This page is used to edit the transport-level security policy of an ws proxy service.</p>	
<p>Providers</p> <p>These are the authorization providers an administrator can select from.</p> <p>Authorization Providers <input style="border: 1px solid blue;" type="button" value="XACMLAuthorizer"/></p>	
<p>Policy Conditions</p> <p>These are the conditions which determine the access control to your Proxy Service resources.</p>	
<input type="button" value="Add Conditions"/> <input type="button" value="Combine"/> <input type="button" value="Uncombine"/> <input type="button" value="Move Up"/> <input type="button" value="Move Down"/> <input type="button" value="Remove"/> <input type="button" value="Negate"/>	
<input type="checkbox"/> Access occurs before : 12/1/07	
<input style="border: 1px solid blue;" type="button" value="And"/>	
<input type="checkbox"/> Combination	
<input type="checkbox"/> User : John	
<input style="border: 1px solid blue;" type="button" value="Or"/>	
<input type="checkbox"/> Group : xyz	
<input style="border: 1px solid blue;" type="button" value="And"/>	
<input type="checkbox"/> NOT Role : abc	
<input type="button" value="Add Conditions"/> <input type="button" value="Combine"/> <input type="button" value="Uncombine"/> <input type="button" value="Move Up"/> <input type="button" value="Move Down"/> <input type="button" value="Remove"/> <input type="button" value="Negate"/>	
<input type="button" value="Save"/>	
<p>Overwritten Policy</p> <p>Group : everyone</p>	

4. Click **Add Conditions**. The Choose a Predicate page appears.
5. Select the required predicate and click **Next**. The Edit Arguments page appears.
6. Enter the parameters and values associated with the predicate to define the policy conditions.

Administrators can:

- Create complex conditions and combine them using the logical operators AND and OR (which is an inclusive OR). A combination of conditions can be uncombined later, if required.
- Negate any condition, which would make sure that the inverse of the specified policy condition is applied.
- Specify the order in which the policy conditions are executed.
- Remove existing policy conditions.
- Apply the AND and OR operators between multiple conditions.

At runtime, the entire collection of conditions must be true for the proxy service.

For more information about the parameters to be specified for each predicate, see [Section 32.3.4.1, "Adding Policy Conditions."](#)

7. Click **Save**.
8. Click **Back** and click **Update**.

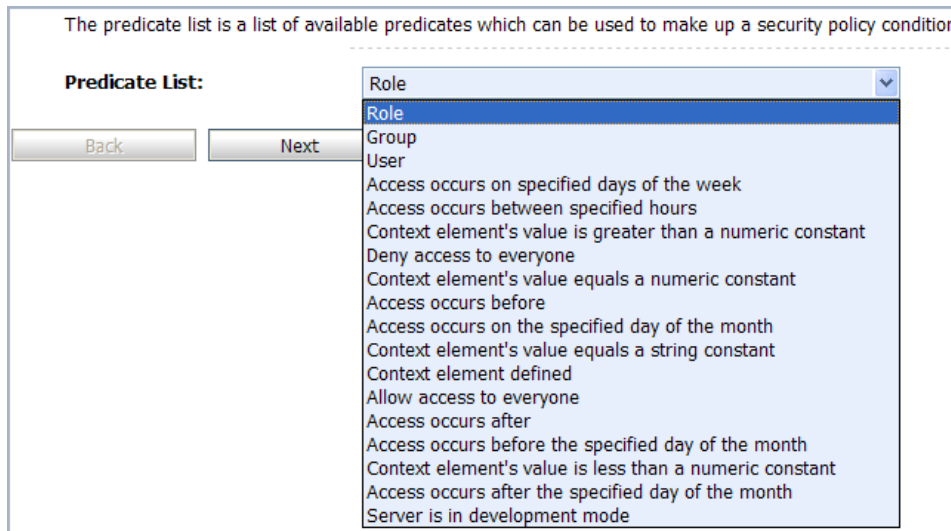
The specified access control policy conditions are now associated with the proxy service and applied at runtime.

32.3.4.1 Adding Policy Conditions

The policy conditions set by an administrator control access to the proxy service resources. When you add a condition to a policy statement, you can use any of the existing predicates or policy conditions. Each predicate is a predefined statement that can be used to define the security policy statement. For each predicate, you need to edit the arguments that are associated with that predicate.

Click **Add Conditions** to view the list of predicates.

Figure 32–4 List of Predicates



These predicates include:

- Basic policy conditions that include adding specified users, groups, or roles, allowing or denying access to everyone, and so on.
- Date and time based policy conditions, which are used to grant access to the resources based on the date or time you specify.
- Context element policy conditions, which are used to create policies based on the value of HTTP Servlet Request attributes, HTTP Session attributes, and EJB method parameters.

To Add or Remove a role to the policy condition:

1. Select **Role** from the **Predicate List**.
2. Click **Next**. The Edit Arguments page appears.
3. Specify the role and click **Add**.

You can add one or more roles to this policy condition. If you add multiple roles, the condition evaluates as true if the user is a member of ANY of the roles associated with this policy condition.

Note: To remove any role, select the role in the Remove list and click **Remove**.

4. Click Finish.

The condition is added to the policy statement and displayed on the policy conditions page.

To add a group to the policy condition:

1. Select **Group** from the **Predicate List**.
2. Click **Next**. The Edit Arguments page appears.
3. Specify the group name and click **Add**.

You can add one or more groups to this policy condition. If you add multiple groups, the condition evaluates as true if the user is a member of ANY of the groups associated with this policy condition.

Note: To remove any group, select the group in the Remove list and click **Remove**.

4. Click Finish.

The condition is added to the policy statement and displayed on the policy conditions page.

To add a user to the policy condition:

1. Select **User** from the **Predicate List**.
2. Click **Next**. The Edit Arguments page appears.
3. Specify the user name and click **Add**.

You can add one or more users to this role.

Note: To remove any user, select the user in the Remove list and click **Remove**.

4. Click Finish.

The condition is added to the policy statement and displayed on the policy conditions page.

To enable access to proxy service resources only on specified days of the week:

1. Select **Access occurs on specified days of the week** from the **Predicate List**.
2. Click **Next**. The Edit Arguments page appears.
3. Enter the day of the week.
4. Enter the GMT offset. If the time zone in your location is ahead of GMT, enter GMT + hh:mm and if time zone in your location is behind GMT, enter GMT -hh:mm.
5. Click **Finish**.

The condition is added to the policy statement and displayed on the policy conditions page.

To enable access to proxy service resources only between a specified time:

1. Select **Access occurs between specified hours** from the **Predicate List**.

2. Click **Next**. The Edit Arguments page appears.
3. Enter the start time and the end time between which access to resources to enabled. Enter time in hh:mm AM | PM format.
4. Enter the GMT offset. If the time zone in your location is ahead of GMT, enter GMT + hh:mm and if time zone in your location is behind GMT, enter GMT -hh:mm.
5. Click **Finish**.

The condition is added to the policy statement and displayed on the policy conditions page.

To enable access to proxy service resources only when the context element's value is greater than a numeric constant:

1. Select **Context element's value is greater than a numeric constant** from the **Predicate List**.
2. Click **Next**. The Edit Arguments page appears.
3. Enter the name of the context element.
4. Enter a numeric value. Access to resources is enabled only when the specified context element's value is greater than this number.
5. Click **Finish**.

The condition is added to the policy statement and displayed on the policy conditions page.

To deny access to everyone:

1. Select **Deny Access to everyone** from the **Predicate List**.
2. Click **Finish**.

All users and groups are denied access to the proxy service resources.

To enable access to proxy service resources only when the context element's value is equal to a numeric constant:

1. Select **Context element's value equals a numeric constant** from the **Predicate List**.
2. Click **Next**. The Edit Arguments page appears.
3. Enter the name of the context element.
4. Enter a numeric value. Access to proxy service resources is enabled only when the specified context element's value is equal to this number.
5. Click **Finish**.

The condition is added to the policy statement and displayed on the policy conditions page.

To enable access to proxy service resources only before a specified date:

1. Select **Access occurs before** from the **Predicate List**.
2. Click **Next**. The Edit Arguments page appears.
3. Enter the date before which access to proxy service resources is enabled. Enter date in m/d/yy or m/d/yy hh:mm:ss AM | PM format.
4. Enter the GMT offset. If the time zone in your location is ahead of GMT, enter GMT + hh:mm and if time zone in your location is behind GMT, enter GMT -hh:mm.

5. Click Finish.

The condition is added to the policy statement and displayed on the policy conditions page.

To enable access to proxy service resources only on the specified day of the month:

1. Select **Access occurs on the specified day of the month** from the **Predicate List**.
2. Click **Next**. The Edit Arguments page appears.
3. Enter the day of the month on which access to proxy service resources is enabled.

Access to proxy service resources is enabled only after an ordinal day in the month. Enter the ordinal number of the day within the current month with values in the range from -31 to 31. Negative values count back from the end of the month, so the last day of the month is specified as -1. 0 indicates the day before the first day of the month.

4. Enter the GMT offset. If the time zone in your location is ahead of GMT, enter GMT + hh:mm and if time zone in your location is behind GMT, enter GMT -hh:mm.
5. Click **Finish**.

The condition is added to the policy statement and displayed on the policy conditions page.

To enable access to proxy service resources only when the context element's value is equal to a string constant:

1. Select **Context element's value equals a string constant** from the **Predicate List**.
2. Click **Next**. The Edit Arguments page appears.
3. Enter the name of the context element.
4. Enter a string value. Access to proxy service resources is enabled only when the specified context element's value is equal to this string value.
5. Click **Finish**.

The condition is added to the policy statement and displayed on the policy conditions page.

To enable access to proxy service resources only when the defined context element is available:

1. Select **Context element defined** from the **Predicate List**.
2. Click **Finish**.

Access to proxy service resources is enabled only when the specified context element exists.

To allow access to everyone:

1. Select **Allow Access to everyone** from the **Predicate List**.
2. Click **Finish**.

Access to proxy service resources is enabled to all users and groups.

To enable access to proxy service resources only after a specified date:

1. Select **Allow access after** from the **Predicate List**.
2. Click **Next**. The Edit Arguments page appears.

3. Enter the date after which access to proxy service resources is enabled. Enter date in m/d/yy or m/d/yy hh:mm:ss AM|PM format.
4. Enter the GMT offset. If the time zone in your location is ahead of GMT, enter GMT + hh:mm and if time zone in your location is behind GMT, enter GMT -hh:mm.
5. Click **Finish**.

The condition is added to the policy statement and displayed on the policy conditions page.

To enable access to proxy service resources only before the specified day of the month:

1. Select **Access occurs before the specified day of the month** from the **Predicate List**.
2. Click **Next**. The Edit Arguments page appears.
3. Enter the day of the month before which access to proxy service resources is enabled.

Access to proxy service resources is enabled only before an ordinal day in the month. Enter the ordinal number of the day within the current month with values in the range from -31 to 31. Negative values count back from the end of the month, so the last day of the month is specified as -1. 0 indicates the day before the first day of the month.

4. Enter the GMT offset. If the time zone in your location is ahead of GMT, enter GMT + hh:mm and if time zone in your location is behind GMT, enter GMT -hh:mm.
5. Click **Finish**.

The condition is added to the policy statement and displayed on the policy conditions page.

To enable access to proxy service resources only when the context element's value is less than a numeric constant:

1. Select **Context element's value is less than a numeric constant** from the **Predicate List**.
2. Click **Next**. The Edit Arguments page appears.
3. Enter the name of the context element.
4. Enter a numeric value. Access to proxy service resources is enabled only when the specified context element's value is less than this number.
5. Click **Finish**.

The condition is added to the policy statement and displayed on the policy conditions page.

To enable access to proxy service resources only after the specified day of the month:

1. Select **Access occurs after the specified day of the month** from the **Predicate List**.
2. Click **Next**. The Edit Arguments page appears.
3. Enter the day of the month after which access to proxy service resources is enabled.

Access to proxy service resources is enabled only after an ordinal day in the month. Enter the ordinal number of the day within the current month with values

in the range from -31 to 31. Negative values count back from the end of the month, so the last day of the month is specified as -1. 0 indicates the day before the first day of the month.

4. Enter the GMT offset. If the time zone in your location is ahead of GMT, enter GMT + hh:mm and if time zone in your location is behind GMT, enter GMT -hh:mm.
5. Click **Finish**.

The condition is added to the policy statement and displayed on the policy conditions page.

To enable access to proxy service resources only when Server is running in development mode:

1. Select **Server is in development mode** from the **Predicate List**.
2. Click **Finish**.

Users and groups can access the proxy service resources only when the server is running in development mode.

32.3.5 Routing the WS Transport Through an HTTP Proxy Server

When an HTTP proxy server is configured, WS business services send outbound messages using the HTTP proxy server. For information about specifying the HTTP proxy server details in your client application, see "Using a Proxy Server When Invoking a Web Service" in "Invoking Web Services" in *Oracle Fusion Middleware Getting Started With JAX-RPC Web Services for Oracle WebLogic Server*.

32.3.6 Configuring Business Services to Use the WS Transport

Business services using the WS transport must be associated with WS-Policy with RM assertions. For more information, see [Section 32.1.2, "Policies."](#) A business service acts as a client for invoking an external reliable web service. It sends a request to the service and the response is received by an application deployed by Oracle Service Bus, which invokes the response path.

When you create a business service from the Oracle Service Bus Console based on the WSDL resource, select the transport protocol as `ws` in the Transport Configuration page.

Note: For more information about configuring business services, see [Section 2.2, "Working with Business Services."](#)

[Table 32–3](#) describes the fields you must specify to configure a business service to use the WS transport, specify the following fields:

Table 32–3 *Configuring a Business Service to use WS Transport*

Field	Description
Protocol	Select <code>ws</code> from the list of available protocols.

Table 32–3 (Cont.) Configuring a Business Service to use WS Transport

Field	Description
Load Balancing Algorithm	Specify the load balancing algorithm as any one of the following values: <ul style="list-style-type: none"> ■ Round-robin ■ Random ■ Random-weighted ■ None
Endpoint URI	Point to the location of the web service. Endpoint configuration for a business service that uses the WS transport is similar to that of http/https configuration. Specify the URI in <code>http://host:port number/name</code> or <code>https://host:port number/name</code> format. Business services can have multiple endpoint URIs pointing to different reliable web services.
Retry Count	In case of delivery failure when sending outbound requests, specify the number of times to retry individual URL endpoints. The number in this field indicates the total number of times URIs are retried (not the number of URIs in the list).
Retry Iteration Interval	Specify the number of seconds the system should pause between retries of all the endpoint URIs in the list.
Retry Application Errors	Select Yes or No. In case of delivery failure when sending outbound requests, specify whether or not to retry endpoint URIs based on application errors (for example, a SOAP fault). For more information, see Section 32.4, "Error Handling."

To configure the WS transport for a business service, specify the values as described in [Table 32–4](#):

Table 32–4 Configuring WS Transport for Business Service

Field	Description
Response Timeout	If the response does not come in the defined interval after sending a request, response pipeline is invoked with an error saying that service is timed out. A value of 0 implies that there would be no response timeout.
Service Account	Specify a service account. Note: This is only applicable if the WS business service has a WS-Policy that requires basic authentication For more information, see Section 2.1.16, "Creating Service Account Resources."
Queue Error Messages	Select this option if you want to send failed requests to an error queue. The following fields are available only if you select this option.
Error Queue URI	Error queue used for failed requests in the business service. Specify the URI in <code>jms://host:port/conn-factory-jndi-name/queue-jndi-name</code> format. Note: This queue can be a distributed queue. It is not created automatically so, make sure that a valid queue is available. For more information, see Section 32.3.2, "Configuring an Error Queue."

Table 32–4 (Cont.) Configuring WS Transport for Business Service

Field	Description
JMS Error Queue Service Account	The service account to be used for connecting to the JMS error queue.
Use SSL for Error Queue	Select this option to use SSL for connecting to a JMS error queue.

For more information about configuring business services using the WS transport, [Section 2.2, "Working with Business Services."](#)

32.4 Error Handling

You can configure the WS transport-based business services to handle application errors by specifying whether or not to retry business service endpoint URIs when application errors occur. See [Section 4.2.4, "Business Service Transport Configuration Page."](#)

An application error occurs when a WS transport-based business service receives a SOAP fault as a response and the BEA-380001 error code is generated.

Note: When a response timeout or sequence timeout occurs for a request to a business service, the Oracle Service Bus server tries to send the message to the next URI based on the load balancing algorithm. This behavior does not depend on the `Retry Application Errors` option.

32.5 Importing and Exporting Resources

When a resource exists in an Oracle Service Bus domain, you can preserve the security and policy configuration details while importing that resource to Oracle Service Bus by selecting the `Preserve Security and Policy Configuration` option. When you select this option, the values in the existing resource are preserved when you import them, even if the security and policy configurations have been updated in the resource.

For information about importing resources, see [Section 2.1.14, "Importing Resources."](#)

32.6 Importing and Publishing Services Using UDDI Registries

When a proxy service is published to an UDDI registry, the service is converted into WS business service with the WSDL. If present, the authentication configuration is also exported to UDDI.

When a WSDL-based business service with WSRM policy is imported from an UDDI registry to Oracle Service Bus, the service is imported as a WS business service that is automatically configured to use the WS transport. For more information, see [Section 32.1.2, "Policies."](#)

For more information, see "UDDI" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

Oracle Service Bus supports access to IBM WebSphere MQ using the MQ Transport. This transport supports both inbound and outbound connectivity. MQ proxy services can receive messages from WebSphere MQ and MQ business services can route messages to WebSphere MQ.

33.1 Key Features

The following key features are supported for the MQ transport:

- Inbound and outbound connectivity. MQ proxy services can receive messages from WebSphere MQ and MQ business services can route messages to WebSphere MQ.
- Access to WebSphere MQ. For more information, see [Section 33.8.1, "Adding MQ Client Libraries to Your Environment."](#)
- Sending and receiving messages of Any XML, Binary, XML, Text and MFL types.
- Processing of all the MQMD (MQ message descriptor) headers. A message descriptor is an attribute representing a property of the message that is either being sent or received.

For a list of MQ headers that can be configured for outbound response and inbound requests, see [Section 33.8.5, "Transport Headers."](#)

- TCP/IP and bindings mode for connecting to queue managers.
Use the bindings mode to connect to WebSphere MQ that is located on the same physical machine as Oracle Service Bus. Use TCP/IP to connect to WebSphere MQ that does not reside on the same machine as Oracle Service Bus.
- One-way and two-way SSL on inbound and outbound transport.
SSL is supported only when TCP/IP is used to connect to WebSphere MQ from Oracle Service Bus.

33.2 Advantages of Using the MQ Transport

Because the WebSphere MQ for Java APIs do not support XA transactions, Oracle Service Bus does not support XA transactions for the MQ transport. If XA is required, use the JMS transport with WebSphere MQ JMS interface. For more information, see [Chapter 33.9, "Using the WebSphere JMS MQ Interface."](#)

Using the MQ transport has the following advantages over using the WebSphere MQ JMS interface:

- Faster connection to WebSphere MQ through the MQ transport than through the WebSphere MQ JMS interface.
- Ability to read and generate MQ messages. Using the JMS interface, it is not possible to set certain headers.
- Support for sending and receiving MQ receipt messages.
- No explicit binding of MQ Connection Factory and MQ Queue to WebLogic's JNDI is required.
- Configuration of resources like a JMS provider, outside of Oracle Service Bus, is not required.
- Performance improvement because messages are sent directly using the transport instead of channeling them through the JMS transport.

33.3 Supported Service Types

The MQ transport is available for the Message Service and XML Service service types.

For more information, see [Section 4.3, "Proxy Service Configuration"](#) and [Section 4.2, "Business Service Configuration."](#)

33.4 Messaging Patterns

The Native MQ transport supports one-way and request-response messaging patterns for both inbound and outbound connectivity. By default, one-way messaging is supported. A proxy or business service supports request-response messaging when you set the `is-response-required` option while configuring the service.

The inbound and outbound transports support the asynchronous request-response pattern using `messageID` or `correlationID` for correlation between the request and the response. You can set the response correlation pattern during service configuration. For more information, see "CorrelationID" and "MessageID" in [Section 33.8.5, "Transport Headers."](#)

The outbound transport provides an option to auto-generate the correlation ID / messageID or use the one specified by you in the message flow. Select the `Auto-generate Correlation Value` option, to indicate that the correlation ID / message ID should be auto-generated by the transport. If the option is not selected, the value specified by you in the message flow is used.

The outbound transport also lets you use dynamic queues for response correlation if you use dynamic queues in your MQ implementation.

If the correlation value (`messageID` / `correlationID`) is not auto-generated and if the managed server goes down, the remaining response messages may never get removed when the server is restarted. Oracle recommends that the `Expiry` header on the request is configured to a finite value and that the `Report` header contains the `MQC.MQRO_PASS_DISCARD_AND_EXPIRY` option.

The `MQC.MQRO_PASS_DISCARD_AND_EXPIRY` option serves as a directive to the receiving client that the message descriptor of the reply should inherit the `Expiry` header value of the request message. This ensures that the response messages are removed by the MQ server after the configured expiry period has elapsed. When the correlation value is automatically generated, the Oracle Service Bus server is responsible for cleaning up any remaining response messages.

The MQ transport supports local transactions but not remote transactions.

For more information about configuring **Is Response Required**, **Response Correlation Pattern**, **Auto Generate Correlation Value** for a service, see [Section 33.8.3](#), "[Configuring Proxy Services to Use the MQ Transport](#)" and [Section 33.8.4](#), "[Configuring Business Services to Use the MQ Transport](#)."

33.5 Environment Values

Environment values are certain predefined fields in the configuration data whose values are very likely to change when you move your configuration from one domain to another (for example, from test to production). For information about updating environment values, see "Finding and Replacing Environment Values" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*. The following sections provide a list of environment values for the Native MQ transport services and the MQ connection resource.

Services based on the MQ transport support the `Work Manager` (Inbound and Outbound) environment value.

MQ connections are sharable resources that can be reused across multiple MQ proxy and business services. MQ Connection resources provide the connection parameters required for connecting to a MQ queue manager. The MQ connection resource supports the following environment and operational values:

- MQ Connection Host Name
- MQ Connection Port
- MQ Queue Manager Name
- MQ Channel Name
- MQ Connection Pool Size
- MQ Connection Timeout
- MQ Model Queue
- MQ Version

33.6 Quality of Service

When the inbound transport is MQ and the Quality of Service (QoS) on the outbound transport is *exactly-once*, the resultant QoS will be *at-least-once*. By default, the QoS on the outbound transport is *exactly-once*.

Note: You must create error handling logic (including any retry logic) in the pipeline error handler. For information about configuring error handling, see [Section 2.4.8](#), "[Adding and Configuring Error Handlers in Message Flows](#)."

When the outbound is request-response, the QoS is *at-least-once* only if the outbound transport is configured to support *exactly-once* QoS.

For more information about QoS in Oracle Service Bus messaging, see "Quality of Service" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

33.7 MQ Clusters and the MQ Transport

The cluster support in WebSphere MQ is that of store-and-forward messaging and not of load-balancing and fail over. The cluster queues in WebSphere MQ are created locally on one of the queue managers and shared with other cluster members that act as remote forwarders to the shared queue.

Requests from the MQ transport are load balanced by sending messages using the load balancing algorithm to the members of the cluster. However, the transport receives messages by accessing only the MQ server node that holds the reference to the local queue.

33.8 Using the MQ Transport

You can use the Oracle Service Bus MQ transport to enable MQ proxy services to get messages from WebSphere MQ and MQ business services to send messages to WebSphere MQ. This document describes how to add WebSphere MQ to your Oracle Service Bus environment, characteristics of the Oracle Service Bus MQ transport, and how to configure MQ proxy and business services.

This section contains the following topics:

- [Section 33.8.1, "Adding MQ Client Libraries to Your Environment"](#)
- [Section 33.8.2, "MQ Connection Resources"](#)
- [Section 33.8.3, "Configuring Proxy Services to Use the MQ Transport"](#)
- [Section 33.8.4, "Configuring Business Services to Use the MQ Transport"](#)
- [Section 33.8.5, "Transport Headers"](#)
- [Section 33.8.6, "Error Handling"](#)

33.8.1 Adding MQ Client Libraries to Your Environment

Oracle Service Bus is a client for WebSphere MQ, and although Oracle Service Bus supports run-time server compatibility for supported versions of WebSphere MQ, these MQ libraries are not bundled with the Oracle Service Bus installer. You need to ensure that the MQ client library, `com.ibm.mq.jar` version 6.0, is available in your environment.

For information about the system requirements for WebSphere MQ, see <http://www-306.ibm.com/software/integration/wmq/requirements/index.html>.

To add the MQ client libraries to your environment:

1. Stop the domain server.
2. From your WebSphere MQ installation, copy the `com.ibm.mq.jar` file to the `DOMAIN_HOME/domain_name/lib` directory.
3. Restart the domain server.

If you use bindings mode to connect MQ Queue Manager located on the same machine as Oracle Service Bus, add `<MQ_install_directory>/bin` and `<MQ_install_directory>/java/lib` to the PATH environment variable.

33.8.2 MQ Connection Resources

MQ connections are sharable resources that can be reused across multiple MQ proxy and business services. MQ proxy and business services must connect to a MQ queue manager before accessing the MQ queue. MQ connection resources provide the connection parameters required for connecting to a MQ queue manager.

Each MQ connection resource has a connection pool. Multiple business services and proxy services using the same queue manager share a connection pool. Any business or proxy service using a given MQ connection resource to connect to a given queue manager uses the same connection pool that was created for that MQ connection resource.

For information about managing MQ connection resources, see [Section 2.6.1, "Adding MQ Connections."](#)

For information about queue managers, see

<http://www.redbooks.ibm.com/redbooks/SG247128/wwhelp/wwhimpl/java/html/wwhelp.htm>.

33.8.2.1 Creating an MQ Connection Resource

Note: The following instructions on creating an MQ Connection resource pertain to the Oracle Service Bus Console. In Eclipse, using the Oracle Service Bus perspective, select **File > New > Custom Resource** to create the MQ Connection resource.

To create an MQ connection resource:

1. In the Project Explorer, select MQ Connection as the resource type. The Create a New MQ Connection Resource page appears.
2. Enter the name and description of the resource.

Note: Do not include spaces in the name.

3. Select the connection type as `tcp mode` or `bindings mode`.
Use `tcp mode` when the MQ Queue Manager is not available on the same machine as Oracle Service Bus. Use `bindings mode` to connect MQ Queue Manager located on the same machine as Oracle Service Bus.
4. If the connection type is `bindings mode`, enter the name of the MQ Queue Manager.
5. If the connection type is `tcp mode`:
 - a. Enter the host name of the MQ Queue Manager.
 - b. Enter the port number of the MQ Queue Manager Listener.
 - c. Enter the name of the MQ Queue Manager.
 - d. Enter the coded character set identifier (CCSID) to be used for connecting to the MQ Queue Manager for client connection mode.
 - e. Enter the Queue Manager server connection channel name.
 - f. Select the **SSL** option, if required, to use HTTPS protocol for sending the messages.

After you select this option, select the 2-way SSL Required option to enable 2-way SSL for both client-side and server-side authentication. Clear the 2-way SSL Required option for 1-way SSL for only server-side authentication.

Note: When you select 2-way SSL, you need to provide a reference to a service key provider. A service key provider contains Public Key Infrastructure (PKI) credentials that proxy services use for decrypting inbound SOAP messages and for outbound authentication and digital signatures. For more information about using service providers, see "Service Key Providers" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

- g.** Specify the static service account.

A service account provides a user name and password that proxy services and business services use for outbound authentication or authentication to a local or remote resource. For more information about using service accounts, see [Section 2.1.16, "Creating Service Account Resources."](#)

- 6.** Select the version of WebSphere MQ.
- 7.** Enter the size of the MQ connection pool. Default is 10.
- 8.** For MQ Connection Timeout, enter the time interval in seconds after which unused connections are destroyed. The default is 1800 seconds.
- 9.** Enter the Max Wait in seconds for the amount of time to wait for a connection to become available. If a connection is not made within that time interval, Oracle Service Bus throws an exception. The default is 3 seconds.
- 10.** Click **Save**.

The MQ connection resource is created. You can use it across proxy and business services that use the MQ transport.

33.8.3 Configuring Proxy Services to Use the MQ Transport

During service configuration, select either Message service or XML service as the service type in the General Configuration page. For more information, see [Section 33.3, "Supported Service Types"](#) and [Section 4.3, "Proxy Service Configuration."](#)

When you create an MQ service, select the transport protocol as `mq` and specify the Endpoint URI in the Transport Configuration page. Specify the Endpoint URI in `mq://local-queue-name?conn=mq-connection-resource-ref` format where `local-queue-name` is the name of the local queue configured on the MQ server `mq-connection-resource-ref` points to the location of the MQ connection resource

For example, if you create a MQ connection resource, `mqConnection` in the `defaultMQ` folder and the queue name is `testQueue`, the URI would be `mq://testQueue?conn=defaultMQ/mqConnection`

Note: The Endpoint URI cannot contain spaces, so do not create MQ Connection resources or projects/folders with spaces in the names.

Configure the MQ transport for a proxy service with values as described in the following table:

Table 33–1 MQ Proxy Service Configuration

Option	To create or edit...
Polling Interval	Enter a polling interval, in milliseconds. The default is 1000.
Is Response Required	Select this option to specify that a response is expected after an outbound message is sent.
Response Correlation Pattern	This option is available only when the Is Response Required check box is selected. Specify whether the response correlation pattern should be based on MessageID or CorrelationID .
MQ Response URI	This option is available only when the Is Response Required check box is selected. The destination to which the response should be published. Enter a response URI in the same format as the endpoint URI: mq://<local-queue-name>?conn=<mq-connection-resource-ref>
Response Message Type	This option is available only when the Is Response Required check box is selected. Select one of the following: <ul style="list-style-type: none"> ■ Bytes (for a stream of uninterpreted bytes) ■ Text (for text messages)
Client Response Timeout	This option is available only when the Is Response Required check box is selected. Enter the number of seconds to wait for a response before dropping the connection.
Dispatch Policy	Select a dispatch policy for this endpoint. Dispatch policy refers to the instance of WLS Work Manager that you want to use for the service endpoint. For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i> .
Backout Threshold	Enter a value representing the number of times the pipeline should retry a message before redirecting the message to the queue specified in the Dead Letter URI field. If you do not specify a value for this field, the message is redirected to the dead letter queue without attempting any retries.
MQ Dead Letter URI	Enter the URI of the dead letter queue to which request messages should be redirected after attempting the number of retries specified in the Backout Threshold field. If you do not specify a value for this field, the message is discarded after retrying the number of times specified in the Backout Threshold field. The Dead Letter URI uses the same format as the EndPoint URI.

Table 33–1 (Cont.) MQ Proxy Service Configuration

Option	To create or edit...
Endpoint URI 'GET' options	<p data-bbox="651 260 1321 287">Enter the MQ GET message options from among the following:</p> <ul style="list-style-type: none"> <li data-bbox="651 302 1198 329">■ MQC.MQGMO_ACCEPT_TRUNCATED_MSG <li data-bbox="651 342 1149 369">■ MQC.MQGMO_ALL_MSGS_AVAILABLE <li data-bbox="651 382 1052 409">■ MQC.MQGMO_BROWSE_FIRST <li data-bbox="651 422 1052 449">■ MQC.MQGMO_BROWSE_NEXT <li data-bbox="651 462 1073 489">■ MQC.MQGMO_COMPLETE_MSG <li data-bbox="651 501 992 529">■ MQC.MQGMO_CONVERT <li data-bbox="651 541 1110 569">■ MQC.MQGMO_FAIL_IF_QUIESCING <li data-bbox="651 581 943 609">■ MQC.MQGMO_LOCK <li data-bbox="651 621 1081 648">■ MQC.MQGMO_LOGICAL_ORDER <li data-bbox="651 661 1154 688">■ MQC.MQGMO_MARK_BROWSE_CO_OP <li data-bbox="651 701 1146 728">■ MQC.MQGMO_MARK_SKIP_BACKOUT <li data-bbox="651 741 1065 768">■ MQC.MQGMO_NO_SYNCPOINT <li data-bbox="651 781 951 808">■ MQC.MQGMO_NONE <li data-bbox="651 821 987 848">■ MQC.MQGMO_NO_WAIT <li data-bbox="651 861 1016 888">■ MQC.MQGMO_SYNCPOINT <li data-bbox="651 900 1203 928">■ MQC.MQGMO_SYNCPOINT_IF_PERSISTENT <li data-bbox="651 940 984 968">■ MQC.MQGMO_UNLOCK <li data-bbox="651 980 1195 1008">■ MQC.MQGMO_UNMARK_BROWSE_CO_OP <li data-bbox="651 1020 1203 1047">■ MQC.MQGMO_UNMARK_BROWSE_HANDLE <li data-bbox="651 1060 1198 1087">■ MQC.MQGMO_UNMARKED_BROWSE_MSG <li data-bbox="651 1100 1003 1127">■ MQC.MQGMO_VERSION_1 <li data-bbox="651 1140 1003 1167">■ MQC.MQGMO_VERSION_2 <li data-bbox="651 1180 1003 1207">■ MQC.MQGMO_VERSION_3 <li data-bbox="651 1220 938 1247">■ MQC.MQGMO_WAIT <p data-bbox="651 1255 1300 1312">You can use either " " or "+" to separate multiple options. For example, you can specify the following:</p> <p data-bbox="651 1325 1354 1381">MQC.MQGMO_ACCEPT_TRUNCATED_MSG MQC.MQGMO_LOCK</p> <p data-bbox="651 1394 1360 1440">The MQ GET message options are applied when reading a message from the inbound queue.</p>
Process RFH2 Headers	<p data-bbox="651 1461 1321 1539">Select this option to parse WebSphere MQ RFH2 headers from a message payload and automatically generate an RFH2Headers transport header containing the RFH2 data.</p> <p data-bbox="651 1551 1321 1602">If you do not select this option, the payload is passed through as received.</p>

For more information about configuring proxy services using MQ transport, see MQ Transport Configuration Page in [Section 2.3, "Working with Proxy Services."](#)

33.8.4 Configuring Business Services to Use the MQ Transport

During service configuration, select either Message service or XML service as the service type in the General Configuration page. For more information, see [Section 4.2, "Business Service Configuration"](#).

When you create an MQ service, select the transport protocol as `mq` and specify the Endpoint URI in the Transport Configuration page. Specify the Endpoint URI in `mq://local-queue-name?conn=mq-connection-resource-ref` format where `local-queue-name` is the name of the local queue configured on the MQ server `mq-connection-resource-ref` points to the location of the MQ connection resource

For example, if you create a MQ connection resource, `mqConnection` in the `defaultMQ` folder and the queue name is `testQueue`, the URI would be `mq://testQueue?conn=defaultMQ/mqConnection`.

Note: The Endpoint URI cannot contain spaces, so do not create MQ Connection resources or projects/folders with spaces in the names.

To configure the MQ transport for a business service, specify the values as described in the following table:

Table 33–2 MQ Business Service Configuration

Option	To create or edit...
Message Type	Select one of the following: <ul style="list-style-type: none"> ■ Bytes (for a stream of uninterpreted bytes) ■ Text (for text messages)
Is Response Required	Select this option to specify that a response is expected after an outbound message is sent.
Polling Interval	This option is available only when the Is Response Required check box is selected. Enter a polling interval, in milliseconds. The default is 1000.
Response Correlation Pattern	This option is available only when the Is Response Required check box is selected. Specify whether the response correlation pattern should be based on: <ul style="list-style-type: none"> ■ MessageID ■ CorrelationID ■ Dynamic Queue – Select this option if your WebSphere MQ implementation uses dynamic queues for response correlation. The MQ transport supports only temporary dynamic queues.
Auto-generate Correlation Value	This option is available only when the Is Response Required check box is selected. Select this check box to automatically generate a CorrelationID or MessageID.
Model Queue	For Dynamic Queue Response Correlation Pattern only. Enter the name of the model queue used to generate the dynamic queue.

Table 33–2 (Cont.) MQ Business Service Configuration

Option	To create or edit...
MQ Response URI	<p>This option is available only when the Is Response Required option is selected.</p> <p>The destination to which the response should be published. Enter a response URI in the same format as the endpoint URI:</p> <pre>mq://local_queue_name?conn=mq_connection_resource</pre> <p>or, if you are using dynamic queues:</p> <pre>mq://dynamic_queue_prefix?conn=mq_connection_resource</pre> <p>The <i>dynamic_queue_prefix</i>, which is limited to 32 characters, is used to create the dynamic queue on the MQ server. The queue name becomes the prefix plus a unique ID. For example, if the <i>dynamic_queue_prefix</i> is <code>example</code>, the dynamic queue would be named something like <code>example123129083821</code>.</p> <p>You can also use an asterisk (*) as a wildcard in the dynamic queue response URI. For example:</p> <pre>mq://dynamic_queue_prefix?*conn=mq_connection_resource</pre> <pre>mq://dynamic_queue_prefix*</pre> <pre>mq://*</pre> <p>If you do not provide a <i>dynamic_queue_prefix</i> in the URI, the transport uses the dynamic queue name generated by the MQ server. If you do not provide an explicit <i>mq_connection_resource</i> in the URI (best practice), the transport uses the <i>mq_connection_resource</i> from the endpoint URI.</p> <p>For more detailed information, see "MQ Transport" in the <i>Oracle Fusion Middleware Developer's Guide for Oracle Service Bus</i> at http://www.oracle.com/pls/asp1111/lookup?id=OSBDV1117.</p>
Response Timeout	<p>This option is available only when the Is Response Required check box is selected.</p> <p>Enter the number of seconds to wait for a response before dropping the connection.</p>
Dispatch Policy	<p>Select a dispatch policy for this endpoint.</p> <p>Dispatch policy refers to the instance of WLS Work Manager that you want to use for the service endpoint.</p> <p>For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i>.</p>
Dynamic Queue Pooling	<p>For Dynamic Queue Response Correlation Pattern only. Select this option if you want the service to use pooled connections to dynamic queues.</p> <p>If you want to use a separate connection pool for dynamic queues, consider configuring a dedicated MQ Connection resource for the dynamic queues.</p> <p>Do not select this option if you want to create a new dynamic queue instance on each request (and destroy the queue after the response).</p>

Table 33–2 (Cont.) MQ Business Service Configuration

Option	To create or edit...
Endpoint URI 'PUT' options	<p data-bbox="732 260 1398 287">Enter the MQ PUT message options from among the following:</p> <ul data-bbox="732 300 1325 961" style="list-style-type: none"> ■ MQC.MQPMO_ALTERNATE_USER_AUTHORITY ■ MQC.MQPMO_DEFAULT_CONTEXT ■ MQC.MQPMO_FAIL_IF QUIESCING ■ MQC.MQPMO_LOGICAL_ORDER ■ MQC.MQPMO_NEW_CORREL_ID ■ MQC.MQPMO_NEW_MSG_ID ■ MQC.MQPMO_NO_CONTEXT ■ MQC.MQPMO_NO_SYNCPOINT ■ MQC.MQPMO_NONE ■ MQC.MQPMO_PASS_ALL_CONTEXT ■ MQC.MQPMO_PASS_IDENTITY_CONTEXT ■ MQC.MQPMO_RESOLVE_LOCAL_Q ■ MQC.MQPMO_SET_ALL_CONTEXT ■ MQC.MQPMO_SET_IDENTITY_CONTEXT ■ MQC.MQPMO_SYNCPOINT ■ MQC.MQPMO_VERSION_1 ■ MQC.MQPMO_VERSION_2 <p data-bbox="732 974 1378 1031">You can use either " " or "+" to separate multiple options. For example, you can specify the following:</p> <p data-bbox="732 1043 1446 1100">MQC.MQPMO_LOGICAL_ORDER MQC.MQPMO_NEW_MSG_ID</p> <p data-bbox="732 1113 1398 1163">The MQ PUT message options are applied when the message is placed in the outbound queue.</p>
MQ Unrecognized Response URI	<p data-bbox="732 1178 1435 1283">Enter the URI representing the queue to which unrecognized response messages should be sent. Note that this setting is enabled only when the Auto-generate Correlation Value check box is selected.</p> <p data-bbox="732 1295 1422 1352">If you do not specify a value for this field, unrecognized response messages are deleted.</p>
Process RFH2 Headers	<p data-bbox="732 1367 1409 1451">Select this option to parse WebSphere MQ RFH2 headers from a message payload and automatically generate an RFH2Headers transport header containing the RFH2 data.</p> <p data-bbox="732 1463 1409 1514">If you do not select this option, the payload is passed through as received.</p>

For more information about configuring business services using MQ transport, see MQ Transport Configuration Page in [Section 2.2, "Working with Business Services."](#)

33.8.5 Transport Headers

The various headers used by the MQ transport are listed in [Table 33–3](#). Most of the headers are common to both outbound requests and inbound response. The Reply To Queue Name, Reply To Queue Manager Name, User ID and Version headers can be edited only for the inbound response.

When you configure a proxy service, you can use a Transport Header action to set the header values in messages.

Table 33–3 Transport Headers

Header	Description	Inbound Response /Outbound Request
Accounting Token	<p>Accounting token is part of the identity context of the message.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both
Application ID Data	<p>Application ID data is part of the identity context of the message. This value can be used to provide additional information about the message or its originator.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both
Application Origin Data	<p>Data about the originating application. This value can be used by the application to provide additional information about the origin of the message.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both
Backout Count	<p>The number of times the message was returned by the MQ Queue, as part of a unit of work, and subsequently backed out.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both
Character Set	<p>The coded character set identifier of character data in the application message data.</p> <p>Inbound Transport Action: This field is used by the inbound transport to convert data in a specific representation. For request-response messaging, the characterSet header from the request message is copied to the response. When this header is not configured on the incoming request, default value of the MQC.MQCCSI_Q_MGR field is assumed.</p> <p>Outbound Transport Action: This header can be set in the pipeline for the outbound transport. If this header value is not set, the default MQC.MQCCSI_Q_MGR value is assumed.</p>	Both

Table 33–3 (Cont.) Transport Headers

Header	Description	Inbound Response /Outbound Request
Correlation ID	<p>The correlation-id of the message that should be retrieved.</p> <p>Inbound Transport Action: For correlationID-based response correlation pattern, the correlationID from the request is echoed on the response. The user can override the correlationID in the response pipeline.</p> <p>Outbound Transport Action: When the Auto-generate correlationID option is selected during service configuration, the outbound transport will automatically generate a correlationID and overwrite the correlationID from the transport header. If this value is not specified, the correlationID specified in the pipeline is used.</p> <p>For one-way messaging, the correlationID specified in the pipeline is used in the (outbound) request.</p>	Both
Encoding	<p>The representation used for numeric values in the application message data.</p> <p>Inbound Transport Action: The inbound transport uses this header to interpret the incoming message data. If this header is not configured in the response pipeline, the default value of <code>MQC.MQENC_NATIVE</code> is used.</p> <p>Outbound Transport Action: If this header is not set in the pipeline for the outbound transport, the default value of <code>MQC.MQENC_NATIVE</code> is used.</p>	Both
Expiry	<p>The expiry time (in tenths of a second) is set by the application that puts the message. After a message's expiry time has elapsed, it is eligible to be discarded by the queue manager.</p> <p>Inbound Transport Action: For request-response messaging, the inbound transport copies the expiry header of the request to the response.</p> <p>Outbound Transport Action: If the corresponding transport header is set in the pipeline, it is copied to the outbound request message.</p> <p>Note: The report header will always contain the <code>MQC.MQRO_PASS_DISCARD_AND_EXPIRY</code> option (in addition to others). This option is a directive to the receiving client that the expiry time of the original message should be copied to the report or reply message.</p>	Both
Feedback	<p>The nature of the feedback report. This value is used with a message of type <code>MQC.MQMT_REPORT</code> to indicate the nature of the report.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both
Format	<p>Format name of the message data. The format name is used by the sender of the message to indicate the nature of the data in the message to the receiver.</p> <p>Inbound Transport Action: When the field is set to <code>MQC.MQFMT_MD_EXTENSION</code>, the inbound transport will read the extended MQMD object.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both

Table 33–3 (Cont.) Transport Headers

Header	Description	Inbound Response /Outbound Request
Group ID	<p>The value that identifies the message group to which the physical message belongs.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both
Offset	<p>In a segmented message, offset of data in the physical message from the start of the logical message.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both
Original Length	<p>Original length of a segmented message.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both
Message Flags	<p>Flags that control the segmentation and status of a message.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both
Message ID	<p>ID of the message to be retrieved.</p> <p>Inbound Transport Action: If messageID is not specified in the response pipeline, the messageID header is set to MQC.MQMI_NONE.</p> <p>For messageID-based correlation, the inbound transport copies the messageID from the request to the correlationID header of the response. MessageID-based correlation is assumed when the report header contains the MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID option.</p> <p>Outbound Transport Action: When the Auto-generate messageID option is specified during service configuration, the outbound transport automatically generates the messageID and overwrites the messageID from the transport header. If this value is not specified, the messageID transport header is used.</p> <p>For one-way messaging, the messageID specified in the pipeline is used in the outbound request. If this value is not specified, the messageID is automatically generated by the transport.</p>	Both
Message Sequence Number	<p>Sequence number of a logical message within group.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both

Table 33–3 (Cont.) Transport Headers

Header	Description	Inbound Response /Outbound Request
Message Type	<p>Message type of the message.</p> <p>Inbound Transport Action: The inbound transport reads and processes messages of any type including MQC.MQMT_REQUEST, MQC.MQMT_DATAGRAM, MQC.MQMT_REPLY and MQC.MQMT_REPORT. The inbound transport does not generate report messages.</p> <p>Outbound Transport Action: The outbound transport generates messages of any type including MQC.MQMT_DATAGRAM, MQC.MQMT_REQUEST, MQC.MQMT_REPLY and MQC.MQMT_REPORT. When the messageType header is not configured in the pipeline, the transport generates messages of type MQC.MQMT_DATAGRAM when the messaging pattern is one-way and MQC.MQMT_REQUEST when the messaging pattern is request-reply.</p>	Both
Persistence	<p>The message persistence.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both
Priority	<p>Priority of the message</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both
Put Application Name	<p>The name of the application that put the message.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both
Put Application Type	<p>The type of the application that put the message.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both
Put Date Time	<p>The time and date when the message was put.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Both

Table 33–3 (Cont.) Transport Headers

Header	Description	Inbound Response /Outbound Request
Reply To Queue Name	<p>The name of the queue to which a reply should be sent.</p> <p>The application that issued the get request for the message can send MQC.MQFMT_REPLY and MQC.MQFMT_REPORT messages to this queue.</p> <p>Inbound Transport Action: The inbound transport uses the <code>replyToQueueName</code> as the response queue name when this field is set. If this values is not set, the queue name is derived from the default destination URI.</p> <p>Outbound Transport Action: In request/response message pattern, <code>replyToQueueName</code> set in the message flow is ignored. In one way message pattern, <code>replyToQueueName</code> set in the message flow is used in the outbound messages.</p>	Inbound Response
Reply To Queue Manager Name	<p>The name of the queue manager to which reply or report messages can be sent.</p> <p>Inbound Transport Action: In request/response message pattern, if the inbound message <code>replyToQueueManager</code> header value does not match the configured value for the queue manager in the response URI, the response message is dropped and a transport error is logged.</p> <p>Outbound Transport Action: In request/response message pattern, <code>replyToQueueManager</code> set in the message flow is ignored. In one way message pattern, <code>replyToQueueManager</code> set in the message flow is used in the outbound messages.</p>	Inbound Response
Report	<p>A report is a message about another message. This field enables the application sending the original message to specify which report messages are required, whether the application message data is to be included in them, and also how the message and correlation ID in the report or reply are to be set. It comprises one or more constants from the MQC class combined by means of the '+' or ' ' operators.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline. For request-response messaging, this header can be configured in the response pipeline.</p> <p>Outbound Transport Action: The transport always sets a combination of the following options in the report field.</p> <p>Set MQC.MQRO_COPY_MSG_ID_TO_CORREL_ID if messageID-based correlation pattern is used and MQC.MQRO_PASS_CORREL_ID if correlationID-based correlation pattern is used. Always set MQC.MQRO_PASS_DISCARD_AND_EXPIRY.</p> <p>Note: These options are set in addition to the options specified on the corresponding transport header in the pipeline.</p>	Both

Table 33–3 (Cont.) Transport Headers

Header	Description	Inbound Response /Outbound Request
User ID	<p>It is part of the identity of the message and identifies the user who originated the message.</p> <p>Inbound Transport Action: No explicit processing is done by the transport. The header is copied to the transport header in the pipeline.</p> <p>Outbound Transport Action: No explicit processing is done by the transport. If the corresponding transport header is set in the pipeline, it is copied to the request message.</p>	Inbound Response
Version	<p>The version number of the message descriptor.</p> <p>Inbound Transport Action: The inbound transport supports both version 1 and version 2 message descriptors.</p> <p>Outbound Transport Action: By default, the outbound transport generates version 2 headers. However, this field can be overridden in the pipeline.</p>	Inbound Response
RFH2Headers	<p>The RFH2 headers in the payload when the Process RFH2 Headers option is set in the transport configuration. The RFH2Headers header is a String.</p> <p>Inbound Transport Action: RFH2 headers are extracted from the MQ payload to construct the corresponding transport metadata header.</p> <p>Outbound Transport Action: RFH2Headers data are parsed to extract the RFH2 headers, which are inserted (along with the content length for each header) into the outbound MQ payload.</p>	Both

33.8.5.1 Configuring Transport Headers

You can configure the transport headers for both inbound and outbound requests in the Message Flow. For information about the transport headers related to the MQ transport, see [Section 33.8.5, "Transport Headers."](#)

In the message flow, use a Transport Header action to set the header values in messages. For more information, see [Section 2.4.32, "Adding and Configuring Transport Headers Actions in Message Flows."](#)

When the transport header is explicitly set in the message flow, this value overrides the header value except in the following scenarios:

- For the outbound request-response pattern, when the `Auto-generate Correlation Value` option is selected for a outbound request with a request-response message pattern, the correlation ID is always generated even if this value is set in the message flow.
- When the report header is set in the message flow, the combination of multiple directives associated with the report header are merged with the default directives.
- When the `replyToQueueManagerName` or `replyToQueueName` headers are set in the message flow for an outbound request with a request/response message pattern, these values are ignored. Instead, these transport header values are derived from the response URI configured for the business service.
- For the inbound response, when the value in the `replyToQueueManagerName` header does not match the queue manager name specified in the response URI, an error message is generated and the response message is not sent.

- The `replyToQueueName` set in the inbound request header overrides the value configured in the reply to URI for the proxy service.
- For a one-way business service, when the message type header is configured to be of type request in the message flow, the `replyToQueueName` header must be specified. If this value is not specified, an error is generated on the MQ server and the message is rolled back.

33.8.5.1.1 About RFH2 Headers RFH2Header headers can contain multiple `<RFH2Header>` blocks, each of which can contain multiple folders. For example:

```
<RFH2Header>
  <mcd><Msd>jms_bytes</Msd></mcd>
</RFH2Header>
<RFH2Header>
  <usr><clientId>DASHBOARD</clientId></usr>
</RFH2Header>
```

33.8.6 Error Handling

You can configure MQ-transport business services to handle application and communications errors as follows:

- Application errors – You can specify whether or not to retry business service endpoint URIs when application errors occur. For more information, see "Retry Application Errors" in [Section 4.2.4, "Business Service Transport Configuration Page."](#)
- Communication errors – You can configure business service URIs to be taken offline when communication errors occur. For more information, see "Enable business service endpoint URIs to be taken offline" and "Viewing Business Services Endpoint URIs Metrics" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

33.8.7 Limitations of the MQ Transport

The following are the limitations of the MQ transport:

- MQ transport does not support:
 - Remote XA for the MQ transport. The MQ transport does support local transactions. For more information, see [Section 33.9, "Using the WebSphere JMS MQ Interface"](#).

Defining MQ exits.
- You cannot call a request-response proxy service based on MQ proxy service:
 - From a proxy service that has been configured with a route action or dynamic routing and routing table actions).

Using the service callout action.
- You cannot call a proxy service with a service callout where the target is a request-response proxy service based on MQ transport.
- You cannot use an indirect call to a request-response MQ proxy service in the Oracle Service Bus test console.

33.9 Using the WebSphere JMS MQ Interface

The WebSphere MQ for Java APIs do not support remote XA transactions, though local transactions are supported. Consequently, Oracle Service Bus does not support remote XA transactions for the MQ transport. If XA is required, you can use the WebSphere MQ JMS interface.

The following sections outline how Oracle Service Bus connects to WebSphere MQ and presents an overview of some message types used in communication between WebSphere MQ and Oracle Service Bus.

- [Section 33.9.1, "Using the WebSphere MQ JMS Interface"](#)
- [Section 33.9.2, "Messaging Types"](#)
- [Section 33.9.3, "Tuning WebSphere MQ"](#)

33.9.1 Using the WebSphere MQ JMS Interface

Oracle Service Bus connects to WebSphere MQ via the WebSphere MQ JMS interface. That is, Oracle Service Bus is a WebSphere MQ JMS client.

The foreign JMS server in WLS specifies the initial context factory, connection factory, and queue to the WebSphere MQ server. For more information, see "Configuring Foreign Server Resources to Access Third-Party JMS Providers" in *Oracle Fusion Middleware Configuring and Managing JMS for Oracle WebLogic Server*.

WebSphere MQ JMS supports two transport types:

- BINDINGS
- CLIENT

If the WebSphere MQ JMS client is running on the same physical machine as the queue manager, it is possible to set the transport type to BINDINGS. Otherwise, you can use only the CLIENT type.

WebSphere MQ can interface with Oracle Service Bus in two ways:

- Oracle Service Bus acts as the front-end of WebSphere MQ to accept service requests from other applications and converts them to WebSphere MQ requests. See [Figure 33-1](#).
- WebSphere MQ sends messages to other applications through Oracle Service Bus. See [Figure 33-2](#).

Figure 33-1 Oracle Service Bus Front End



Figure 33-2 Messages Sent Through Oracle Service Bus



33.9.2 Messaging Types

Oracle Service Bus supports the following messaging types:

- [Section 33.9.2.1, "Non-Persistent Messaging"](#)
- [Section 33.9.2.2, "Non-XA Persistent Messaging"](#)
- [Section 33.9.2.3, "XA Messaging"](#)

33.9.2.1 Non-Persistent Messaging

If you decide to accept an unreliable delivery, such as some missing requests, you can use non-persistent messages where appropriate. WebSphere MQ logging and WebLogic JMS message persistence are only performed for persistent messages; therefore, the use of non-persistent messages eliminates any related I/O activity.

Note: Non-persistent message throughput is usually limited by the processor speed of the machine. However, in case of a shortage of physical memory, the server system may consume CPU cycles on a paging I/O.

33.9.2.2 Non-XA Persistent Messaging

WebSphere MQ persistent message throughput is usually limited by the queue manager and the I/O latency writing to the log.

33.9.2.3 XA Messaging

To enable support for transactional (XA) access to queues, use:

- `BINDINGS` to access the queue manager when Oracle Service Bus is co-located with IBM WebSphere MQ
- `CLIENT` when Oracle Service Bus and IBM WebSphere MQ are on different machines. However, with `CLIENT`, you need a special version of the IBM WebSphere MQ client that supports XA transactions, called the WebSphere MQ Extended Transaction Client.

Tip: For the deployment descriptors to be set appropriately for XA capable resources (JMS, TUXEDO, EJB), you must set the XA attribute on the referenced connection factory before creating a proxy service.

33.9.3 Tuning WebSphere MQ

The following guidelines help you tune WebSphere MQ when you are working with Oracle Service Bus. For detailed WebSphere MQ information, refer to relevant WebSphere MQ documentation.

- Use the `BINDINGS` transport type if Oracle Service Bus and the queue manager are deployed on the same machine.
- If you need XA for only a small section of application requests, create a separate connection object and disable XA.
- Distribute active logs across many volumes. If your system is required to handle high persistent message throughput, you must place the log files on a fast Direct Access Storage Device (DASD) with a minimum of contention from other data set usage. Ideally, you can allocate each of the active logs on separate, low-usage volumes.

- To reduce buffer overflow, tune the buffer pools and pagesets. Buffer overflow results in flushing of the hard disk.
- To avoid broken Oracle Service Bus JMS connections to MQ queues, increase the number of active channels to more than 100. By default, the number of active channels is 10.

Oracle BPEL Process Manager Transport

This document describes how to bring Oracle BPEL Process Manager into your service oriented architecture (SOA) environment using Oracle Service Bus. It contains the following sections:

- [Section 34.1, "Before You Begin"](#)
- [Section 34.2, "Overview"](#)
- [Section 34.3, "Simple Use Cases \(Synchronous\)"](#)
- [Section 34.4, "Advanced Use Cases \(Asynchronous\)"](#)
- [Section 34.5, "Transport Configuration Reference"](#)
- [Section 34.6, "Security"](#)
- [Section 34.7, "Error Handling"](#)
- [Section 34.8, "WS-Addressing Reference"](#)
- [Section 34.9, "XML Examples"](#)

34.1 Before You Begin

The BPEL Process Manager transport (bpel-10g in the user interface) is for messaging with only Oracle SOA Suite 10g Release 3. Oracle Service Bus provides the SOA-DIRECT transport for use with Oracle SOA Suite 11g and later. For more information, see [Chapter 24, "Oracle SOA Suite Transport \(SOA-DIRECT\)."](#)

34.2 Overview

Oracle Service Bus provides support for communication with Oracle BPEL Process Manager, letting you include BPEL processes in your service oriented architecture (SOA). With Oracle Service Bus's native transport for Oracle BPEL Process Manager (BPEL transport), you can expose BPEL processes as Web services in the service bus layer, letting other services invoke BPEL processes.

Likewise, Oracle BPEL Process Manager can call services in the service bus layer for use in its own processes.

While this guide focuses on built-in integration between Oracle Service Bus and Oracle BPEL Process Manager—integration enabled at the Oracle Service Bus API level, you are not limited to only the solutions provided in this guide. You may also use other standard communication protocols provided with Oracle Service Bus, such as HTTP, JMS, and File.

For detailed information on Oracle BPEL Process Manager, go to <http://www.oracle.com/technology/products/ias/bpel/index.html>.

34.2.1 SOAP Support

Communication between Oracle BPEL Process Manager and Oracle Service Bus is done over SOAP only.

Oracle Service Bus and Oracle BPEL Process Manager do not provide full support for SOAP RPC encoding. The BPEL transport accepts SOAP RPC encoding bindings, but some encoding mechanisms, such as multiRef, may cause run time failures.

The BPEL transport supports the following features:

- SOAP 1.1. SOAP 1.2 is supported only from Oracle Service Bus to Oracle BPEL Process Manager using synchronous communication.
- SOAP headers

The BPEL transport has the following restrictions:

- No attachments
- No WS-Security or WS-RM

34.2.2 Transaction Propagation

Oracle BPEL Process Manager supports transaction propagation through its API, and the BPEL transport is transactional to support transaction propagation when Oracle BPEL Process Manager is deployed on Oracle WebLogic Server. For example, if a process begins in a service outside of Oracle BPEL Process Manager, Oracle Service Bus can propagate the transaction to Oracle BPEL Process Manager through the BPEL transport to complete the transaction.

Transaction propagation is also supported from BPEL Process Manager to Oracle Service Bus through an SB transport-based proxy service.

Note: Transaction propagation is not yet supported for Oracle servers OC4J and Oracle AS and not yet certified on IBM WebSphere.

34.2.3 SSL Support

Because calls from Oracle Service Bus to Oracle BPEL Process Manager are made using RMI, the BPEL transport supports security at the call level through one-way SSL. For more information, see [Section 34.5.1, "Endpoint URI."](#)

34.2.4 Environment Variables

The BPEL transport declares the following environment values, which can be maintained when moving an Oracle Service Bus configuration among different deployment environments:

Table 34–1 BPEL transport environment variables

Environment Variable	Description
Dispatch Policy	The Oracle WebLogic Server Work Manager instance used for the service endpoint's dispatch policy.

Table 34–1 (Cont.) BPEL transport environment variables

Environment Variable	Description
Service Account	For JNDI context security; the service account used to access the Oracle BPEL Process Manager delivery service.
Endpoint URI	The URI of the service.

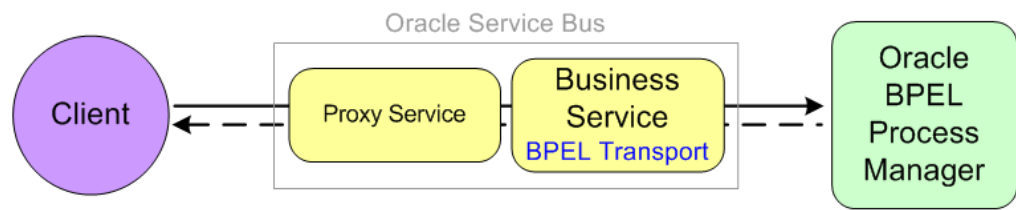
34.3 Simple Use Cases (Synchronous)

This section describes the simple, most common use cases for communicating to and from Oracle BPEL Process Manager through Oracle Service Bus: synchronous one-way or request/response communication.

34.3.1 Synchronous: Invoking Processes in Oracle BPEL Process Manager

Figure 34–1 illustrates a synchronous communication pattern between a client and Oracle BPEL Process Manager through Oracle Service Bus.

Figure 34–1 Invoking Oracle BPEL Processes Synchronously Through Oracle Service Bus



34.3.1.1 Creating and Configuring the Services

Use the following guidelines to invoke Oracle BPEL Process Manager processes from a client:

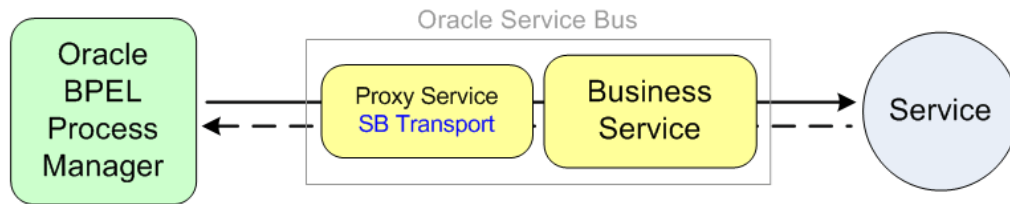
- Create a Business Service in Oracle Service Bus that represents the BPEL process you want to invoke.
 - Create a WSDL-based business service. Generate the WSDL from Oracle BPEL Process Manager.
 - Select the `bpel-10g` transport in the business service configuration.
 - Set the Endpoint URI as described in [Table 34–2](#).
 - Configure the remainder of the business service. See [Section 34.5, "Transport Configuration Reference."](#)
- Create a Proxy Service in Oracle Service Bus that invokes the business service. See [Section 2.3, "Working with Proxy Services."](#)

To ensure that messages are associated with the correct conversation, see [Section 34.3.3, "Associating Messages with the Correct Conversation."](#)

34.3.2 Synchronous: Calling External Services from Oracle BPEL Process Manager

Figure 34–2 illustrates a synchronous communication pattern between Oracle BPEL Process Manager and a service provider through Oracle Service Bus.

Figure 34–2 Oracle BPEL Processes Invoking a Service Synchronously Through Oracle Service Bus



34.3.2.1 Creating and Configuring the Services

Use the following guidelines to invoke an external service from Oracle BPEL Process Manager:

- Create a Business Service in Oracle Service Bus that represents the external service you want to invoke. See [Section 2.2, "Working with Business Services."](#)
- Create a Proxy Service in Oracle Service Bus that invokes the business service.
 - You must create the proxy with a SOAP WSDL to invoke the business service. When defining your proxy service, for the Service Type select WSDL Web Service, and select the desired port or binding.
 - Select the sb transport in the proxy service configuration.
 - To invoke the proxy service from Oracle BPEL Process Manager, export the proxy service's effective WSDL and import it into your Oracle BPEL Process Manager development environment. Invoke the proxy service from Oracle BPEL Process Manager as you normally would.

For configuration details, see [Section 2.3, "Working with Proxy Services"](#) and [Chapter 27, "SB Transport."](#)

To ensure that messages are associated with the correct conversation, see [Section 34.3.3, "Associating Messages with the Correct Conversation."](#)

34.3.3 Associating Messages with the Correct Conversation

When using stateful services, the messages sent synchronously between Oracle Service Bus and Oracle BPEL Process Manager are known as a conversation. Oracle BPEL Process Manager supports the following mechanisms for ensuring that messages are correctly associated with each other as part of a conversation. These mechanisms are independent of each other, and you may choose to use both to ensure correct association.

- **BPEL Correlation** – BPEL correlation is part of the BPEL specification. When a WSDL-based business service in Oracle Service Bus sends a message to a BPEL process, the BPEL engine examines the message to find the target BPEL process instance.
- **Opaque Correlation using WS-Addressing** – When a conversation is initiated by a client through Oracle Service Bus to a BPEL process, the BPEL engine looks in the WS-Addressing SOAP header for the "messageID" value to use as the ID for the new conversation. The conversation ID is carried through the conversation as the "RelatesTo" value.

For more information on WS-Addressing, see [Section 34.8.2, "MessageID / RelatesTo"](#) in the WS-Addressing reference. For an example of conversation ID setting, see [Section 34.9.1, "Conversation ID Examples."](#)

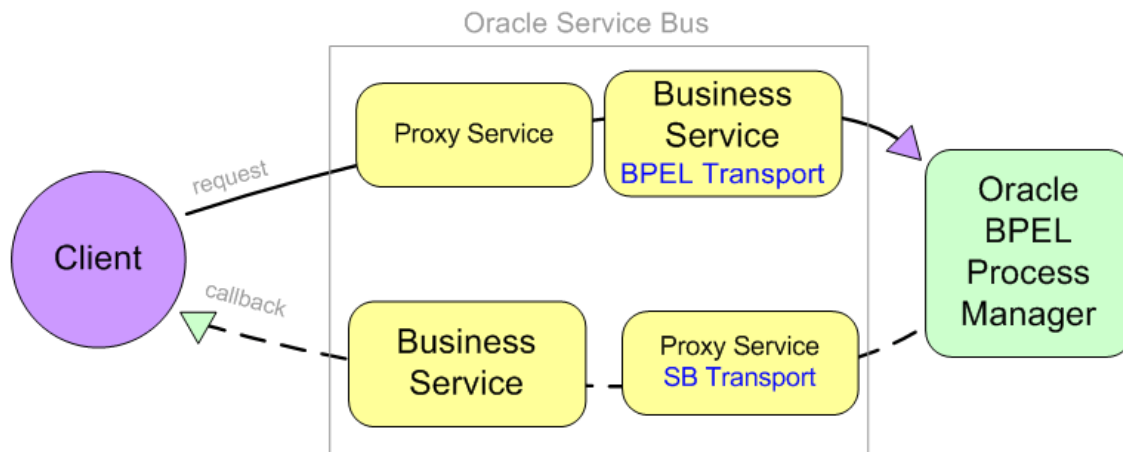
34.4 Advanced Use Cases (Asynchronous)

This section describes more advanced use cases for communicating to and from Oracle BPEL Process Manager through Oracle Service Bus using asynchronous communication.

34.4.1 Asynchronous: Invoking Processes in Oracle BPEL Process Manager

Figure 34–3 illustrates an asynchronous communication pattern between Oracle BPEL Process Manager and a service provider through Oracle Service Bus.

Figure 34–3 Invoking Oracle BPEL Processes Asynchronously Through Oracle Service Bus



In an asynchronous message exchange, a callback is sent on a different connection than the request.

34.4.1.1 Creating and Configuring the Services

Use the following guidelines to invoke Oracle BPEL Process Manager processes asynchronously from a client through Oracle Service Bus:

- Create two proxy services in Oracle Service Bus: one that invokes the business service and another that handles the callback.

Request Proxy Service

- Since the callback will be sent on a different connection in asynchronous communication, you must establish the callback address in the request proxy. This callback address will be passed to the callback proxy and callback business services so that the message is sent back to the correct client.

As part of the business service configuration, you select a Callback Proxy. At run time, the BPEL transport uses this proxy as the callback proxy.

For approaches to setting a callback address if you do not select a callback proxy in the business service, see [Section 34.8, "WS-Addressing Reference"](#) and [Section 34.9.2, "Asynchronous BPEL to BPEL Through Oracle Service Bus Example."](#)

Callback Proxy Service

- Configure the proxy to use a Service Type of WSDL SOAP or Any SOAP Service and the SB or HTTP transport. Use the SB transport if you want

transaction propagation from Oracle BPEL Process Manager to Oracle Service Bus.

If you select this proxy service as the business service's callback proxy, the BPEL transport provides the correct callback URI at run time.

For proxy configuration details, see [Section 2.3, "Working with Proxy Services."](#)

- Create two business services in Oracle Service Bus: one that makes the request to the Oracle BPEL Process Manager process you want to interact with and another that handles the callback.

Request Business Service

- Create a WSDL-based business service. Generate the WSDL from Oracle BPEL Process Manager. Select a Service Type of WSDL Web Service, and select the appropriate binding or port in the WSDL.
- Select the bpel-10g transport in the business service configuration.
- Set the role to Asynchronous Client.
- Set the Endpoint URI, described in [Table 34-2](#).
- Use the Callback Proxy field on the bpel-10g transport configuration page to select the callback proxy you created.

Callback Business Service

Configure the business process you need to handle the callback.

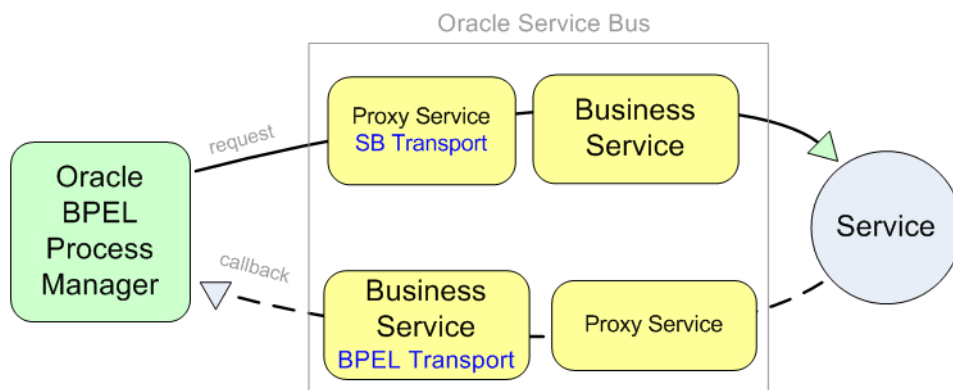
See [Section 34.5, "Transport Configuration Reference."](#) See also [Section 4.2.4, "Business Service Transport Configuration Page"](#) for configuration information not covered in this document.

34.4.2 Asynchronous: Calling Service Providers from Oracle BPEL Process Manager

This section describes the steps and configuration needed for Oracle BPEL Process Manager to make service calls through Oracle Service Bus.

[Figure 34-4](#) illustrates an asynchronous communication pattern between Oracle BPEL Process Manager and a service provider through Oracle Service Bus.

Figure 34-4 Oracle BPEL Processes Invoking a Service Asynchronously Through Oracle Service Bus



In an asynchronous message exchange, a callback is sent on a different connection than the request.

34.4.2.1 Creating and Configuring the Services

Use the following guidelines to invoke an external service asynchronously from Oracle BPEL Process Manager through Oracle Service Bus.

- Create two proxy services in Oracle Service Bus: one for the request that invokes the business service and another that handles the callback.

Request Proxy Service

- Configure the proxy service to use the sb transport.
- Since the callback will be sent on a different connection in asynchronous communication, you must establish a callback address so that the message is sent back to the correct client.

For information on setting a callback address, see [Section 34.8.1, "ReplyTo"](#) of the WS-Addressing reference and [Section 34.9.2, "Asynchronous BPEL to BPEL Through Oracle Service Bus Example."](#)

Callback Proxy Service

- Configure the proxy service to pass the callback address to the business service. The callback URI is provided in the request. Use URI rewriting to extract the callback URI and forward it to the business service.

For proxy configuration details, see [Section 2.3, "Working with Proxy Services"](#) and [Chapter 27, "SB Transport."](#)

- Create two business services in Oracle Service Bus: a request business service that invokes the external service and a callback business service.

Request Business Service

- Configure the business service to invoke the external service.

Callback Business Service

- The callback business service receives the callback address from the callback proxy. The URI rewriting performed by the callback proxy service determines which BPEL process to send the response to.

Create a WSDL-based business service. Generate the WSDL from Oracle BPEL Process Manager. Select a Service Type of **WSDL Web Service**, and select the appropriate binding or port in the WSDL

- Select the bpel-10g transport in the business service configuration.
- Set the Endpoint URI to `bpel://callback`, as described in [Table 34–2](#). The callback URI is provided by the callback proxy service.

Note: If the callback address is always known, for example when the client and BPEL service are linked together because of a trading partner agreement, you can provide the exact callback address in the callback business service instead of using `bpel://callback`.

- Set the role to Service Callback on the bpel-10g transport configuration page, as described in [Table 34–2](#).

Configure the remainder of the business service. See [Section 34.5, "Transport Configuration Reference."](#) See also [Section 4.2.4, "Business Service Transport Configuration Page"](#) for configuration information not covered in this document.

34.5 Transport Configuration Reference

This section provides descriptions for BPEL transport-specific configuration options. For descriptions of other business service configuration options, see [Section 2.2, "Working with Business Services."](#)

34.5.1 Endpoint URI

[Table 34–2](#) describes the URI format for the BPEL transport, which you configure on the main Transport page of the business service in either Eclipse or in the Oracle Service Bus console.

Table 34–2 Specifying an Endpoint URI

Transport used as	Endpoint URI format
Synchronous Client or Asynchronous Client role	<p>For the following endpoint URI format, optional elements are shown in square brackets. Descriptions follow.</p> <p>protocol://host[:port][[/protocol-path]/domain/process[/version[/partnerlink/role]]</p> <p>protocol</p> <p>Use one of the following RMI / JNDI protocols:</p> <ul style="list-style-type: none"> ▪ iiop / iiops – For generic, server-agnostic use. ▪ t3 / t3s – For use with Oracle WebLogic Server. ▪ http / https – For tunneling and use with Oracle WebLogic Server. ▪ ormi / ormis – For stand-alone deployment on OC4J (Oracle Container for JEE). <p>Following are descriptions of the other endpoint URI elements:</p> <ul style="list-style-type: none"> ▪ port – Optional, for the ormi and opmn protocols only, if the server is configured to use the default port. ▪ protocol-path – For use only with the opmn / opmns protocol. The protocol-path is the server instance in a cluster. ▪ version – Optional. The version of the process to invoke. ▪ partnerlink/role – Optional. Include for a full path description when you specify version. <p>Cluster example</p> <p>You can also use the following URI format for targeting specific nodes in a cluster:</p> <p>protocol://host1:port1,host2:port2/<remainder_of_URI></p>
Service Callback	<p>bpel://callback</p> <p>If the callback address is always known, for example when the client and BPEL service are linked together because of a trading partner agreement, you can provide the exact callback address for the Endpoint URI instead of using bpel://callback.</p>

34.5.2 bpel-10g Transport Configuration

[Table 34–2](#) describes the options available on the BPEL transport configuration page in either Eclipse or in the Oracle Service Bus console.

Table 34–3 BPEL transport configuration

Property	Description
Role	<p>The BPEL transport is used to send request messages from Oracle Service Bus to Oracle BPEL Process Manager. The business service can serve one of the following roles:</p> <ul style="list-style-type: none"> <li data-bbox="683 365 1442 512"> <p>■ Synchronous Client</p> <p>For synchronous communication with an Oracle Service Bus client, the only location information that is required is the BPEL address. This address is captured statically as the Endpoint URI (described in Table 34–2) and/or dynamically through URI rewriting.</p> <li data-bbox="683 527 1442 722"> <p>■ Asynchronous Client</p> <p>For asynchronous communication with an Oracle Service Bus client, a callback from Oracle BPEL Process Manager to Oracle Service Bus is sent on a different connection than the request, and you must configure Oracle Service Bus to provide the correct callback address. For more information, see the guidelines in Section 34.4, "Advanced Use Cases (Asynchronous)."</p> <li data-bbox="683 737 1442 1041"> <p>■ Service Callback</p> <p>Use this role if the business service is designed to be a service callback to Oracle BPEL Process Manager (where Oracle BPEL Process Manager is calling an service provider asynchronously through Oracle Service Bus).</p> <p>A Service Callback business service does not support load balancing or failover.</p> <p>For instructions on using Service Callback, see "Service Callback" in Table 34–2 and Section 34.4.2, "Asynchronous: Calling Service Providers from Oracle BPEL Process Manager."</p>
Callback Proxy	<p>This optional field is available only for the Asynchronous Client role. This field lets you select the proxy service (must be either an SB or HTTP proxy of type WSDL SOAP or Any SOAP) that will be used to route callbacks to the Oracle Service Bus client that made the request. For more information, see the guidelines in Section 34.4, "Advanced Use Cases (Asynchronous)."</p>
Service Account	<p>For JNDI context security, used to access the Oracle BPEL Process Manager delivery service. Click Browse and select a service account. If no service account is specified, an anonymous subject is used.</p> <p>There is no restriction on the type of service account that can be configured, such as static or pass-through, but the run time must be able to access a user name and password.</p>

Table 34–3 (Cont.) BPEL transport configuration

Property	Description
Suspend Transaction	<p>Selecting Suspend Transaction makes the business service non-transactional even if the business service is invoked by a transaction.</p> <p>If you do not select Suspend Transaction:</p> <ul style="list-style-type: none"> ▪ If the protocol indicates an Oracle WebLogic Server-supported protocol (t3, iiop, http), the transaction is propagated. ▪ If the protocol indicates an OC4J server (ormi, opmn), the BPEL transport throws an exception, since OC4J does not support transaction propagation. <p>The BPEL transport ignores the Suspend Transaction option in the following situations:</p> <ul style="list-style-type: none"> ▪ The business service is called with quality of service (QoS) BEST_EFFORT. The BPEL transport automatically suspends any transaction that does not support QoS. ▪ The business service is called with QoS set to EXACTLY_ONCE and there is no transaction. <p>For a description of transaction propagation, see Section 34.2.2, "Transaction Propagation."</p>
Dispatch Policy	<p>Select the instance of Oracle WebLogic Server Work Manager that you want to use for the dispatch policy for this endpoint. The default Work Manager is used if no other Work Manager exists.</p> <p>For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in <i>Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server</i>.</p>

34.6 Security

Security in Oracle BPEL Process Manager is handled at the EJB level.

The BPEL transport supports a JNDI "Service Account" option used for the creation of the JNDI context and the EJB lookup, as described in [Table 34–2](#). The BPEL transport gets the user name and password. If the service account is not configured, an anonymous subject is assumed.

The BPEL transport also supports security at the call level by letting you indicate one-way SSL in the protocol portion of the URI from Oracle Service Bus to Oracle BPEL Process Manager. For more information, see [Section 34.5.1, "Endpoint URI."](#)

34.6.1 Using SSL from Oracle Service Bus to Oracle Servers

To use SSL from Oracle Service Bus to OC4J and Oracle AS servers (using the ormis and opmns protocols), you must configure SSL on your Oracle Service Bus server by adding the following properties to your domain's setDomainEnv script:

```
-Djavax.net.ssl.trustStorePassword=passphrase
```

```
-Djavax.net.ssl.trustStore=<file path to a keystore of trust certificate>
```

34.7 Error Handling

The BPEL transport handles Oracle BPEL Process Manager exceptions in different ways.

34.7.1 Application Errors

If a BPEL process replies with a fault, the BPEL transport intercepts the fault message at the API and translates it into a SOAP fault.

The transport's automated application error-handling functionality lets you decide whether or not to automatically retry application errors—in this case BPEL faults—when they occur. For example, you may determine that application errors will always fail until the problem is fixed. Use the **Retry Application Errors** option on the transport configuration page to turn application retries on and off.

34.7.2 Connection Errors

Oracle BPEL Process Manager can throw the following non-fault exceptions, which the BPEL transport automatically categorizes as `TransportException` connection errors:

- `NamingException`
- `RemoteException`

The transport's automated connection error-handling functionality lets you determine if and how often connection errors are retried. Use the **Retry Count** and the **Retry Iteration Interval** options on the transport configuration page to control the number of retries and the number of seconds to wait between retries on connection errors. Setting `Retry Iteration Interval` to zero (0) means connection errors are not retried.

34.7.3 Other Errors

Other non-application and non-connection exceptions are re-thrown as a generic `TransportException` error.

34.8 WS-Addressing Reference

This section describes specific WS-Addressing properties that the BPEL transport uses to communicate with Oracle BPEL Process Manager. This section also describes ways to provide callback addresses in asynchronous communication, as described in the [Section 34.4, "Advanced Use Cases \(Asynchronous\)"](#) guidelines.

See [Section 34.9, "XML Examples"](#) for WS-Addressing examples.

34.8.1 ReplyTo

In asynchronous communication, a service callback is sent on a different connection than the request. As a service developer, you must supply the correct callback address in an asynchronous exchange so that the callback is sent to the correct client.

34.8.1.1 Calling a BPEL Process Asynchronously Through Oracle Service Bus

The BPEL transport provides a built-in mechanism for providing the correct callback address. When you create a BPEL business service in Oracle Service Bus, you can select a `Callback Proxy` to handle the callback, and the BPEL transport automatically sets the correct callback address. You do not need to manually use `"ReplyTo."`

For more information, see [Section 34.4.1, "Asynchronous: Invoking Processes in Oracle BPEL Process Manager."](#)

34.8.1.2 BPEL Processes Calling External Services Through Oracle Service Bus

When calling an external service from Oracle BPEL Process Manager through Oracle Service Bus, you must manually set a callback address. To do this using WS-Addressing, in the request proxy service set the callback address as the "ReplyTo" value. The BPEL transport in the business service uses that URI as the callback address.

For more information, see [Section 34.4.2, "Asynchronous: Calling Service Providers from Oracle BPEL Process Manager."](#)

34.8.2 MessageID / RelatesTo

"MessageID" and "RelatesTo" are used to store the conversation ID in conversations between Oracle Service Bus and Oracle BPEL Process Manager, making sure related messages remain in the same conversation.

The BPEL transport does not let you specify whether a given operation is a start or continue operation. Instead, the BPEL transport looks for the "MessageID" and "RelatesTo" properties and sets them accordingly.

The following describes how the BPEL transport uses "MessageID" and "RelatesTo" in synchronous and asynchronous conversations:

- **Synchronous conversation:** In the initial request, the "MessageID" determines the conversation ID. In the remaining communication, the BPEL transport provides the conversation ID as the RelatesTo value.

If there is no value assigned to "MessageID" or "RelatesTo," the transport assumes either no conversation is occurring or that Oracle BPEL Process Manager is handling the correlation.

- **Asynchronous callbacks** - In the initial request, the "MessageID" determines the conversation ID. In the remaining communication, the BPEL transport provides the conversation ID as the "RelatesTo" value in the callback.

If there is no value assigned to "MessageID" or "RelatesTo," the transport assumes either no conversation is occurring or that Oracle BPEL Process Manager is handling the correlation.

For more implementation on establishing a conversation ID to make sure messages participate in the correct conversation, see [Section 34.3.3, "Associating Messages with the Correct Conversation"](#) and the [Section 34.9.1, "Conversation ID Examples."](#)

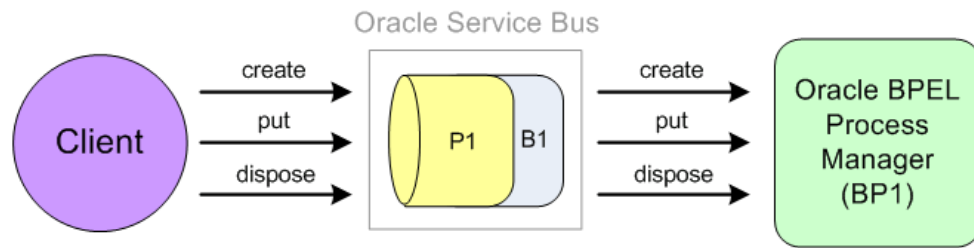
34.9 XML Examples

Following are examples of XML messaging issues between Oracle Service Bus and Oracle BPEL Process Manager.

34.9.1 Conversation ID Examples

This section provides different examples of establishing a conversation ID among messages in a conversation between Oracle Service Bus and Oracle BPEL Process Manager.

In [Figure 34-5](#), a client is synchronously invoking a process in Oracle BPEL Process Manager. The business service (B1) uses the BPEL transport to invoke a process. The proxy service (P1) handles any necessary conversation ID mapping.

Figure 34–5 Operations in a Synchronous Exchange Through Oracle Service Bus

34.9.1.1 Port and Message Definitions

The examples in this section use the following port and message definitions defined in the WSDL.

```

<wsdl:types>
  <xsd:schema
    targetNamespace="http://www.sample.org/spec/samples/types"
    elementFormDefault="qualified">
    <xsd:complexType name="ValueHolder">
      <xsd:all>
        <xsd:any minOccurs="1"/>
      </xsd:all>
    </xsd:complexType>
  </xsd:schema>
</wsdl:types>
<message name="create"/>
<message name="putRequest">
  <part name="key" type="xsd:string"/>
  <part name="value" type="types:ValueHolder"/>
</message>
<message name="putResponse">
  <part name="value" type="types:ValueHolder"/>
</message>
...
<message name="dispose"/>
<portType name="ServiceMap">
  <operation name="create">
    <input message="tns:create"/>
  </operation>
  <operation name="put">
    <input message="tns:putRequest"/>
    <output message="tns:putResponse"/>
  </operation>
  ...
  <operation name="dispose">
    <input message="tns:dispose"/>
  </operation>
</portType>
  
```

34.9.1.2 WS-Addressing that Sets the Conversation ID

This example shows how WS-Addressing is used to set the conversation ID among messages in a conversation.

Figure 34–5 shows communication pattern.

Create Operation

```
<soap:Envelope>
```

```

    <soap:Header>
      <wsa03:MessageID>uuid:123456789</wsa03:MessageID>
    </soap:Header>
    <soap:Body>
      <create/>
    </soap:Body>
  </soap:Envelope>

```

Put Operation

```

<soap:Envelope>
  <soap:Header>
    <wsa03:MessageID>uuid:111111111</wsa03:MessageID>
    <wsa03:RelatesTo>uuid:123456789</wsa03:RelatesTo>
  </soap:Header>
  <soap:Body>
    <put>
      <key>key</key>
      <value>
        <PO/>
      </value>
    </put>
  </soap:Body>
</soap:Envelope>
<soap:Envelope>
  <soap:Body>
    <putResponse>
      <value/>
    </putResponse>
  </soap:Body>
</soap:Envelope>

```

The <put> operation also has a MessageID, but it is ignored because the RelatesTo has a value that provides the conversation ID.

34.9.1.3 Message Payload Data that Sets the Conversation ID

This example shows how message payload data can be used to set the conversation ID among messages in a conversation.

In these examples, the proxy service maps the ID to the MessageID / RelatesTo SOAP headers.

[Figure 34–5](#) shows communication pattern.

Create Operation

Client to proxy service

```

<soap:Envelope>
  <soap:Body>
    <create/>
  </soap:Body>
</soap:Envelope>
<soap:Envelope>
  <soap:Body>
    <createResponse>
      <mapID>uuid:123456789</mapID>
    </createResponse>
  </soap:Body>
</soap:Envelope>

```

Proxy service to BPEL process (via a business service)

```

<soap:Envelope>
  <soap:Header>
    <wsa03:MessageID>uuid:123456789</wsa03:MessageID>
  </soap:Header>
  <soap:Body>
    <create/>
  </soap:Body>
</soap:Envelope>

```

Not shown: The ID was generated in the request of the proxy service pipeline and inserted as a <wsa03:MessageID> before invoking the process. On the process side, the Create operation is one-way, so a SOAP response must be created before replying to the client. The response sends back the ID that was generated by the proxy service.

Put Operation

Client to proxy service

```

<soap:Envelope>
  <soap:Body>
    <put>
      <mapID>uuid:123456789</mapID>
      <key>key</key>
      <value>
        <PO/>
      </value>
    </put>
  </soap:Body>
</soap:Envelope>
<soap:Envelope>
  <soap:Body>
    <putResponse>
      <value/>
    </putResponse>
  </soap:Body>
</soap:Envelope>

```

Proxy service to BPEL process (via a business service)

```

<soap:Envelope>
  <soap:Header>
    <wsa03:RelatesTo>uuid:123456789</wsa03:RelatesTo>
  </soap:Header>
  <soap:Body>
    <put>
      <key>key</key>
      <value>
        <PO/>
      </value>
    </put>
  </soap:Body>
</soap:Envelope>
<soap:Envelope>
  <soap:Body>
    <putResponse>
      <value/>
    </putResponse>
  </soap:Body>
</soap:Envelope>

```

Dispose Operation

Client to proxy service

```

<soap:Envelope>
  <soap:Body>
    <dispose>
      <mapID>uuid:123456789</mapID>
    </dispose>
  </soap:Body>
</soap:Envelope>

```

Proxy service to BPEL process (via a business service)

```

<soap:Envelope>
  <soap:Header>
    <wsa03:RelatesTo>uuid:123456789</wsa03:RelatesTo>
  </soap:Header>
  <soap:Body>
    <dispose/>
  </soap:Body>
</soap:Envelope>

```

34.9.1.4 Transformation Examples

In these examples, the client uses a more recent version of the WS-Addressing spec (wsa04 prefix). The proxy service is responsible for transforming the SOAP headers to use the wsa03 prefix. The proxy service developer configures the transformation.

[Figure 34–5](#) shows communication pattern.

Create Operation

Client to proxy service

```

<soap:Envelope>
  <soap:Header>
    <wsa04:MessageID>uuid:123456789</wsa04:MessageID>
  </soap:Header>
  <soap:Body>
    <create/>
  </soap:Body>
</soap:Envelope>

```

Proxy service to BPEL process (via a business service)

```

<soap:Envelope>
  <soap:Header>
    <wsa03:MessageID>uuid:123456789</wsa03:MessageID>
  </soap:Header>
  <soap:Body>
    <create/>
  </soap:Body>
</soap:Envelope>

```

Put Operation

Client to proxy service

```

<soap:Envelope>
  <soap:Header>
    <wsa04:MessageID>uuid:11111111</wsa04:MessageID>
    <wsa04:RelatesTo>uuid:123456789</wsa04:RelatesTo>
  </soap:Header>

```

```

    <soap:Body>
      <put>
        <key>key</key>
        <value>
          <PO/>
        </value>
      </put>
    </soap:Body>
  </soap:Envelope>
</soap:Envelope>
<soap:Body>
  <putResponse>
    <value/>
  </putResponse>
</soap:Body>
</soap:Envelope>

```

Proxy service to BPEL process (via a business service)

```

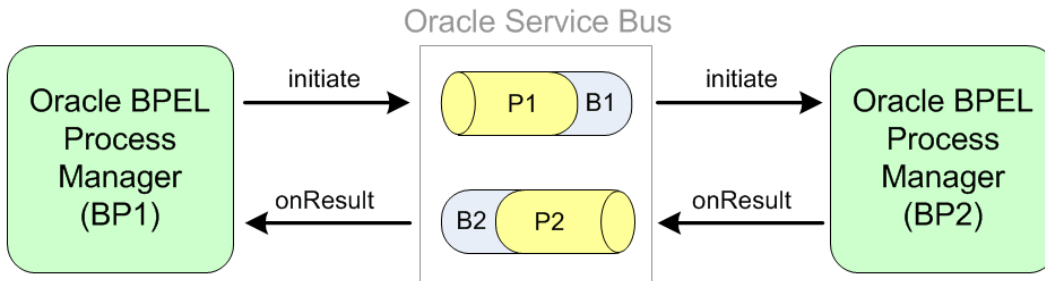
<soap:Envelope>
  <soap:Header>
    <wsa03:MessageID>uuid:111111111</wsa03:MessageID>
    <wsa03:RelatesTo>uuid:123456789</wsa03:RelatesTo>
  </soap:Header>
  <soap:Body>
    <put>
      <key>key</key>
      <value>
        <PO/>
      </value>
    </put>
  </soap:Body>
</soap:Envelope>
<soap:Envelope>
  <soap:Body>
    <putResponse>
      <value/>
    </putResponse>
  </soap:Body>
</soap:Envelope>

```

34.9.2 Asynchronous BPEL to BPEL Through Oracle Service Bus Example

The following example shows the SOAP headers involved in one BPEL process invoking another BPEL process asynchronously through Oracle Service Bus.

In [Figure 34–6](#), P1 and P2 are proxy services that pass messages (and perform transformations) to B1 and B2 business services, which are required to make calls to BPEL processes using the Oracle Service Bus BPEL transport.

Figure 34–6 One BPEL Process Invoking Another BPEL Process Through Oracle Service Bus

Refer to [Figure 34–6](#) for the following SOAP header examples.

34.9.2.1 Port and Message Definitions

```
<message name="LoanServiceRequestMessage">
  <part name="payload" element="types:loanApplication"/>
</message>
<message name="LoanServiceResultMessage">
  <part name="payload" element="types:loanOffer"/>
</message>
<portType name="LoanService">
  <operation name="initiate">
    <input message="tns:LoanServiceRequestMessage"/>
  </operation>
</portType>
<portType name="LoanServiceCallback">
  <operation name="onResult">
    <input message="tns:LoanServiceResultMessage"/>
  </operation>
</portType>
```

34.9.2.2 BP1 to P1 – Initiate operation

```
<soap:Envelope>
  <soap:Header>
    <wsa3:ReplyTo>
      <wsa3:Address>
ormi://serverB:7001/default/AmericanLoanClient/1.0/service/LoanServiceRequester
      </wsa3:Address>
    </wsa3:ReplyTo>
    <MessageID>AmericanLoanClient-1.0/60007</MessageID>
  </soap:Header>
  <soap:Body >
    <loanApplication>
      ...
    </loanApplication>
  </soap:Body>
</soap:Envelope>
```

34.9.2.3 P1/B1 to BP2

```
<soap:Envelope>
  <soap:Header>
    <wsa3:ReplyTo>
      <wsa3:Address>http://serverB:7001/P2</wsa3:Address>
      <wsa3:ReferenceProperties>
        <osb:Callback>
          <osb:Address>
```



```

ormi://localhost/default/AmericanLoanClient/1.0/service/LoanServiceRequester
  </osb:Address>
  </osb:Callback>
  </wsa03:ReferenceProperties>
  </wsa03:ReplyTo>
  <MessageID>AmericanLoanClient-1.0/60007</MessageID>
</soap:Header>
<soap:Body >
  <loanApplication>
    ...
  </loanApplication>
</soap:Body>
</soap:Envelope>

```

The ReplyTo callback address is set by B1, which gets the value from the Callback Proxy field in the BPEL transport configuration, as described in [Table 34–2](#). B1's callback proxy is P2.

You must wrap the original replyTo information and send it as reference properties so that it is echoed back in the onResult callback message (to follow).

Note: This sample uses osb:Callback and osb:Address for illustration purpose only. There is no standard or Oracle Service Bus standard elements defined for WS-Addressing support.

34.9.2.4 BP2 to P2 – onResult operation

```

<soap:Envelope>
  <soap:Header>
    <wsa03:RelatesTo>AmericanLoanClient~1.0/60007</wsa03:RelatesTo>
    <osb:Callback>
      <osb:Address>
ormi://localhost/default/AmericanLoanClient/1.0/service/LoanServiceRequester
      </osb:Address>
    </osb:Callback>
  </soap:Header>
  <soap:Body >
    <loanOffer>
      ...
    </loanOffer>
  </soap:Body>
</soap:Envelope>

```

The reference property osb:Callback is sent back as a SOAP header by the Oracle BPEL Process Manager engine.

34.9.2.5 P2/B2 to BP1 – onResult operation

```

<soap:Envelope>
  <soap:Header>
    <wsa03:RelatesTo>AmericanLoanClient~1.0/60007</wsa03:RelatesTo>
  </soap:Header>
  <soap:Body >
    <loanOffer>
      ...
    </loanOffer>
  </soap:Body>
</soap:Envelope>

```

Proxy service P2 removes the temporary `osb:Callback` header; but prior to deleting this header, the `replyTo` address value is copied to the `$outbound` variable so that the BPEL transport in business service B2 can send the callback message to the correct BPEL process.

Tuxedo Transport

The Oracle Service Bus Tuxedo transport lets you bring Tuxedo services into the service bus environment.

This chapter includes the following topics:

- [Section 35.1, "Overview"](#)
- [Section 35.2, "Configuring the Oracle Tuxedo Connector"](#)
- [Section 35.3, "Using Tuxedo Services from Oracle Service Bus"](#)
- [Section 35.4, "Using Oracle Service Bus from Tuxedo"](#)
- [Section 35.5, "Tuxedo Transport Buffer Transformation"](#)
- [Section 35.6, "Tuxedo Transport Transaction Processing"](#)

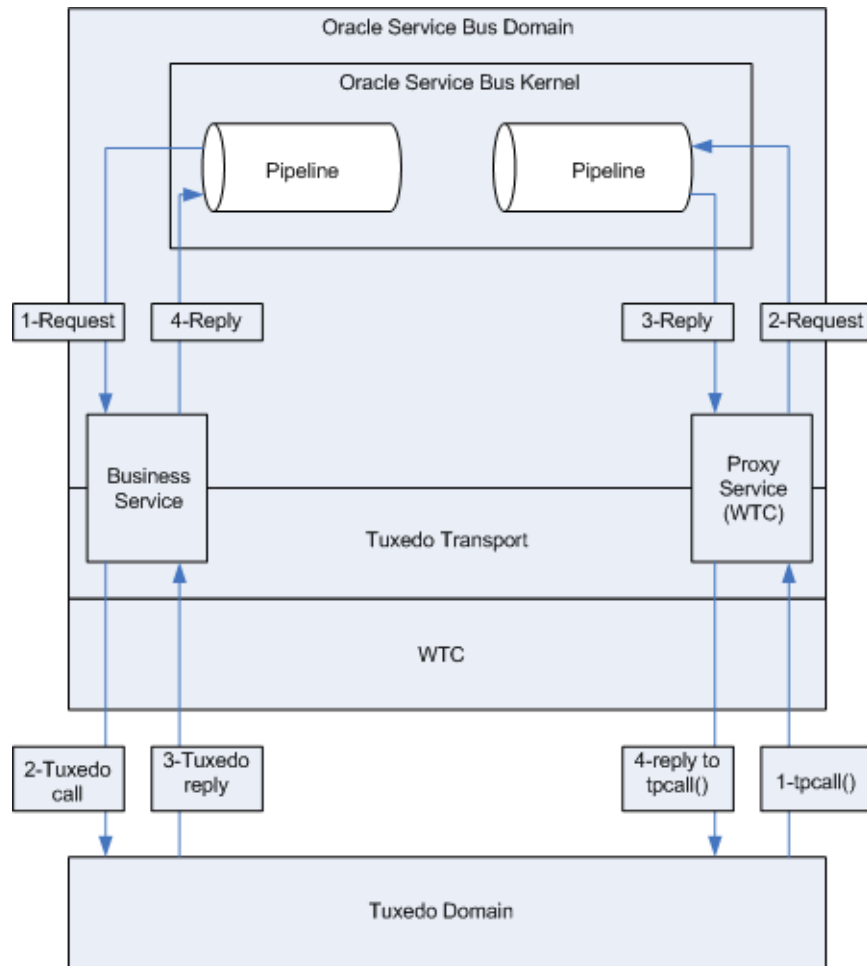
35.1 Overview

Oracle Service Bus and Oracle Tuxedo can interoperate to use the services that each product offers. The Tuxedo transport allows secure, guaranteed, high performance, bi-directional access to a Tuxedo domain from Oracle Service Bus. The Tuxedo transport allows Tuxedo domains to call services as well as have services called in a Tuxedo domain.

- When Oracle Service Bus uses services offered by Tuxedo, the Tuxedo transport facilitates access to those Tuxedo services. The term *outbound* refers to this business service scenario.
- When Tuxedo uses services offered by Oracle Service Bus, Tuxedo services can call Oracle Service Bus services as though they were another Tuxedo application. The term *inbound* refers to this proxy service scenario.

You configure the Tuxedo transport using the Oracle Service Bus Console. Specific parameters provide definitions for both proxy and business services. A basic Oracle Tuxedo Connector (WTC) configuration with one local access point and one remote access point is required to enable configuration of the Tuxedo transport. Support for transactional and security contexts are available as well.

The following diagram summarizes this message handling processes.

Figure 35-1 WTC Message Handling

35.1.1 Capabilities of the Tuxedo transport

The following capabilities are available in the native Tuxedo transport in Oracle Service Bus.

- First-class tier transport

The native Tuxedo transport is fully integrated into the Oracle Service Bus configuration, management, and monitoring console. You can configure, manage, and monitor both Tuxedo proxy services and Tuxedo businesses services.

- Bi-directional access

Oracle Service Bus is an intermediary between SOAP, JMS, or other services and Tuxedo. The Tuxedo transport enables access to Tuxedo ATMI services as business services in Oracle Service Bus and allows Oracle Service Bus proxy services to be seen by Tuxedo as another ATMI service.

- Buffer transformation

You can transform XML messages to Tuxedo buffer types and Tuxedo buffer types to XML. All standard Tuxedo buffer types are supported; transformation is automatic and transparent. For more information, see [Section 35.5, "Tuxedo Transport Buffer Transformation."](#)

- Transactional integrity

Tuxedo transport provides transactional integrity for inbound and outbound messages. You can call Tuxedo services in the context of a global transaction allowing Exactly Once quality of service (QoS). A Tuxedo application can start a transaction, call an Oracle Service Bus-based service and the XA transaction context is carried through to Oracle Service Bus, through the pipeline, and finally to the destination transport. For more information, see [Section 35.6, "Tuxedo Transport Transaction Processing."](#)

- Security propagation

The security context established at the beginning of the message flow, from either a Tuxedo client or an Oracle Service Bus client, is propagated to the other system. This means that an incoming SOAP over HTTP request to Oracle Service Bus that requires authentication is authenticated by Oracle Service Bus. As with transactions, this support is fully bi-directional, so a client authenticated to Tuxedo can make requests to Oracle Service Bus services without requiring authentication a second time.

- Encrypted network links

You can encrypt the connections between Oracle Service Bus and Tuxedo through WTC configuration to ensure the security and privacy of communications between the two systems.

- Load balancing

A single network connection is the only requirement to connect Oracle Service Bus to a Tuxedo domain. However, it might be necessary to support multiple connections in case of a machine or network failure. You can make multiple connections to a single domain or multiple domains for purposes of load balancing.

See the following sections for detailed information on configurations for the following scenarios:

- [Section 35.2, "Configuring the Oracle Tuxedo Connector"](#)
- [Section 35.3, "Using Tuxedo Services from Oracle Service Bus"](#)
- [Section 35.4, "Using Oracle Service Bus from Tuxedo"](#)
- "How to Configure Oracle WebLogic Tuxedo Connector to Provide Security between Oracle Tuxedo and Oracle WebLogic Server" in the *Oracle Fusion Middleware Tuxedo Connector Administration Guide for Oracle WebLogic Server*.

35.2 Configuring the Oracle Tuxedo Connector

Oracle Service Bus Tuxedo transport enables access to Tuxedo services using Oracle Tuxedo Connector (WTC). To use the Tuxedo transport, you must configure a basic WTC server including one local access point and one remote access point.

The following sections describe how to configure WTC:

- [Section 35.2.1, "Before You Begin"](#)
- [Section 35.2.2, "Configuring Oracle Tuxedo Connector"](#)

35.2.1 Before You Begin

Gather the following information about the Tuxedo application that Oracle Service Bus will use:

- ID of the Tuxedo local access point.
- Network address of the Tuxedo local access point.
- Name of the exported Tuxedo service.
- Whether the service needs XML-to-FML and FML-to-XML conversion or VIEW-to-XML or XML-to-VIEW conversion.

The example described in the following sections assumes the use of FML/FML32 buffer types.

- ID of the access point that the Tuxedo domain gateway will use to refer to this Oracle Tuxedo Connector instance. **Access Point ID**; it is referred to as the remote access point ID.)
- Network address that the Tuxedo domain gateway has defined for this Oracle Tuxedo Connector local access point. **Network Address**; it is referred to as the remote network address.)

35.2.2 Configuring Oracle Tuxedo Connector

When you create or import Tuxedo business and proxy services in Oracle Service Bus, the service configuration includes WebLogic Tuxedo Connector configurations that appear as WebLogic Tuxedo Connector resources in the Oracle WebLogic Server Console. Oracle Service Bus needs to keep the WebLogic Tuxedo Connector resources it uses in sync. Modifications to those WebLogic Tuxedo Connector resources in the Oracle WebLogic Server Console can cause those resources to become out of sync with Oracle Service Bus, and a re-import of those services into Oracle Service Bus results in service activation failure.

Use the following guidelines for using and configuring WebLogic Tuxedo Connector resources in Oracle Service Bus:

- Do not modify WebLogic Tuxedo Connector resources in the Oracle WebLogic Server Console that you use in Oracle Service Bus proxy and business services. Modify the WebLogic Tuxedo Connector configuration in your Oracle Service Bus service configurations.
- If the WebLogic Tuxedo Connector configurations do become out of sync between your Tuxedo services in Oracle Service Bus and the Oracle WebLogic Server Console, the easiest way to get back in sync is to delete the WebLogic Tuxedo Connector resources in the Oracle WebLogic Server Console and re-configure or re-import the Tuxedo services in Oracle Service Bus.

To use Tuxedo services from Oracle Service Bus, follow the instructions in [Section 35.3.1, "Configuring a New Business Service."](#) To use Oracle Service Bus services from Tuxedo, see [Section 35.4.1, "Adding and Configuring a Proxy Service."](#)

35.3 Using Tuxedo Services from Oracle Service Bus

The following sections describe how to use Tuxedo services from Oracle Service Bus:

- [Section 35.3.1, "Configuring a New Business Service"](#)
- [Section 35.3.2, "Load Balancing and Failover"](#)
- [Section 35.3.3, "Handling Errors"](#)
- [Section 35.3.4, "Testing Your Configuration"](#)

35.3.1 Configuring a New Business Service

To use Tuxedo services from Oracle Service Bus, you must configure a new business service in the Oracle Service Bus Console. For more information about business services, see [Section 2.2, "Working with Business Services."](#)

Log in to the Oracle Service Bus Console. Perform the configuration steps in the order presented, using the instructions in the following sections.

- [Section 35.3.1.1, "Add a New Project"](#)
- [Section 35.3.1.2, "Add a Business Service"](#)

35.3.1.1 Add a New Project

Follow these steps:

1. In the Change Center, click **Create** to create a new session or click **Edit** to enter an existing session.

You must be in a session to edit resources.

2. Select **Project Explorer**.
3. Enter a name for the new project and click **Add Project**.

A message at the top of the page indicates that the project was added successfully.

35.3.1.2 Add a Business Service

Follow these steps:

1. Click the name of the newly created project.
2. In the **Create Resource** field, select **Business Service**.
3. On the **Create a Business Service – General Configuration** page, enter the following values:

Service Name – The name of the service.

Service Type – Select **Any XML Service** (the default).

Note: Tuxedo transport only supports **Any XML Service** and **Messaging Service** service types.

4. Click **Next**.
5. On the **Create a Business Service – Transport Configuration** page, enter the following values:

Protocol – Select **tuxedo**.

Load Balancing Algorithm – Keep the default or select another algorithm.

Endpoint URI – Specify one or more endpoint URIs, using one of the following formats, then click **Add**.

 - The URI format for outbound calls to Tuxedo services is `tuxedo:resourcename[/remotename]` where,
 - `resourcename` corresponds to a WTC Import service name; `resourcename` is required.

- remotename corresponds to the service name exported by the remote Tuxedo domain; remotename is optional.

If more than one URI is specified, you must have unique resource names for the endpoints. If no remote name is specified, its value is the value of the resource name. If no remote name is entered or if the remote and resource name are the same, only one URI is allowed. This allows already defined WTC Import services to use WTC load balancing and failover.

Note: If you configure two identical URIs, an error displays notifying you that the service name already exists.

The Tuxedo transport uses the resource name and remote name from the URI to dynamically create a WTC Import service.

- The URI format for outbound calls to Tuxedo resources of type /Q is `tuxedo-queue:sendQspace/sendQname [/ [rcvQspace:] /rcvQname] [/failureQname]` where,
 - tuxedo-queue indicates that /Q calls will be made.
 - sendQspace corresponds to the unique name of the queue space in the Tuxedo domain; sendQspace is required.
 - sendQname corresponds to the queue name in the queue space in which requests will be stored; sendQname is required.

The following two values are optional. If you specify neither one, the run time will return immediately and not expect a response. Also, the **Response Required?** option on the **Tuxedo Transport Configuration** page will be unavailable.

If you specify either value and do not select the **Response Required?** option, the values you specified will be ignored.

- rcvQspace corresponds to the unique name of the queue space in the Tuxedo domain from which replies will be received; rcvQspace is optional. If not specified, the sendQspace value is used.
- rcvQname corresponds to the name of the queue in the Tuxedo domain from which replies will be received; rcvQname is optional.

The last value is also optional. If you specify neither `rcvQspace` nor `rcvQname`, but specify `failureQname`, the URI format is `tuxedo-queue:sendQspace/sendQname//failureQname`

- failureQname corresponds to the name of the queue in the Tuxedo domain where error messages will be stored; failureQname is optional.

Note: When a response is expected, it occurs in the same thread that sends the request.

6. Click **Next**.

7. On the **Tuxedo Transport Configuration** page, enter the following values:

Field Table Classes – Optional. Enter the name of the class or classes describing the FML/FML32 buffer received. These are used for the FML/FML32-to-XML

conversion routines to map field names to element names. This is a space-separated list of fully qualified class names.

View Classes – Optional. Enter the name of the class or classes describing the VIEW/VIEW32 buffer received or sent. These are used for the VIEW/VIEW32-to-XML conversion routines to map field names to element names. This is a space-separated list of fully qualified class names.

Classes Jar – Select a JAR resource that contains a JAR file with the FML/FML32 or VIEW/VIEW32 classes necessary for this endpoint operation.

Remote Access Point – From the list, select a remote access point that is associated with the WTC Import service. The list contains remote access points configured in WTC. You cannot create a business service without an associated remote access point.

If no remote access points exist or to create a new one, select **New**. Enter the corresponding **Access Point Name** and **Network Address** in the adjacent fields. Upon validation of the endpoint, the access point is added to the WTC configuration for each WTC server. If no WTC server exists, one is created.

You can enter an existing access point name after selecting the **New** option. This causes the existing information to be updated with the new parameters. You can change only the host name and port number.

If more than one URI has been specified, there will be one remote access point field per URI and the URI displays for informative purposes. If more than one URI exists, each requires a different remote access point. If the URI specified already corresponds to an existing WTC resource, the corresponding remote access point displays, but cannot be modified.

Local Access Point – This field appears only when you select **New** in the **Remote Access Point** field. From the list, select a local access point to be associated with the newly created remote access point. If none exist or to create a new one, select **New**. Enter the corresponding **Local Access Point Name** and **Local Network Address** in the adjacent fields.

Note: Access points are not deleted by the transport when the endpoints are removed, since they may be used by multiple endpoints. To remove access points, use the Oracle WebLogic Server Administration Console.

Request Buffer Type – From the list, select the type of buffer that the remote Tuxedo service will receive.

Request Buffer Subtype – This field is enabled if the Request Buffer Type value is VIEW or VIEW32. Enter the buffer subtype with which to associate the request buffer.

Response Required? – Select the check box to indicate a bidirectional call. If not selected, the underlying `tpcall` is invoked with `TPNOREPLY` flag, and a null response is posted asynchronously.

Suspend Transaction – Select the check box to suspend the transaction, if one exists. This is useful when the remote service does not support transactions.

When making calls to Tuxedo resources of the type `/Q`, use the **Suspend Transaction** option whether or not you expect a reply. A successful return from a one-way call means that a message has been successfully queued.

Note: Tuxedo transports to /Q mode endpoints are considered asynchronous transactional, if the **Suspend Transaction** option is not selected. This allows the framework to prevent a deadlock situation.

In /Q mode, when an endpoint expects a reply, multiple threads on multiple Managed Servers may reply using the same destination. For this reason, at run time, when a reply is expected, a unique correlation ID is sent along with the request.

The dequeue operation then waits for the message containing that correlation ID. Correlation IDs are composed in the same manner as those used by JMS transports in similar situations.

Dispatch Policy – From the list, select a Oracle WebLogic Server Work Manager, if available. The default Work Manager is used if no other one exists. The Work Manager asynchronously posts a null reply in the case of a one-way invocation.

Request Encoding – Specify a character set encoding for requests in Tuxedo transports.

Response Encoding – Specify a character set encoding for responses in Tuxedo transports.

Transformation Style – The ordering or grouping of elements when FML or FML32 buffers are transformed into XML. Select one of the following choices:

- **None:** (default) The order of fields may not be respected.
- **Ordered:** The fields will be presented with all their occurrences in the correct order.
- **Ordered and Grouped:** If the fields are logically structured as records, the fields will be ordered by occurrence and also grouped by record.

Timeout – Specify the maximum amount of time (in seconds) that the transport run time waits for replies; an integer value that is greater than or equal to 0. If not specified or set to zero (default), replies will time out at BLOCKTIME, the maximum number of seconds that the local Tuxedo access point allows for a blocking call.

At run time, replies exceeding the time-out value are ignored and an error message with a TPETIME exception is returned.

Specify the timeout value only for request/response services. For /Q or one-way endpoints, the **Timeout** field is unavailable. If the outbound call is part of a transaction, the timeout value is ignored.

Note: The WTC BLOCKTIME value takes precedence if it is less than the time-out value. For example, if the endpoint time-out value is 30 seconds and the WTC BLOCKTIME is 20 seconds, calls will time out in 20 seconds.

8. Click **Finish**.
9. On the **Summary** page, click **Save**.

35.3.2 Load Balancing and Failover

When specifying a business service and defining the endpoint URIs, entering a remote name that is different from the resource name allows you to use the Oracle Service Bus load balancing and failover capabilities. In this case, you can define multiple service names and associate them to a service that is replicated across multiple remote domains. The resource name must be unique, but remote names do not have the same restriction.

35.3.3 Handling Errors

You can configure tuxedo-transport business services to handle application and communication errors as follows:

- Application errors—you can specify whether or not to retry business service endpoint URIs when application errors occur. See "Retry Application Errors" in [Section 4.2.4, "Business Service Transport Configuration Page."](#)
- Communication errors—you can configure business service URIs to be taken offline when communication errors occur. See Enable business service endpoint URIs to be taken offline in "Configuring Operational Settings for Business Services" and "Viewing Business Services Endpoint URIs Metrics" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

[Table 35–1](#) describes the following Tuxedo exceptions and the type of Oracle Service Bus error they denote.

Table 35–1 Tuxedo Exceptions

Exception	Description
TPESVCFail	The service failed—an application error
TPENOENT	The requested entity does not exist—a communication error
TPEPERM	A permissions error has occurred—a communication error

35.3.4 Testing Your Configuration

Once you have configured Oracle Service Bus to work with Tuxedo, you can test the configuration using the test console in the Oracle Service Bus Console.

The following list of tasks summarizes the process of testing outbound usage of Tuxedo by Oracle Service Bus.

1. Build and start the Tuxedo servers.
2. Set up a Tuxedo service to call the Oracle Service Bus proxy service.
3. In the Oracle Service Bus Console, click **Activate** under Change Center, to enable the test console.
4. In the **Project Explorer**, click the **Launch Test Console** icon associated with the business service you want to test.
5. Enter a payload in the test console. For more information, see "Testing Business Services" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.
6. Click **Execute**.

A response page displays the results of the service request.

35.4 Using Oracle Service Bus from Tuxedo

The following sections describe how to use Oracle Service Bus services from Tuxedo:

- [Section 35.4.1, "Adding and Configuring a Proxy Service"](#)
- [Section 35.4.2, "Testing Your Configuration"](#)

35.4.1 Adding and Configuring a Proxy Service

To use Oracle Service Bus services from Tuxedo, you must configure a new proxy service using the Oracle Service Bus Console. For more information about proxy services, see [Section 2.3, "Working with Proxy Services."](#)

Log in to the Oracle Service Bus Console and perform these steps in the order presented.

To complete this configuration, you will perform the tasks described in the following sections:

- [Section 35.4.1.1, "Add a New Project"](#)
- [Section 35.4.1.2, "Add a Proxy Service"](#)
- [Section 35.4.1.3, "Configure the Proxy Service"](#)

35.4.1.1 Add a New Project

Follow these steps:

1. In the Change Center, click **Create** to create a new session or click **Edit** to enter an existing session.
You must be in a session to edit resources.
2. Select **Project Explorer**.
3. Enter a name for the new project and click **Add Project**.

A message at the top of the page indicates that the project was added successfully.

35.4.1.2 Add a Proxy Service

Follow these steps:

1. Click the name of the newly created project.
2. In the **Create Resource** field, select **Proxy Service**.
3. On the **Create a Proxy Service – General Configuration** page, enter the following values:

Service Name – The name of the service

Service Type – Select **Any XML Service** (the default).

Note: Tuxedo Transport only supports **Any XML Service** and **Messaging Service** service types.

4. Click **Next**.
5. On the **Create a Proxy Service – Transport Configuration** page, enter the following required values:

Protocol – Select **tuxedo**.

Endpoint URI - Enter a service name that corresponds to the endpoint URI on the Tuxedo server where the service was deployed.

6. Click **Next**.
7. On the **Tuxedo Transport Configuration** page, enter the following values:

Field Table Classes – Optional. Enter the name of the class or classes describing the FML/FML32 buffer received. These are used for the FML/FML32-to-XML conversion routines to map field names to element names. This is a space-separated list of fully qualified class names.

View Classes – Optional. Enter the name of the class or classes describing the VIEW/VIEW32 buffer received or sent. These are used for the VIEW/VIEW32-to-XML conversion routines to map field names to element names. This is a space-separated list of fully qualified class names.

Note: X_C_TYPE and X_COMMON Tuxedo buffer types are handled in the same manner as VIEW/VIEW32 buffers.

If an incoming request contains VIEW buffers, then specify the corresponding VIEW classes in the Oracle Service Bus CLASSPATH.

Classes Jar – Select a JAR resource that contains a JAR file with the FML/FML32 or VIEW/VIEW32 classes necessary for this endpoint operation.

Local Access Point – From the list, select a local access point that is associated with the WTC Export service. The list contains local access points configured in WTC. You cannot create a proxy service without an associated local access point.

If no local access points exist or to create a new one, select **New**. Enter the corresponding **Local Access Point Name** and **Local Network Address** in the adjacent fields. Upon validation of the endpoint, the access point is added to the WTC configuration for each WTC server. If no WTC server exists, one is created.

You can enter an existing access point name after selecting the **New** option. This causes the existing information to be updated with the new parameters. You can change only the host name and port number.

Remote Access Point – This field appears only when you select **New** in the **Local Access Point** field. From the list, select a remote access point to be associated with the newly created local access point. If none exist or to create a new one, select **New**. Enter the corresponding **Access Point Name** and **Network Address** in the adjacent fields.

The remote access point will also be the authentication principal for the WTC connection for inbound requests. Optionally, you can create a user with the same access point ID in the default security realm to allow incoming calls.

To do so, select **Yes** from the **Create User?** field. The password is randomly generated using a temporary variable to avoid security issues.

Note: Access points and users are not deleted by the transport when the endpoints are removed, since they may be used by multiple endpoints. To remove access points, use the Oracle WebLogic Server Administration Console.

Reply Buffer Type – From the list, select the type of buffer that the remote Tuxedo client will receive. This field is enabled if the **Response Required** field is selected.

Reply Buffer Subtype – This field is enabled if the Request Buffer Type value is VIEW or VIEW32. Enter the buffer subtype with which to associate the reply buffer.

Response Required? – Select the check box if this service is expected to send a response. The default status is selected. However, if the service type is **Messaging Service** and the response type is **None** then it is not selected; in this case, the field is not enabled.

Request Encoding – Specify a character set encoding for requests in Tuxedo transports.

Response Encoding – Specify a character set encoding for responses in Tuxedo transports.

Transformation Style – The ordering or grouping of elements when FML or FML32 buffers are transformed into XML. Select one of the following choices:

- None: (default) The order of fields may not be respected.
- Ordered: The fields will be presented with all their occurrences in the correct order.
- Ordered and Grouped: If the fields are logically structured as records, the fields will be ordered by occurrence and also grouped by record.

8. Click **Finish**.

9. On the **Summary** page, click **Save**.

35.4.1.3 Configure the Proxy Service

Oracle Service Bus message flows define the implementation of proxy services. Message flows can include zero or more pipeline pairs: Request and Response Pipelines for the proxy service (or for the operations on the service); and Error Handlers that can be defined for Stages, Pipelines, and proxy services. Pipelines can include one or more Stages, which in turn include actions.

For more information about creating, editing, and viewing message flows, see [Section 2.4, "Working with Proxy Service Message Flows"](#) and "Modeling Message Flow in Oracle Service Bus" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

The following sections provide one example of how to change the routing behavior of a proxy service and edit the message flow by:

- Adding a Route Node
- Configuring an action to route the proxy service to a business service

Follow these steps:

1. In the Oracle Service Bus Console, select **Resource Browser > Proxy Services**.
2. Click the **Edit Message Flow** icon in the row of the proxy service you created in [Section 35.4.1.2, "Add a Proxy Service."](#)

The **Edit Message Flow** page displays the default message flow configuration which consists of a single **Proxy Service** icon that signifies the starting node for the service. This is the minimum configuration of a message flow. The behavior of the message flow is sequential.

3. Click the **Proxy Service** icon and select **Add Route**.
4. Click the **Route Node** icon, then click **Edit Name and Description**. Change the name and description, as desired, then click **Save**.

In the message flow, the name of the node changes to display the Route Node name.

5. Click the named **Route Node** icon and select **Edit Route**.

On the **Edit Stage Configuration** page, the only object displayed is the **Add an Action** icon.

A Stage is an element of a pipeline and it is a container for actions defined in a pipeline. Actions are the elements of a pipeline stage that define the handling of messages as they flow through a proxy service.

6. Click the **Add an Action** icon, then select **Add an Action > Communication > Routing**.

The **Edit Stage Configuration** page changes to display the contents of the action. The contents of the action are defined by the type of node you created—a Route Node.

7. Click **Service**.

The Service Browser displays the proxy services and business services that you created.

8. Select the business service that you want to expose to Tuxedo.

9. Click **Submit**.

The display updates to show routing to the business service.

The configuration is completed and ready to test.

35.4.2 Testing Your Configuration

Once you have configured Tuxedo to work with Oracle Service Bus, you can perform a test to verify that it is working correctly. If you are using XML-to-FML32 and FML32-to-XML conversions, you can test this configuration using the `ud32` Tuxedo client program that is included with Tuxedo. (If you are using FML conversions, you can use the `ud` client.) `ud32` reads input consisting of the text representation of FML buffers.

If you are not using XML-to-FML and FML-to-XML conversions, you must develop a test client program in Tuxedo to test this configuration.

35.5 Tuxedo Transport Buffer Transformation

Oracle Service Bus and Tuxedo can interoperate to use the services that each product offers, which can include buffer transformation. The Oracle Service Bus service type and the Tuxedo buffer type determine how transformation occurs.

Tuxedo transport supports **Any XML Service** and **Messaging Service** service types in Oracle Service Bus.

- **Any XML Services (Non SOAP)**—The messages to XML-based services are XML, but can be of any Tuxedo buffer type allowed by the service configuration.

- **Messaging Services**—Messaging services are those that can receive messages of one data type and respond with messages of a different data type. The supported data types include XML, MFL, text, and untyped binary.

The following sections explain how the Tuxedo transport handles buffer transformation.

- [Section 35.5.1, "Any XML Service Type"](#)
- [Section 35.5.2, "Messaging Service Type"](#)

35.5.1 Any XML Service Type

[Table 35–2](#) shows the behavior of the Tuxedo transport depending on the Tuxedo buffer type when the service type is **Any XML Service**.

Note: For **Any XML Service**, the payload must be a well-formed XML document.

Table 35–2 Buffer Transformation for Any XML Service

Tuxedo Buffer Type	Tuxedo Transport Behavior
STRING	Passes the buffer as is.
CARRAY	Passes the buffer as is.
X_OCTET	
FML/FML32	Converts the buffer to and from XML using the configured field classes.
XML	Passes the buffer as is. Note: The Tuxedo application <i>should not</i> send NULL bytes when the Tuxedo buffer is XML.
VIEW/VIEW32	Converts the buffer to and from XML using the configured view classes.
X_C_TYPE	
X_COMMON	
MBSTRING	Passes the buffer as is. Note: Encoding is determined by the <code>encoding</code> element of the <code>MetaData</code> of the message sent or received.

35.5.2 Messaging Service Type

[Table 35–3](#) shows the behavior of the Tuxedo transport depending on the Tuxedo buffer type when the service type is **Messaging Service**.

Note: If **None** is specified in the subtype, the Tuxedo transport should not receive any buffer.

Table 35–3 Buffer Transformation for Messaging Service

Tuxedo Buffer Type	Binary Messaging Type	Text Messaging Type	MFL Messaging Type	XML Messaging Type
STRING	Passed as is	Passed as is	Passed as is, assuming the buffer is in a suitable format. If not, the transport returns an error.	XML
CARRAY	Passed as is	Not supported	Passed as is, assuming the buffer is in a suitable format. If not, the transport returns an error.	XML
FML/FML32	Passed as is	Not supported	Not supported	XML
XML	Passed as is	Passed as is	Not supported	XML
VIEW/VIEW32	Passed as is	Not supported	Not supported	XML
MBSTRING	Passed as is	Passed as is	Passed as is, assuming the buffer is in a suitable format. If not, the transport returns an error.	XML

The Tuxedo transport handles the buffer manipulation the same way as if the service was Any XML service type.

35.6 Tuxedo Transport Transaction Processing

Oracle Service Bus and Tuxedo can interoperate to use the services that each product offers, which often includes transaction processing. Tuxedo transport takes advantage of transactions or starts transactions in Oracle Service Bus.

Note: The exception to this transaction support is when the inbound transport is Tuxedo with a transactional message and the outbound is request/response XA-JMS. In this case, Oracle Service Bus detects this exception and it results in a TPESYSTEM error.

The Tuxedo transport transactional behavior is driven by the Quality of Service (QoS) setting available at the message context level. For more information, see "Quality of Service" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

The following sections explain how the Tuxedo transport handles transactions.

- [Section 35.5.1, "Any XML Service Type"](#)
- [Section 35.6.2, "Outbound Services"](#)

35.6.1 Inbound Services

When a transactional context is received, the message going into the pipeline sets the QoS to **Exactly Once**, otherwise QoS is set to **Best Effort**.

When a `TransportException` is caught before the reply is sent back to the client, the request aborts by throwing a `TPESYSTEM` exception and a transaction rollback results.

35.6.2 Outbound Services

When the thread calling the business service has a transactional context, the Tuxedo transport behaves in the following manner:

- If QoS is set to **Exactly Once**, the Tuxedo transport automatically forwards the transactional context to the remote domain unless the endpoint is configured to suspend the transaction.
- If QoS is set to **Best Effort**, the Tuxedo transport suspends the transaction before making the call and resumes it after the call. This is equivalent to making an ATMI call with `TPNOTRAN` flag set.

When the thread calling the business service has no transaction associated, the Tuxedo call occurs non-transactionally, regardless of the QoS setting. In this case, it will correspond to a `tpcall()` or `tpacall()` with the `TPNOTRAN` flag set.

DSP and Oracle Data Service Integrator Transport

Note: The DSP transport is being deprecated and will be removed in future releases.

For information on supported Oracle Service Bus interoperability with Oracle Data Service Integrator, see the "Oracle Fusion Middleware Supported System Configurations" at http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html.

Oracle Data Service Integrator can be accessed through an Oracle Service Bus transport. In this way Oracle Service Bus can make full use of data services. This approach also allows a more efficient and flexible approach to accessing data services as compared with exposing such services as Web services.

Note: The Oracle name for AquaLogic Data Services Platform (ALDSP) Oracle Data Service Integrator. The new product name is used in this document unless procedural accuracy requires using ALDSP terminology. The name of the Oracle Service Bus transport remains "DSP."

For more information about Oracle Data Service Integrator, see the documentation at http://download.oracle.com/docs/cd/E13162_01/odsi/docs10gr3/index.html.

This section contains the following topics:

- [Section 36.1, "Enabling Data Services for Oracle Service Bus"](#)
- [Section 36.2, "Using the DSP Transport"](#)

36.1 Enabling Data Services for Oracle Service Bus

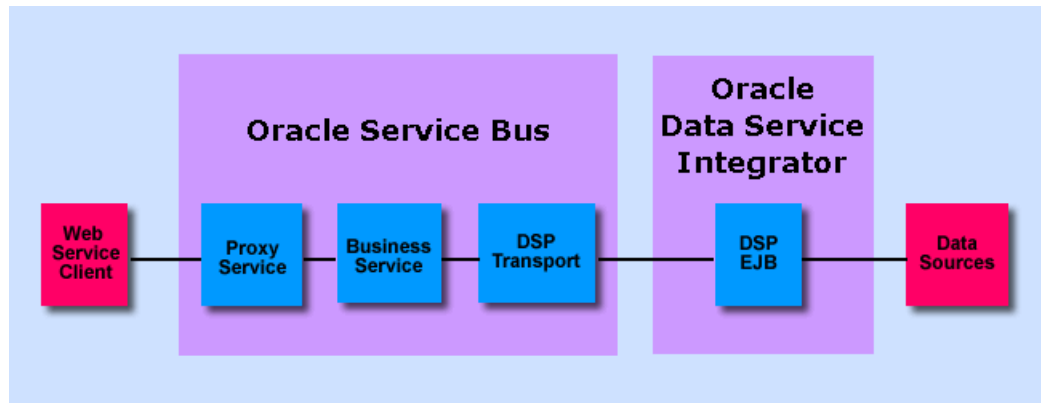
To make an Oracle Data Service Integrator data service available to an Oracle Service Bus client you need to:

- Generate a WSDL file for your data service and import the new WSDL into Oracle Service Bus using the recommended procedure.
- Configure or create a business service based on the WSDL.
- Configure or create a proxy service based on the business service.

After you have completed these tasks, you can invoke data services through Oracle Service Bus.

Figure 36–1 shows a client accessing a data source through Oracle Service Bus and Oracle data Service Integrator.

Figure 36–1 From Data Source to Web Service Client



36.2 Using the DSP Transport

The example in this section illustrates use of a data service in Oracle Service Bus.

36.2.1 Actions Needed Within Oracle Data Service Integrator

Perform the following steps in Oracle Data Service Integrator.

36.2.1.1 Step 1. Start Your Server

Start the Oracle Data Service Integrator server if it is not already running. (For the purpose of this discussion, the sample RetailDataspac provided with Oracle Data Service Integrator is used.)

36.2.1.2 Step 2. Generate a WSDL from the Data Service

You can generate a WSDL from your data service in two ways.

- Option A: Generating a WSDL file using Data Services Studio – For detailed instructions, see "How To Generate a Web Service Map and WSDL from a Data Service" at http://download.oracle.com/docs/cd/E13162_01/odsi/docs10gr3/datasrvc/Generate%20a%20Web%20Service%20Map%20from%20a%20Data%20Service.html.
- Option B: Export a WSDL Through the Oracle Data Service Integrator Console.

If a WS file is available, you can use the Oracle Data Service Integrator Console to generate a WSDL file.

 1. Launch the Oracle Data Service Integrator Console (<http://host:port/dspconsole>).
 2. Log in. The sample uses **weblogic** as both the username and password.
 3. Click **Service Explorer**.
 4. Navigate to the Web service map (example: RetailWebServices.ws) corresponding to the data service for which you want to create the WSDL file.
 5. Click **View WSDL Definition** in the General tab. The Oracle Data Service Integrator Console opens a new window and displays the WSDL definition.

6. Using a text editor, copy and paste the WSDL definition into a new text document and save the WSDL file.

See "Getting Started with Oracle Data Service Integrator Administration" at http://download.oracle.com/docs/cd/E13162_01/odsi/docs10gr3/admin/getting_strtd.html.

7. If you have not already done so, build your data service application (deployment is automatic with a build). A deployed application is needed in order for a client processes to access data through your data services.

36.2.1.3 Step 3: Obtaining the Web Service Address

The URL address to the WSDL is needed. To obtain this address in Data Services Studio:

1. Right-click on the WS file (example: OrderService.ws)
2. Select **Test Web Service**.
3. When the Test Client opens, save the URL address. Following is the address for the OrderService example:

`http://localhost:7001/RetailDataspace/RetailApplication/OrderManagement/OrderService.ws?WSDL`

36.2.2 Actions Needed Within Oracle Service Bus

Perform the following steps in Oracle Service Bus.

36.2.2.1 Step 4: Import the Data Service WSDL into Oracle Service Bus

The following generally describes the steps needed to import a WSDL generated in Oracle Data Service Integrator into Oracle Service Bus.

1. Start the Oracle Service Bus server if it is not already running. For the purposes of this example, the Oracle Service Bus example server and the Default project is used.
2. Launch the Oracle Service Bus Console (<http://host:port/sbconsole>).
3. Log in. The default user name and password is **weblogic**.
4. Click Project Explorer.
5. Enter a new project name (example: dataServiceTest).
6. Click on the name of your new project.
7. In the Project Explorer, bulk import the WSDL and schema resources.

Note: You can also import the resources into the Oracle Service Bus IDE, as described in [Section 2.1.14, "Importing Resources."](#)

36.2.2.2 Step 5: Create the Business Service

Create a business service from a WSDL imported from Oracle Data Service Integrator. Use the following guidance to configure the business service:

- Select **WSDL Web Service** for the Service Type, browse to the imported WSDL, and select the appropriate port or binding; for example, OrderServiceSoapBinding.
- Select the **dsp** transport.

Figure 36–2 Request and Response from the Oracle Service Bus Test Console

Proxy Service Testing - orderService2 Help

Back Close

Request Document

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  </soap:Header>
  <soapenv:Body>
    <ord:getOrderByCustID xmlns:ord="Id:RetailApplication/OrderManagement/OrderService.ws">
      <ord:custID>CUSTOMER3</ord:custID>
    </ord:getOrderByCustID>
  </soapenv:Body>
</soapenv:Envelope>
```

Response Document

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <env:Body xmlns:env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <ns:getOrderByCustIDResponse xmlns:ns="Id:RetailApplication/OrderManagement/OrderService.ws">
      <ns0:ORDER xmlns:ns0="urn:retailer">
        <ns1:ORDER_TYPE="APPL" xmlns:ns1="urn:retailerType">
          <OrderID>ORDER_3_0</OrderID>
          <CustomerID>CUSTOMER3</CustomerID>
          <OrderDate>2001-10-01</OrderDate>
          <ShippingMethod>PRIORITY-1</ShippingMethod>
          <HandlingCharge>6.8</HandlingCharge>
          <SubTotal>649.85</SubTotal>
          <TotalOrderAmount>656.65</TotalOrderAmount>
          <SaleTax>0</SaleTax>
          <EstimatedShipDate>2001-10-03</EstimatedShipDate>
          <Status>CLOSED</Status>
          <ShipTo>ADDR_3_0</ShipTo>
          <ShipToName>Britt Pierce</ShipToName>
          <BillTo>CC_3_1</BillTo>
          <TrackingNumber>ORDER_3_00379624444</TrackingNumber>
        <LINE_ITEM>
          <LineItemID>0</LineItemID>
          <OrderID>ORDER_3_0</OrderID>
          <ProductID>APPA_SH_4</ProductID>
          <ProductDescription>Debra Sandal at Nodstrom</ProductDescription>
          <Quantity>1</Quantity>
          <Price>249.95</Price>
          <Status>CLOSED</Status>
        </LINE_ITEM>
        <LINE_ITEM>
          <LineItemID>1</LineItemID>
          <OrderID>ORDER_3_0</OrderID>
          <ProductID>APPA_SH_5</ProductID>
          <ProductDescription>Audrey Hepbun from Farragamo</ProductDescription>
          <Quantity>1</Quantity>
          <Price>299.95</Price>
          <Status>CLOSED</Status>
        </LINE_ITEM>
        <LINE_ITEM>
          <LineItemID>2</LineItemID>
          <OrderID>ORDER_3_0</OrderID>
          <ProductID>APPA_BA_1</ProductID>
          <ProductDescription>Cucci Dejavu Hobo</ProductDescription>
          <Quantity>1</Quantity>
          <Price>99.95</Price>
          <Status>CLOSED</Status>
        </LINE_ITEM>
      </ns0:ORDER>
    </ns:getOrderByCustIDResponse>
  </env:Body>
</soapenv:Envelope>
```


Part VI

Transport SDK

This part provides information on designing, creating, and deploying custom transport providers in Oracle Service Bus using the Transport SDK. Chapters include:

- [Chapter 37, "Introduction"](#)
- [Chapter 38, "Design Considerations"](#)
- [Chapter 39, "Developing a Transport Provider"](#)
- [Chapter 40, "Developing Oracle Service Bus Transports for Eclipse"](#)
- [Chapter 41, "Transport SDK Interfaces and Classes"](#)
- [Chapter 42, "Sample Socket Transport Provider"](#)
- [Chapter 43, "Deploying a Transport Provider"](#)

This chapter describes the purpose of this guide, its intended audience, and general organization. The chapter includes these topics:

- [Section 37.1, "Purpose of this Guide"](#)
- [Section 37.2, "Audience for this Guide"](#)
- [Section 37.3, "Overview of this Guide"](#)

37.1 Purpose of this Guide

This guide provides developers with the information needed to design, create, and deploy a new custom transport provider.

37.2 Audience for this Guide

This guide is written for experienced Java developers who want to add a new custom transport provider to Oracle Service Bus. It is assumed that you have solid knowledge of Web services technologies, Oracle Service Bus, the transport protocol that you want to use with Oracle Service Bus and Oracle WebLogic Server.

37.3 Overview of this Guide

This guide provides developers with the information needed to design, create, and deploy a new custom transport provider.

Design Considerations

Careful planning of development activities can greatly reduce the time and effort you spend developing a custom transport provider. The following sections describe transport provider concepts and functionality to help you get started:

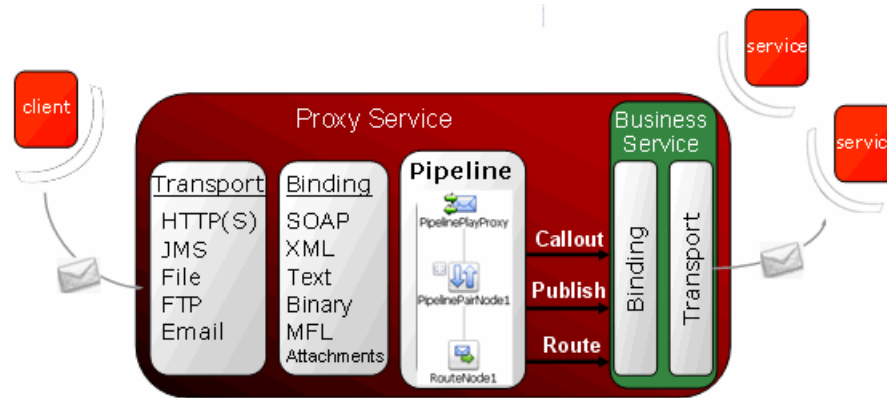
- [Section 38.1, "What is a Transport Provider?"](#)
- [Section 38.2, "What is the Transport SDK?"](#)
- [Section 38.3, "Do You Need to Develop a Custom Transport Provider?"](#)
- [Section 38.4, "Transport Provider Components"](#)
- [Section 38.5, "The Transaction Model"](#)
- [Section 38.6, "The Security Model"](#)
- [Section 38.7, "The Threading Model"](#)
- [Section 38.8, "Designing for Message Content"](#)

38.1 What is a Transport Provider?

A transport provider implements the interfaces of the Transport SDK and provides a bridge between Oracle Service Bus and mechanisms by which messages are sent or received. Such mechanisms can include specific transport protocols, such as HTTP, as well as other entities, such as a file or an e-mail message. A transport provider manages the life cycle and runtime behavior of transport endpoints. An endpoint is a resource where messages originate or are targeted.

[Figure 38-1](#) illustrates the basic flow of messages through Oracle Service Bus. A client sends a message to Oracle Service Bus using a specific transport protocol. A transport provider processes the inbound message, handling communication with the service client endpoint and acting as the entry point for messages into Oracle Service Bus.

Figure 38–1 Message Flow Through Oracle Service Bus



The binding layer, also shown in [Figure 38–1](#), packs and unpacks messages, handles message security, and hands messages off to the Oracle Service Bus Pipeline.

Tip: For more information on Oracle Service Bus message brokering and the role of the transport layer, see *Oracle Fusion Middleware Concepts and Architecture for Oracle Service Bus*. For more detailed sequence diagrams that describe the message flow through Oracle Service Bus, see [Appendix A, "Transport SDK UML Sequence Diagrams."](#)

By default, Oracle Service Bus includes transport providers that support several commonly used transport protocols, such as HTTP, JMS, File, FTP, and others. These native providers let you configure proxy and business services that require these common transport protocols.

Tip: For more information using and configuring native transport providers, see [Part V, "Transports"](#).

38.2 What is the Transport SDK?

This section briefly describes the purpose and features of the Transport SDK. This section includes these topics:

- [Section 38.2.1, "Purpose of the SDK"](#)
- [Section 38.2.2, "Transport SDK Features"](#)
- [Section 38.2.3, "Transport Provider Modes"](#)
- [Section 38.2.4, "Related Features"](#)

38.2.1 Purpose of the SDK

Oracle Service Bus processes messages independently of how they flow into or out of the system. The Transport SDK provides a layer of abstraction between Oracle Service Bus and components that deal with the flow of data in and out of Oracle Service Bus. This layer of abstraction allows you to develop new transport providers to handle unique transport protocols.

The SDK abstracts from the rest of Oracle Service Bus:

- Handling specific transport bindings

- Deploying service endpoints on the transport bindings. An endpoint is either capable of transmitting or receiving a message.
- Collecting monitoring information
- Managing endpoints (such as performing suspend/resume operations and setting connection properties)
- Enforcing Service Level Agreement (SLA) behavior (such as timing out connections)

38.2.2 Transport SDK Features

This section describes the primary features of the Transport SDK.

38.2.2.1 Handling Inbound and Outbound Messages

A transport provider developed with the Transport SDK handles inbound and outbound messages as follows:

- Inbound messages typically come into Oracle Service Bus from an outside source, such as an HTTP client. The Transport SDK packages the payload and transport level headers, if any, into a generic data structure. The Transport SDK then passes the message, in its generic format, to the Oracle Service Bus pipeline.
- Outbound messages originate from Oracle Service Bus business services and go to an externally managed endpoint, such as a Web service or JMS queue. The Transport SDK receives a generic data structure from the Oracle Service Bus pipeline, converts it to the corresponding transport-specific headers and payload, and sends it out to an external system.

The Transport SDK handles outbound and inbound messages independently. An inbound message can be bound to one transport protocol and bound to a different transport protocol on the outbound endpoint.

38.2.2.2 Deploying Transport-Related Artifacts

Certain transports include artifacts that need to be deployed to Oracle WebLogic Server. For instance, a JMS proxy is implemented as a message-driven bean. This artifact, an EAR file, must be deployed when the new JMS proxy is registered. Similarly, the EJB transport provider employs an EAR file that must be deployed when a new EJB business service is registered. Other kinds of artifacts might require deployment, such as a JMS transport, which may create queues and topics as part of the service registration. The SDK allows you to support these artifacts and lets you participate in the WLS deployment cycle. If the deployment of one of these artifacts fails, the Oracle Service Bus session is notified and the deployment is canceled. This feature of the SDK allows for the atomic creation of services. If something fails, the session reverts to its previous state.

Note: To participate in WLS deployment cycle, the transport provider must implement the `TransportWLSArtifactDeployer` interface. The primary benefit of this technique is atomic Oracle WebLogic Server deployment, which can be rolled back if needed. For more information on this interface, see [Section 41.3.2, "Summary of General Interfaces,"](#) and see [Section 39.11, "When to Implement TransportWLSArtifactDeployer."](#)

38.2.2.3 Processing Messages Asynchronously

Because the server has a limited number of threads to work with when processing messages, asynchrony is important. This feature allows Oracle Service Bus to scale to handle large numbers of messages. After a request is processed, the thread is released. When the business service receives a response (or is finished with the request if it is a one-way message), it notifies Oracle Service Bus asynchronously through a callback.

See also [Section 38.5.2, "Support for Synchronous Transactions"](#) and [Section 38.7, "The Threading Model."](#)

38.2.3 Transport Provider Modes

With the Transport SDK, you can implement inbound property modes and outbound property modes. These connection and endpoint modes are specified in the transport provider's XML Schema definition file. For more information on this file, see [Section 39.3.3, "3. Create an XML Schema File for Transport-Specific Artifacts."](#) This schema is available to the Oracle Service Bus Pipeline for filtering and routing purposes.

38.2.4 Related Features

This section lists related features that are provided by the transport manager. The transport manager provides the main point of centralization for managing different transport providers, endpoint registration, control, processing of inbound and outbound messages, and other functions. These features do not require specific support by a transport provider.

38.2.4.1 Load Balancing

The Transport SDK supports load balancing and failover for outbound messages. Supported load balancing options are:

- **None** – For each outbound request, the transport provider cycles through the URIs in the list in which they were entered and attempts to send a message to each URI until a successful send is completed.
- **Round Robin** – Similar to None, but in this case, the transport provider keeps track of the last URI that was tried. Each time a message is sent, the provider starts from the last position in the list.
- **Random** – The transport provider tries random URIs from the list in which they were entered.
- **Weighted Random** – Each URI is associated with a weight. An algorithm is used to pick a URI based on this weight.

38.2.4.2 Monitoring and Metrics

The transport manager handles monitoring metrics such as response-time, message-count, error-count, failover-count, throttling-time, and cache-hit-count.

38.3 Do You Need to Develop a Custom Transport Provider?

This section explains the basic use cases for writing a custom transport provider. In some cases, it is appropriate to choose an alternative approach. This section includes the following topics:

- [Section 38.3.1, "When to Use the Transport SDK"](#)

- [Section 38.3.2, "When Alternative Approaches are Recommended"](#)

38.3.1 When to Use the Transport SDK

One of the prime use cases for the Transport SDK is to support a specialized transport that you already employ for communication between your internal applications. Such a transport may have its own concept of setup handshake, header fields, metadata, or transport-level security. Using the Transport SDK, you can create a transport implementation for Oracle Service Bus that allows configuring individual endpoints, either inbound, outbound or both. With a custom transport implementation, the metadata and header fields of the specialized transport can be mapped to context variables available in a proxy service pipeline.

Use the Transport SDK when the transport provider needs to be seamlessly integrated into all aspects of Oracle Service Bus for reliability, security, performance, management, user interface, and the use of the UDDI registry.

Some cases where it is appropriate to use the Transport SDK to develop a custom transport include:

- Using a proprietary transport that requires custom interfaces and supports an organization's existing applications.
- Using a CORBA or IIOP protocol for communicating with CORBA applications.
- Using other legacy systems, such as IMS and Mainframe.
- Using variations on existing transports, such as SFTP (Secure FTP) and the native IBM WebSphere MQ API (instead of WebSphere MQ JMS).
- Using industry-specific transports, such as LLP, AS3, and ACCORD.
- Using raw sockets, perhaps with TEXT or XML messages. A sample implementation of this type of transport is described in [Chapter 42, "Sample Socket Transport Provider."](#)

Alternatively, you can use the Transport SDK to support a specialized protocol over one of the existing transports provided with Oracle Service Bus. Examples of this could include supporting:

- Messages consisting of parsed or binary XML over HTTP.
- WS-RM or other new Web service standards over HTTP.
- Request-response messaging over JMS, but with a different response pattern than either of the two patterns supported by the Oracle Service Bus JMS transport (for example, a response queue defined in the message context).

38.3.2 When Alternative Approaches are Recommended

Creating a new Oracle Service Bus transport provider using the Transport SDK can be a significant effort. The Transport SDK provides a rich, full featured environment so that a custom transport has all of the usefulness and capabilities of the transports that come natively with Oracle Service Bus. But such richness brings with it some complexity. For certain cases, you might want to consider easier alternatives.

If you need an extension merely to support a different format message sent or received over an existing protocol, it may be possible to use the existing transport and use a Java Callout to convert the message. For example, suppose you have a specialized binary format (such as ASN.1 or a serialized Java object) being sent over the standard JMS protocol. In this case, you might consider defining the service using the standard JMS transport with the service type being a messaging service with binary

input/output messages. Then, if the contents of the message are needed in the pipeline, a Java Callout action can be used to convert the message to or from XML. For information on using Java Callouts, see "Extensibility Using Java Callouts and POJOs" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

Other cases where it is best not to use the Transport SDK to develop a custom transport provider include:

- When combining existing Oracle solutions with Oracle Service Bus satisfies the transport requirement: Oracle WebLogic Server, Oracle WebLogic Integration, Oracle Data Service Integrator, Oracle Business Process Management, Oracle Tuxedo, Oracle WebLogic Portal.
- When service enablement tools, like Eclipse, provide a simpler and more standards-based mechanism to implement SOA practices.
- When alternative connectivity solutions (certified with Oracle Service Bus) also address the requirement. For example: iWay adapters and Cyclone B2B.
- When EJBs can be used instead as a means to abstract some type of simple Java functionality.

38.4 Transport Provider Components

This section presents UML diagrams that depict the runtime and design-time components of a transport provider. This section includes these topics:

- [Section 38.4.1, "Overview"](#)
- [Section 38.4.2, "Design-Time Component"](#)
- [Section 38.4.3, "Runtime Component"](#)

38.4.1 Overview

In general, a custom transport provider consists of a design-time part and a runtime part. The design-time part is concerned with registering endpoints with the transport provider. This configuration behavior is provided by the implementation of the UI interfaces. The runtime part implements the mechanism of sending and receiving messages.

When you develop a new custom transport provider, you need to implement a number of interfaces provided by the SDK. This section includes UML diagrams that model the organization of the design-time and runtime parts of the SDK.

Tip: In Oracle Service Bus, implementations of the `TransportProvider` interface represent the central point for management of transport protocol-specific configuration and runtime properties. A single instance of a `TransportProvider` object exists for every supported protocol. For example, there are single instances of HTTP transport provider, JMS transport provider, and others.

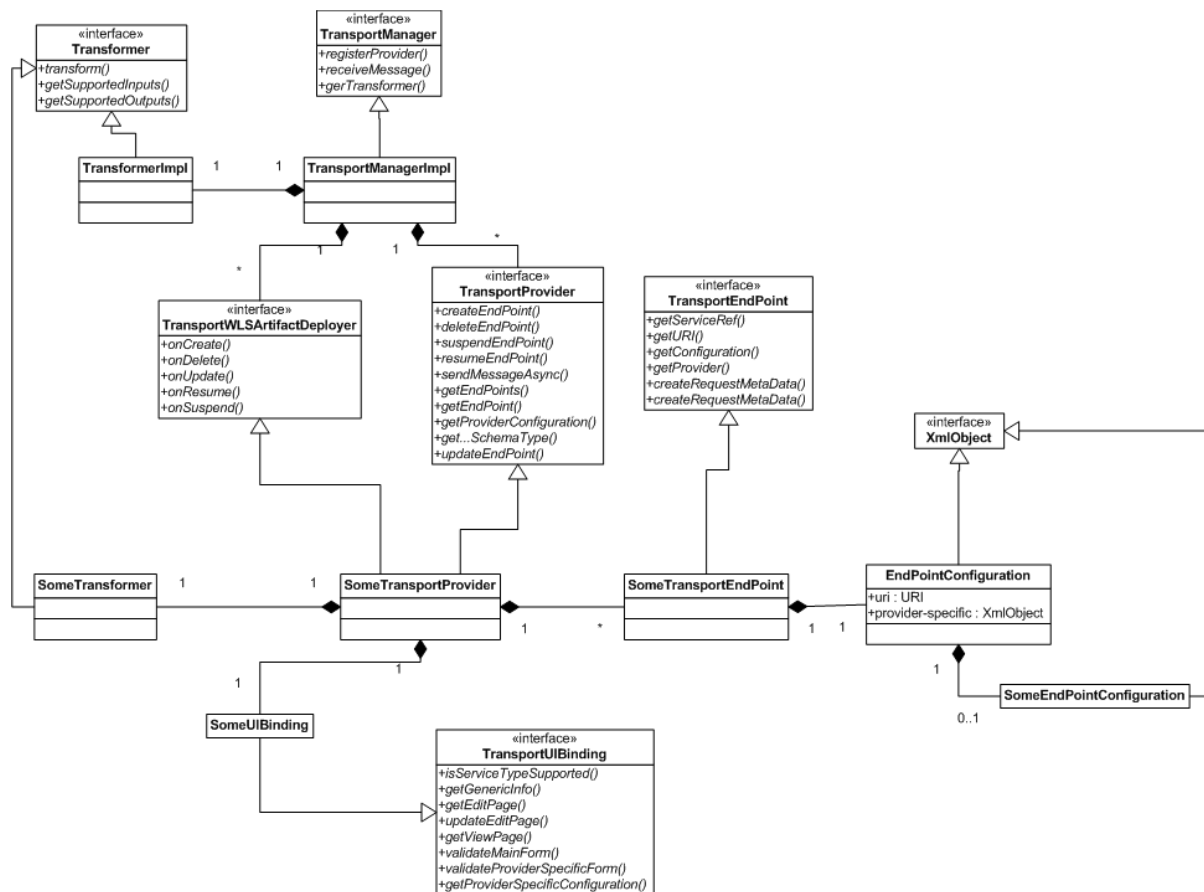
For more information, see [Chapter 39, "Developing a Transport Provider,"](#) for a list of the required interfaces. A summary of the interfaces and classes provided by the Transport SDK are discussed in [Chapter 41, "Transport SDK Interfaces and Classes."](#) Detailed descriptions are provided in the *Oracle Fusion Middleware Java API Reference for Oracle Service Bus*.

38.4.2 Design-Time Component

The design-time part of a custom transport provider consists of the user interface configuration. This configuration is called by the Oracle Service Bus Console or IDE when a new business or proxy service is being registered. Figure 38–2 shows a UML diagram that depicts the structure of the design time part of a transport provider. Some of the interfaces described in the diagram include:

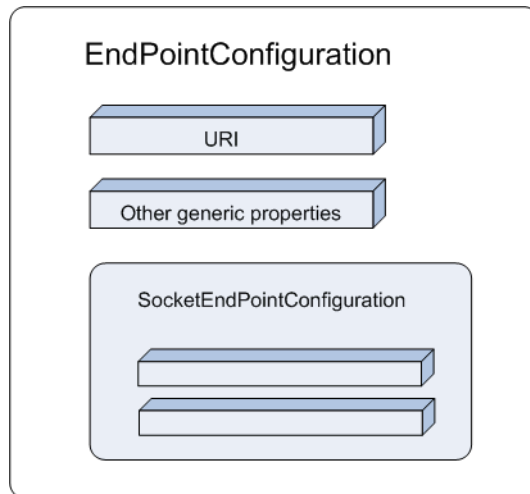
- **TransportManager** – A transport provider communicates with the transport manager through this interface. The implementation is not public.
- **TransportProvider** – Third parties must implement this interface. The TransportProvider keeps track of TransportEndpoint objects. TransportProvider also manages the life cycle of the endpoints. For example, you can suspend a transport endpoint, which is managed through the TransportProvider interface.
- **TransportUIBinding** – Helps the Oracle Service Bus Console render the transport specific pages.

Figure 38–2 Design Time UML Diagram



Note: Each transport endpoint has a configuration that consists of some properties that are generic to all endpoints of any transport provider, such as a URI, and some properties that are specific to endpoints of that provider only. [Figure 38–3](#) shows the relationship between the shared endpoint configuration properties and transport provider specific configuration properties. See [Section 38.5.1, "Overview of Transport Endpoint Properties."](#) for more information.

Figure 38–3 *EndPointConfiguration Properties*



38.4.3 Runtime Component

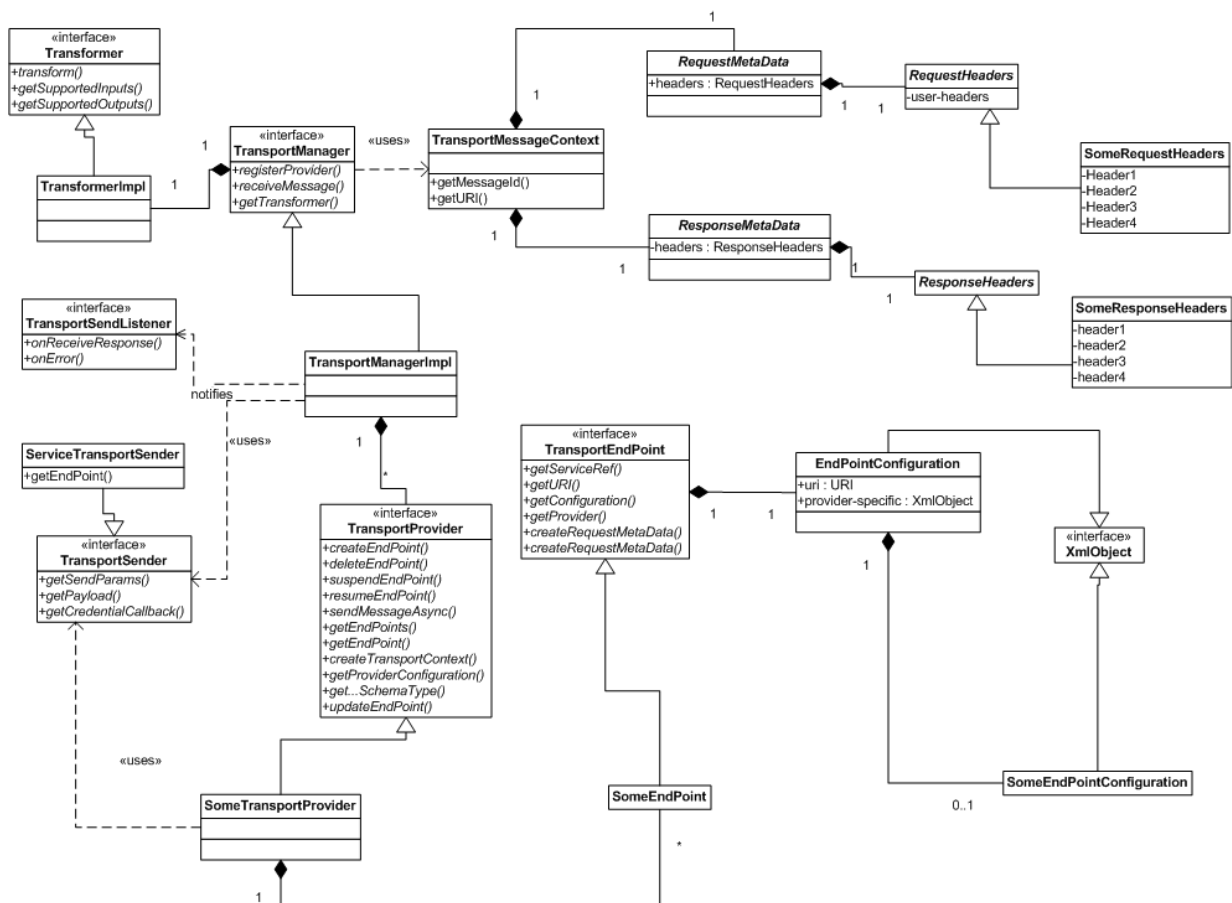
The runtime part of a custom transport provider:

- Receives messages and delivers them to the Oracle Service Bus runtime.
- Delivers outbound messages from Oracle Service Bus runtime to external services.

In the runtime framework, the transport provider calls the transport manager to acknowledge that an inbound message has been received. The transport message context contains the header and body of the inbound message. For the outbound message, there is a `TransportSendListener` and `TransportSender`. The transport provider retrieves the header and body from the message.

[Figure 38–2](#) shows a UML diagram that depicts the structure of the runtime part of a transport provider.

Figure 38–4 Runtime UML Diagram



38.5 The Transaction Model

Before you develop a new transport provider using the Transport SDK, it is important to consider the transaction model for your message endpoints. This section discusses the transaction model used by Oracle Service Bus and how that model relates to the Transport SDK.

This section includes these topics:

- [Section 38.5.1, "Overview of Transport Endpoint Properties"](#)
- [Section 38.5.2, "Support for Synchronous Transactions"](#)

38.5.1 Overview of Transport Endpoint Properties

A transport endpoint is an Oracle Service Bus resource, such as a JMS proxy service, where messages are originated or targeted. In Oracle Service Bus, transport endpoints are managed by protocol-specific transport providers, plug-in objects that manage the life cycle and runtime behavior of transport endpoints.

To understand the transactional model of Oracle Service Bus, it is useful to review some of the properties of service transport endpoints.

38.5.1.1 Transactional vs. Non-Transactional Endpoints

A given endpoint may or may not be transactional. A transactional endpoint has potential to start or enlist in a global transaction context when processing a message. The following examples illustrate how transactional properties vary depending on the endpoint:

- A JMS proxy service that uses the XA connection factory is a transactional endpoint. When the message is received, the container ensures that a transaction is started so that the message is processed in the context of a transaction.
- A Tuxedo proxy service may or may not be a transactional endpoint. A Tuxedo proxy service is only transactional if a transaction was started by the Tuxedo client application before the message is received.
- While an HTTP proxy service will not typically have an associated transaction when invoked by an HTTP client, you can set an option in the HTTP proxy service configuration that starts a transaction and executes the message flow in the context of that transaction.

38.5.1.2 Supported Message Patterns

A given endpoint can use one of the following message patterns:

- **One Way** – No responses are expected. An example of a one-way endpoint is a JMS proxy service that does not expect a response.
- **Synchronous** – A request or response is implied. In this case, the response message is paired with the request message implicitly because no other traffic can occur on the transport channel from the time the request is issued until the time the response is received. In most cases, a synchronous message implies blocking calls for outbound requests. An EJB endpoint is synchronous. An HTTP endpoint is also synchronous: a new request cannot be sent until a response is received.
- **Asynchronous** – A request and response is implied. The response is correlated to a request through a transport-specific mechanism, such as a JMS transport and correlation through a JMSCorrelationID message property. For example, a JMS business service endpoint with request and response is asynchronous.

38.5.2 Support for Synchronous Transactions

All Oracle Service Bus proxy services support transaction propagation, can start a transaction if none already exists, and can optionally ensure that the response occurs in the context of the transaction, even if the outbound business service is asynchronous—in essence transforming an asynchronous pattern effectively into a synchronous pattern. Outbound business services can provide additional transaction support, such as suspending an existing transaction.

Synchronous transactional transports support the following use cases:

38.5.2.1 Use Case 1 (Response Pipeline Processing)

Response pipeline processing is included in an incoming transaction when the inbound transport supports synchronous transactions or when you configure a proxy service to propagate a transaction to the response. This case is supported when the inbound transport is paired with any other outbound transport, with the exception described in the note below.

Note: A deadlock situation occurs when the inbound transport is synchronous transactional and the outbound transport is asynchronous transactional. The deadlock occurs because the outbound request is not available to be received by the business service until after the transaction commits, but the transaction was started externally and does not commit until Oracle Service Bus gets the response and returns. The transport manager recognizes this situation and avoids the deadlock by throwing a runtime error.

For example, if a synchronous transactional inbound endpoint is used, such as a Tuxedo proxy service, and the outbound endpoint is asynchronous transactional, such as a JMS business service, the outbound request does not commit the transaction until the response is received. It cannot be received until the external entity receives the request and processes it.

Also in this case, the Oracle Service Bus Publish action performed in the response pipeline is part of the transaction just like publish actions in the request pipeline are part of the transaction.

Note: There are several actions that can potentially participate in a transaction (in either the request or response pipeline). These include Publish, Service Callout, and Report actions.

For example, if an inbound Tuxedo transport is synchronous transactional, it can be committed only after the request and response pipeline have been completed. In this case, the transport manager transfers the transaction context from the inbound to the outbound thread. When the response thread is finished, the transaction control and outcome are returned to the invoking client.

38.5.2.2 Use Case 2 (Service Callout Processing)

Oracle Service Bus Service Callouts allow you to make a callout from a proxy service to another service. If a Service Callout action is made to a synchronous transactional transport, the case of *Exactly Once* quality of service is supported in addition to *Best Effort* quality of service. *Exactly Once* means that messages are delivered from inbound to outbound exactly once, assuming a terminating error does not occur before the outbound message send is initiated. *Best Effort* means that each dispatch defines its own transactional context (if the transport is transactional). When *Best Effort* is specified, there is no reliable messaging and no elimination of duplicate messages; however, performance is optimized. See also [Section 39.7, "Working with TransportOptions."](#)

Callouts to synchronous transactional transports are optionally part of an existing transaction. For example, while the request pipeline is executing during a global transaction, Service Callouts are permitted to participate in the transaction. For example, if there is a callout to an EJB service, the service can participate in that transaction if it wants to by setting its quality of service value to *Exactly Once*.

For more information on Service Callouts, see [Section 2.4.29, "Adding and Configuring Service Callout Actions in Message Flows."](#)

38.5.2.3 Use Case 3 (Suspending Transactions)

Before calling the transport provider to send an outbound request the transport framework will suspend a transaction if the following conditions apply:

- The outbound service endpoint is transactional.
- There is a global XA transaction in progress.
- The quality of service is set to *Best Effort*.

The suspended transaction will resume, after the "send" operation is complete.

38.5.2.4 Use Case 4 (Multiple URIs)

If a given outbound service endpoint has multiple URIs associated with it, and is transactional, failover only occurs while the transaction, if any, is not marked for rollback. For example, if a URI is called, and the service returns an error, a failover is normally triggered. In this event, the transport framework detects that the transaction has been marked for rollback; therefore, the framework does not perform a failover to a different URI.

38.6 The Security Model

The Transport SDK allows customers and third-parties to plug in new transports into Oracle Service Bus. Within the Oracle Service Bus security model, transport providers are considered trusted code. It is critical that transport provider implementations are carefully designed to avoid potential security threats by creating security holes. Although this document does not contain specific guidelines on how to develop secure transport providers, this section discusses the following security goals of the Transport SDK:

- [Section 38.6.1, "Inbound Request Authentication"](#)
- [Section 38.6.2, "Outbound Request Authentication"](#)
- [Section 38.6.3, "Link-Level or Connection-Level Credentials"](#)
- [Section 38.6.4, "Uniform Access Control to Proxy Services"](#)
- [Section 38.6.5, "Identity Propagation and Credential Mapping"](#)

38.6.1 Inbound Request Authentication

Transport providers are free to implement whatever inbound authentication mechanisms are appropriate to that transport. For example: the HTTP transport provider supports these authentication methods:

- HTTP BASIC
- Custom authentication tokens carried in HTTP headers

The HTTPS transport provider supports SSL client authentication, in addition to the ones listed above. Both HTTP and HTTPS transport providers also support anonymous client requests.

The transport provider is responsible for implementing any applicable transport level authentication schemes, if any. If the transport provider authenticates the client it must make the client Subject object available to Oracle Service Bus by calling `TransportManager.receiveMessage()` within the scope of `weblogic.security.Security.runAs(subject)`. For information on this method, see the *Oracle Fusion Middleware Java API Reference for Oracle Service Bus*.

Tip: For information on the Java class Subject, see <http://java.sun.com/j2se/1.5.0/docs/api/javax/security/auth/Subject.html>.

The proxy will use this Subject in the following ways:

- During access control to the proxy service
- To populate the message context variable `$inbound/ctx:security/ctx:transportClient/*`
- As the input for identity propagation and credential mapping (unless there is also message-level client authentication)

If the transport provider does not support authentication, or if it supports anonymous requests, it must make sure the anonymous subject is on the thread before dispatching the request. Typically the transport provider will already be running as anonymous, but if this is not the case, then the provider must call:

```
Subject anonymous = SubjectUtils.getAnonymousUser()
Security.runAs(anonymous, action)
```

The transport provider is also responsible for providing any Oracle Service Bus Console configuration pages required to configure inbound client authentication.

The transport provider must clearly document its inbound authentication model.

38.6.2 Outbound Request Authentication

Transport providers are free to implement whatever outbound authentication schemes are appropriate to that transport. The transport SDK includes APIs to facilitate outbound username/password authentication, (two-way) SSL client authentication, and JAAS Subject authentication.

38.6.2.1 Outbound Username/Password Authentication

Outbound username/password authentication can be implemented by leveraging Oracle Service Bus service accounts. Service accounts are first-class, top-level Oracle Service Bus resources. Service accounts are created and managed in the Oracle Service Bus Console. Transport providers are free to design their transport-specific configuration to include references to service accounts. That way the transport provider can make use of the credential management mechanisms provided by Oracle Service Bus service accounts.

Transport providers don't have to worry about the details of service account configuration. There are three types of service accounts:

- **Static** – A static service account is configured with a fixed username/password.
- **Mapped** – A mapped service account contains a list of remote-users/remote-passwords and a map from local-users to remote-users. Mapped service accounts can optionally map the anonymous subject to a given remote user.
- **Pass-through** – A pass-through service account indicates that the username/password of the Oracle Service Bus client must be sent to the back-end.

An outbound endpoint can have a reference to a service account. The reference to the service account must be stored in the transport-specific endpoint configuration. When a proxy service routes a message to this outbound endpoint, the transport provider

passes the service account reference to `CredentialCallback.getUsernamePasswordCredential(ref)`. Oracle Service Bus returns the username/password according to the service account configuration. This has the advantage of separating identity propagation and credential mapping configuration from the transport-specific details, simplifying the transport SDK. It also allows sharing this configuration. Any number of endpoints can reference the same service account.

Note: The `CredentialCallback` object is made available to the transport provider by calling `TransportSender.getCredentialCallback()`.

`CredentialCallback.getUsernamePasswordCredential()` returns a `weblogic.security.UsernameAndPassword` instance. This is a simple class which has methods to get the username and password. The username/password returned depends on the type of service account. If the service account is of type static, the fixed username/password is returned. If it is mapped, the client subject is used to look up the remote username/password. If it is pass-through, the client's username/password is returned.

Note: A mapped service account throws `CredentialNotFoundException` if:

- if there is no map for the inbound client, or
 - the inbound security context is anonymous and there is no anonymous map
-
-

38.6.2.2 Outbound SSL Client Authentication (Two-Way SSL)

Oracle Service Bus also supports outbound SSL client authentication. In this case, the proxy making the outbound SSL request must be configured with a PKI key-pair for SSL. (This is done with a reference to a proxy service provider, the details are out of the scope of this document. To obtain the key-pair for SSL client authentication, the transport provider must call `CredentialCallback.getKeyPair()`. The HTTPS transport provider is an example of this.

38.6.2.3 Outbound JAAS Subject Authentication

Some transport providers send a serialized JAAS Subject on the wire as an authentication token. To obtain the inbound subject the transport provider must call `CredentialCallback.getSubject()`.

Note: The return value may be the anonymous subject.

38.6.3 Link-Level or Connection-Level Credentials

Some transports require credentials to connect to services. For example, FTP endpoints may be required to authenticate to the FTP server. Transport providers can make use of static service accounts to retrieve a username/password for establishing the connection. Note that mapped or pass-through service accounts cannot be used in this case because these connections are not made on behalf of a particular client request. If a transport provider decides to follow this approach, the endpoint must be configured with a reference to a service account. At runtime, the provider must call

`TransportManagerHelper.getUsernamePassword()`, passing the reference to the static service account.

38.6.4 Uniform Access Control to Proxy Services

Oracle Service Bus enforces access control to proxy services for every inbound request. Transport providers are not required to enforce access control or to provide interfaces to manage the access control policy.

Note: The access control policy covers the majority of the use cases; however, a transport provider can implement its own access control mechanisms (in addition to the access control check done by Oracle Service Bus) if required for transport provider specific reasons. If that is the case, please contact your Oracle representative. In general Oracle recommends transport providers let Oracle Service Bus handle access control.

When access is denied, `TransportManager.receiveMessage()` throws an `AccessNotAllowedException` wrapped inside a `TransportException`. Transport providers are responsible for checking the root-cause of the `TransportException`. A transport provider may do special error handling when the root cause is an `AccessNotAllowedException`. For example, the HTTP/S transport provider returns an HTTP 403 (forbidden) error code in this case.

Note: Oracle Service Bus makes the request headers available to the authorization providers for making access control decisions.

38.6.5 Identity Propagation and Credential Mapping

As explained in [Section 38.6.2, "Outbound Request Authentication,"](#) Oracle Service Bus provides three types of service accounts. A transport provider can make use of service accounts to get access to the username/password for outbound authentication. A service account hides all of the details of identity propagation and credential mapping from Oracle Service Bus transport providers.

38.7 The Threading Model

This section discusses the threading model used by Oracle Service Bus and how the model relates to the Transport SDK. This section includes these topics:

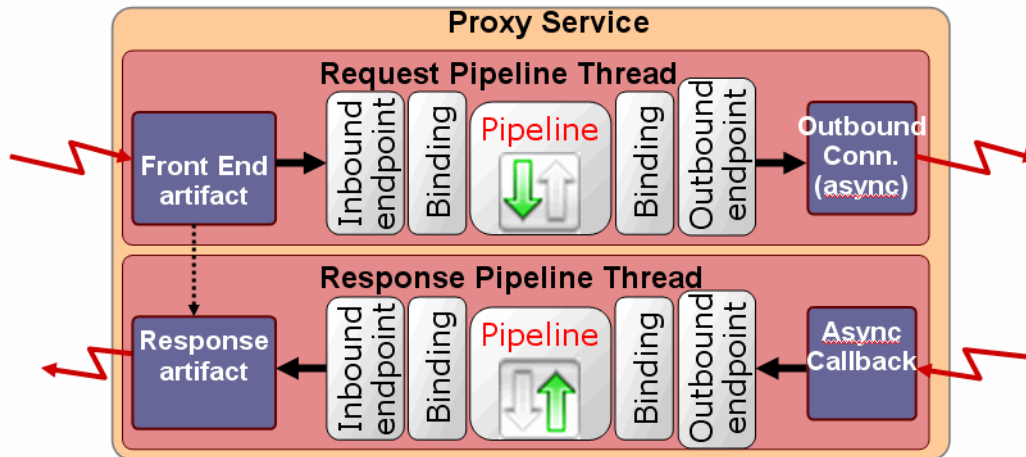
- [Section 38.7.1, "Overview"](#)
- [Section 38.7.2, "Inbound Request Message Thread"](#)
- [Section 38.7.3, "Outbound Response Message Thread"](#)
- [Section 38.7.4, "Support for Asynchrony"](#)
- [Section 38.7.5, "Publish and Service Callout Threading"](#)

38.7.1 Overview

[Figure 38–5](#) illustrates the Oracle Service Bus threading model for a hypothetical transport endpoint processing a single inbound message.

A front end artifact, such as a Servlet, is responsible for getting the inbound message. A request can be routed to an outbound endpoint and sent asynchronously. At this point, the thread is released. At some later point, a response is sent back to Oracle Service Bus (using a callback). The response is received, packaged, and handed to the Oracle Service Bus pipeline. Later, the pipeline notifies the inbound endpoint that the response is ready to be sent to the client. This processing is scalable because a thread is only tied up as long as it is needed.

Figure 38–5 Sample Oracle Service Bus Threading Model



38.7.2 Inbound Request Message Thread

The following actions occur in the same thread:

1. An inbound message is received by the front end artifact of the transport endpoint. This front end artifact could be, for example, an HTTP servlet or JMS message-driven bean instance.
2. The message is packaged into a `TransportMessageContext` object by the transport endpoint implementation and passed to the Oracle Service Bus runtime. For more information on the `TransportMessageContext` interface, see [Section 41.5, "Metadata and Header Representation for Request and Response Messages."](#)
3. The pipeline performs request pipeline actions configured for the proxy.
4. While processing the inbound message in Oracle Service Bus pipeline, in the same (request) thread, Oracle Service Bus runtime calls on the registered outbound transport endpoint, which may or may not be managed by the same provider, to deliver an outbound message to an external service.
5. At some later point, the external service asynchronously calls on the outbound endpoint to deliver the response message. The outbound endpoint must have been registered previously with a transport specific callback object.

Note: At this point, the initial request thread is released and placed back into the Oracle WebLogic Server thread pool for use by another request.

38.7.3 Outbound Response Message Thread

The following actions occur in the same thread:

1. The response message is packaged into a `TransportMessageContext` object and delivered back to Oracle Service Bus runtime for response processing. This processing occurs in a different thread than the request thread. This new thread is called the response thread.
2. After the response message is processed, Oracle Service Bus runtime calls on the `InboundTransportMessageContext` object to notify it that it is now time to send the response back to the original caller. For more information on the `InboundTransportMessageContext` interface, see [Section 41.5, "Metadata and Header Representation for Request and Response Messages."](#)

If the transport provider does not have a native implementation of an asynchronous (non-blocking) outbound call, it still needs to deliver the response back to Oracle Service Bus runtime on a separate thread than that on which the inbound request message was received. To do this, it can execute the call in a blocking fashion in the request thread and then use a Transport SDK helper method to deliver the response back to Oracle Service Bus runtime.

For example, the EJB transport provider does not have an asynchronous (non-blocking) outbound call. The underlying API is a blocking API. To work around this, the provider makes its blocking call, then schedules the response for processing with `TransportManagerHelper.schedule()`. For more information on the EJB transport provider, see [Chapter 28, "EJB Transport."](#)

38.7.4 Support for Asynchrony

By design, the transport subsystem interacts asynchronously with Oracle Service Bus. The reason for this is that asynchronous behavior is more scalable, and therefore, more desirable than synchronous behavior. Rather than create two separate APIs, one for asynchronous and one for synchronous interaction, Oracle Service Bus runtime expects asynchronous interaction. It is up to the transport developer to work around this by a method such as posting a blocking call and posting the response in a callback. In any case, the response must be executed in a different thread from the request.

38.7.5 Publish and Service Callout Threading

The threading diagram shown in [Figure 38–5](#) focuses on routing. The transport subsystem behaves the same way for Oracle Service Bus Publish and Service Callout actions which can occur in the middle of the request or response pipeline processing. These actions occur outside the scope of the transport subsystem and in the scope of an Oracle Service Bus pipeline. Therefore, some differences exist between the threading behavior of Publish and Service Callout actions and transport providers.

Note, however, the following cases:

- **Service Callout** – The pipeline processor will block the thread until the response arrives asynchronously. The blocked thread would then resume execution of the pipeline. The purpose is to bind variables that can later be used in pipeline actions to perform business logic. Therefore, these actions must block so that the business logic can be performed before the response comes back.
- **Publish** – The pipeline processor may or may not block the thread until the response arrives asynchronously. This thread then continues execution of the rest of the request or response pipeline processing.

Tip: A Service Callout action allows you to configure a synchronous (blocking) call to a proxy or business service that is already registered with Oracle Service Bus. Use a Publish action to identify a target service for a message and configure how the message is packaged and sent to that service. For more information on Service Callout and Publish actions, see [Section 2.4.29, "Adding and Configuring Service Callout Actions in Message Flows"](#) and [Section 2.4.17, "Adding and Configuring Publish Actions in Message Flows."](#)

38.8 Designing for Message Content

This section includes these topics:

- [Section 38.8.1, "Overview"](#)
- [Section 38.8.2, "Sources and Transformers"](#)
- [Section 38.8.3, "Sources and the MessageContext Object"](#)
- [Section 38.8.4, "Built-In Transformations"](#)

38.8.1 Overview

Transport providers have their own native representation of message content. For example, HTTP transport uses `java.io.InputStream`, JMS has Message objects of various types, Tuxedo has buffers, and the WLS Web Services stack uses SAAJ. However, within the runtime of a proxy service, the native representation of content is the Message Context. While Oracle Service Bus supports some common conversion scenarios, such as `InputStream` to/from Message Context, this conversion between transport representation and the Message Context is ultimately the transport provider's responsibility.

In general, the Transport SDK is not concerned with converting directly between two different transport representations of content. However, if two transports use compatible representations and the content does not require re-encoding, the SDK may allow the source content to be passed-through directly (for example, passing a `FileInputStream` from an inbound File transport to an outbound HTTP transport). However, if the source content requires any sort of processing, it makes more sense to unmarshal the source content into the Message Context first and then use the standard mechanisms to generate content for the outgoing transport.

38.8.2 Sources and Transformers

Content is represented as an instance of the Source interface. Transport SDK interfaces that deal with message content, such as `TransportSender` and `TransportMessageContext`, all use the Source interface when passing message payloads. The requirements on a Source are minimal. A Source must support push- and pull-based conversions to byte-based streams using the two methods defined in the base Source interface. A Source may or may not take into account various transformation options, such as character-set encoding, during serialization, as specified by the `TransformOptions` parameter.

While all Source objects must implement the base serialization interface, the underlying representation of the Source object's content is implementation specific. This allows for Source objects based on `InputStreams`, JMS Message objects, Strings, or whatever representation is most natural to a particular transport. Typically, Source implementations allow direct access to the underlying content, in addition to the base serialization methods. For example, `StringSource`, which internally uses a String object

to store its content offers a `getString()` method to get at the internal data. The ultimate consumer of a Source can then extract the underlying content by calling these source-specific APIs and potentially avoid any serialization overheads.

Sources may also be transformed into other types of Sources using a Transformer object. If a Source consumer, such as a transport provider, is given a Source instance that it does not recognize, it can often transform it into a Source instance that it does recognize. The underlying content can then be extracted from that known Source using the source-specific APIs. However, often a transport provider simply serializes the content and send it using the base serialization methods. See also [Section 41.4, "Source and Transformer Classes and Interfaces."](#)

38.8.3 Sources and the MessageContext Object

Sources are the common content representation between the transport layer and the binding layer. The binding layer is the entity responsible for converting content between the Source representation used by the transport layer and the MessageContext used by the pipeline runtime. How that conversion happens depends upon the type of service (its binding type) and the presence of attachments. While not strictly part of the Transport SDK, any transport provider that defines its own Source objects should be familiar with this conversion process.

When attachments are not present, the incoming Source represents just the core message content. The MessageContext is initialized by converting the received Source to a specific type of Source and then extracting the underlying content. For example, for XML-based services, the incoming Source is converted to an `XmlObjectSource`. The `XmlObject` is then extracted from the `XmlObjectSource` and used as the payload inside the `$body` context variable. SOAP services are similarly converted to `XmlObjectSource` except that the extracted `XmlObject` must be a SOAP Envelope so that the `<SOAP:Header>` and `<SOAP:Body>` elements can be extracted to initialize the `$header` and `$body` context variables.

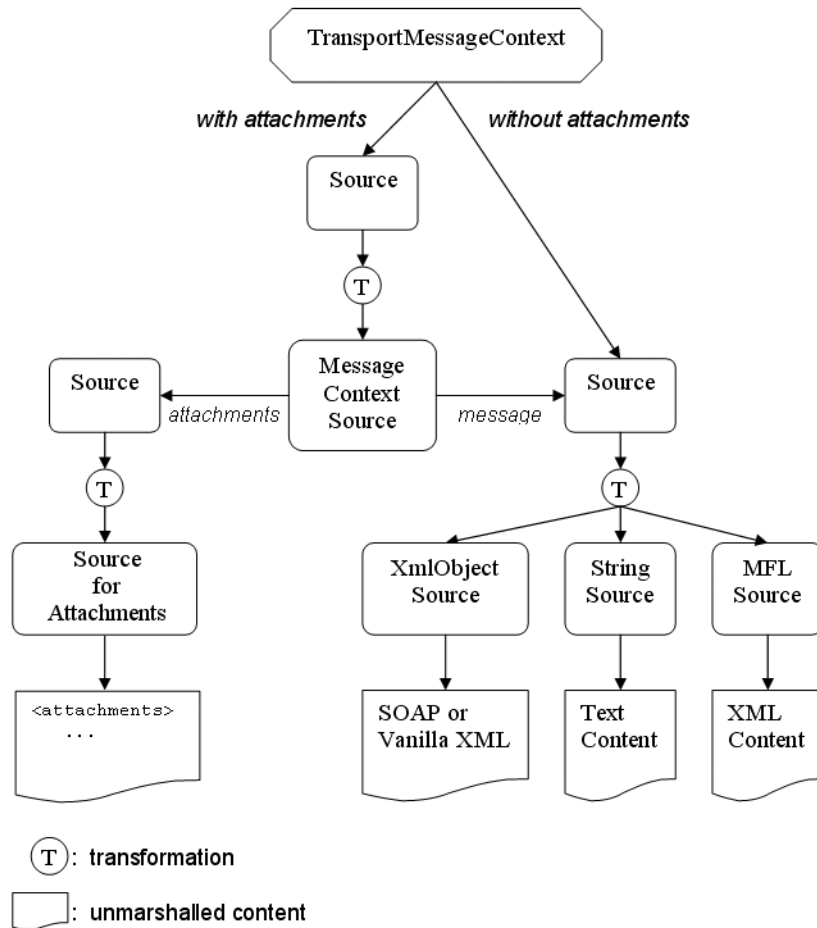
Below are the canonical Source types used for the set of defined service-types:

- **SOAP** – `XmlObjectSource`
- **XML** – `XmlObjectSource`
- **TEXT** – `StringSource`
- **MFL** – `MFLSource`

For binary services, no Source conversion is done. Instead, the Source is registered with a `SourceRepository` and the resulting `<binary-content />` XML is used as the payload inside `$body`.

When attachments are present, the incoming Source is first converted to a `MessageContextSource`. From the `MessageContextSource`, two untyped Source objects are obtained, one representing the attachments and one representing the core message. The Source for the core message is handled as described previously. The Source representing attachments is converted to an `AttachmentsSource`. From the `AttachmentsSource`, XML is obtained and is used to initialize the `$attachments` context variable and a `SourceRepository` containing the registered Sources that represent any binary attachment content. This entire process is illustrated in [Figure 38-6](#).

Figure 38–6 Flow of Attachments



A similar conversion occurs when creating a Source from data in the MessageContext to be passed to the transport layer. The core message is represented by a Source instance that can be converted to the canonical Source for the service type. In most cases, the Source will already be an instance of the canonical Source, but not always. When attachments are present, the Source delivered to the transport layer will be a source that can be converted to an instance of MessageContextSource. If the transport provider supports Content-Type as a pre-defined transport header, then the delivered Source will likely be an instance of MessageContextSource. Otherwise, the delivered Source will likely be an instance of MimeSource, but this can also be converted to a MessageContextSource.

The reason for this difference is that transports that natively support Content-Type as a transport header require that the top-level MIME headers appear in the transport headers rather than in the payload. Examples of this are HTTP and Email. Transports that do not natively support Content-Type must have these top-level MIME headers as part of the payload, as the Content-Type header is critical for decoding a multipart MIME package.

38.8.4 Built-In Transformations

Below is a matrix showing the set of supported transformations offered by the built-in transformers. The column of Source names on the left indicates the initial Source type and the row of Source names on the top indicates the target Source type. An "X" in a given row R and column C means that it is possible to directly convert from initial

Source R to target Source C. For example, there is some built-in transformer that handles converting a `StringSource` into an `XmlObjectSource`; however, there is no transformer that can convert a `StringSource` into an `MessageContextSource`. Typically, these transformers take advantage of their knowledge of the internal data representation used by both Source types.

Figure 38–7 Transformation Matrix

		Public Sources										
		Source	StreamSource	ByteArraySource	StringSource	XmlObjectSource	DOMSource	MFLSource	MimeSource	SAAJSource	MessageContextSource	
Public Sources	Source	X	X	X	X	X	X	X		X		
	StreamSource		X									
	ByteArraySource			X								
	StringSource				X	X	X					
	XmlObjectSource				X	X	X	X				
	DOMSource				X	X	X	X				
	MFLSource					X	X	X				
	MimeSource									X	X	X
	SAAJSource									X	X	X
	MessageContextSource									X	X	X

Of special interest is the very first row of "X" values in the matrix, as it represents supported transformations from arbitrary Sources into specific Sources. For example, while there is no transformer that specifically handles converting an `XmlObjectSource` to a `ByteArraySource`, there is a transformer that will handle converting any instance of `Source` to a `ByteArraySource`. These generic transformations are done without any knowledge of the initial Source type but instead rely on the base serialization methods that are implemented by all Sources: `getInputStream()` and `writeTo()`. So, although it is ultimately possible to convert an `XmlObjectSource` to a `ByteArraySource`, it is not done using any special knowledge of the internal details of `XmlObjectSource`.

Note: Many custom Sources implemented by Transports can be handled by these generic transformations, especially if the underlying data is an unstructured collection of bytes. For example, the File Transport uses a custom Source that pulls its content directly from a file on disk. However, as that data is just a set of bytes without structure, there is no need to provide custom transformations to, for example, `XmlObjectSource`. The generic transformation `Source XmlObjectSource` can handle this custom `FileSource` using just the base serialization methods that all Sources must implement.

For more information, see [Section 41.4, "Source and Transformer Classes and Interfaces."](#)

Developing a Transport Provider

The Transport SDK provides a layer of abstraction between transport protocols and the Oracle Service Bus runtime system. This layer of abstraction makes it possible to develop and plug in new transport providers to Oracle Service Bus. The Transport SDK interfaces provide this bridge between transport protocols, such as HTTP, and the Oracle Service Bus runtime.

Tip: Before beginning this chapter, be sure to review [Chapter 38, "Design Considerations,"](#) first.

This chapter explains the basic steps involved in developing a custom transport provider. This chapter includes these topics:

- [Section 39.1, "Development Road Map"](#)
- [Section 39.2, "Before You Begin"](#)
- [Section 39.3, "Basic Development Steps"](#)
- [Section 39.4, "Important Development Topics"](#)

39.1 Development Road Map

The process of designing and building a custom transport provider is complex. This section offers a recommended path to follow as you develop your transport provider. Development of a custom transport provider breaks down into these basic stages:

- [Section 39.1.1, "Planning"](#)
- [Section 39.1.2, "Developing"](#)
- [Section 39.1.3, "Packaging and Deploying"](#)

39.1.1 Planning

Review the following planning steps before developing a custom transport provider.

1. Decide if you really need to develop a custom transport provider. See [Section 38.3, "Do You Need to Develop a Custom Transport Provider?"](#)
2. Run and study the example socket transport provider. The source code for this provider is installed with Oracle Service Bus and is publicly available for you to examine and reuse. See [Chapter 42, "Sample Socket Transport Provider."](#)
3. Review [Chapter 38, "Design Considerations."](#) This chapter discusses the architecture of a transport provider and many aspects of transporter provider

design, such as the security model and the threading model employed by transport providers.

4. Review [Section 39.2, "Before You Begin."](#)

39.1.2 Developing

[Section 39.3, "Basic Development Steps"](#) outlines the steps you need to take to develop a transport provider, including schema configurations and interface implementations.

[Section 39.4, "Important Development Topics"](#) discusses in detail several topics that you might need to refer to during the development cycle. This section includes detailed information on topics such as [Section 39.5, "Handling Messages,"](#) [Section 39.6, "Transforming Messages,"](#) [Section 39.8, "Handling Errors,"](#) and others.

39.1.3 Packaging and Deploying

For detailed information on packaging and deploying a transport provider, see [Chapter 43, "Deploying a Transport Provider."](#)

39.2 Before You Begin

Before you begin to develop a custom transport provider, you need to consider and review a number of design issues, which include:

- Deciding if you really need to develop a custom transport provider. See [Section 38.3, "Do You Need to Develop a Custom Transport Provider?"](#)
- Deciding if your message endpoints are transactional or non-transactional. See [Section 38.5.1.1, "Transactional vs. Non-Transactional Endpoints."](#)
- Deciding if your message endpoints are one way, synchronous, or asynchronous. See [Section 38.5.1.2, "Supported Message Patterns,"](#) and [Section 38.5.2, "Support for Synchronous Transactions."](#)
- Deciding on the security requirements for outgoing and incoming messages. See [Section 38.6, "The Security Model."](#)
- Understanding the threading model used by Oracle Service Bus. See [Section 38.7, "The Threading Model."](#)
- Understanding whether your transport provider will support synchronous or asynchronous outbound calls. See [Section 38.7.4, "Support for Asynchrony."](#)
- Reviewing the interfaces and classes provided with the Transport SDK, and becoming familiar with how they fit into the design time and runtime parts of a transport provider. See [Chapter 41, "Transport SDK Interfaces and Classes."](#)
- Understanding how to package and deploy a custom transport provider. See [Chapter 43, "Deploying a Transport Provider."](#)
- Reviewing the flow of method calls through the transport framework. See [Appendix A, "Transport SDK UML Sequence Diagrams."](#)

39.3 Basic Development Steps

The basic steps to follow when developing a custom transport provider include:

[Section 39.3.1, "1. Review the Transport Framework Components"](#)

[Section 39.3.2, "2. Create a Directory Structure for Your Transport Project"](#)

[Section 39.3.3, "3. Create an XML Schema File for Transport-Specific Artifacts"](#)

[Section 39.3.4, "4. Define Transport-Specific Artifacts"](#)

[Section 39.3.5, "5. Define the XMLBean TransportProviderConfiguration"](#)

[Section 39.3.6, "6. Implement the Transport Provider User Interface"](#)

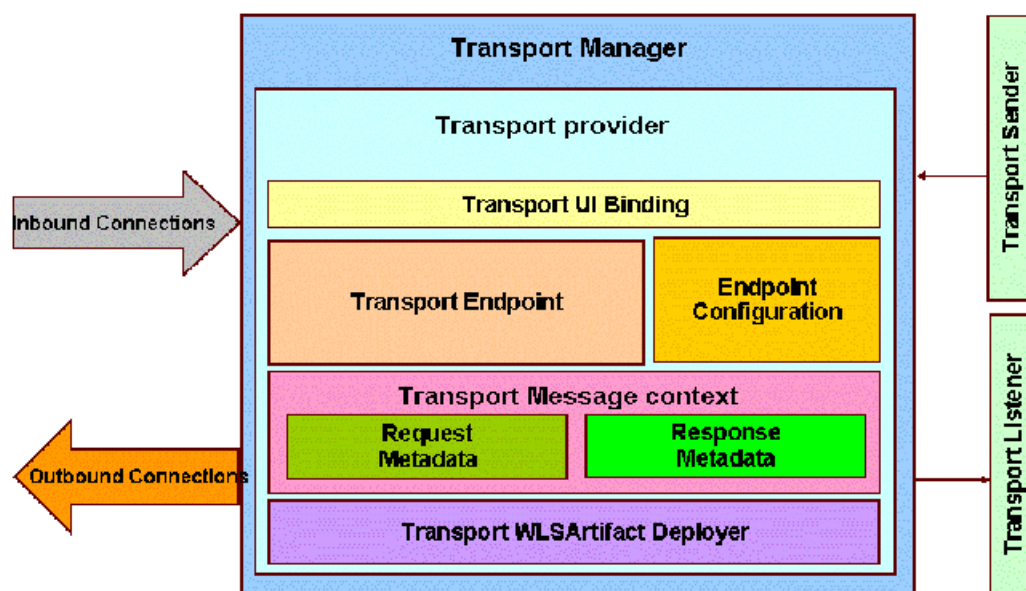
[Section 39.3.7, "7. Implement the Runtime Interfaces"](#)

[Section 39.3.8, "8. Deploy the Transport Provider"](#)

39.3.1 1. Review the Transport Framework Components

Figure 39–1 illustrates the components that you must implement and configure to create a custom transport provider. The transport manager controls and manages the registration of transport providers and handles communication with Oracle Service Bus. A transport provider manages the life cycle and runtime behavior of transport endpoints (resources where messages originate or are targeted). You use the Transport SDK to develop custom transport providers. Using the Transport SDK to develop a custom transport provider is the subject of this chapter.

Figure 39–1 *Transport Subsystem Overview*



The parts of the transport subsystem that you must implement and configure include:

- **Transport UI Bindings** – The user interface component for the transport provider. Related interfaces are summarized in [Section 41.6, "User Interface Configuration."](#)
- **Transport endpoint** – Responsible for sending and accepting messages. Related interfaces are summarized in [Section 41.3, "General Classes and Interfaces."](#)
- **Endpoint configuration** – Stores endpoint configurations. Related interfaces are listed in [Section 41.2, "Schema-Generated Interfaces."](#)
- **Transport message context** – Contains metadata for request and response headers and other parts of the message (inbound and outbound). See also [Section 41.4, "Source and Transformer Classes and Interfaces,"](#) and [Section 41.5, "Metadata and Header Representation for Request and Response Messages."](#)

- **WLS Artifact deployer** – (optional) Deploys artifacts, such as servlets that receive and send messages.
- **Transport sender** – Retrieves metadata for the outbound message and the payload. Related interfaces are summarized in [Section 41.3.2, "Summary of General Interfaces."](#)
- **Transport listener** – Allows the outbound endpoint to post the result of an outbound request to the rest of Oracle Service Bus. See also [Section 41.5, "Metadata and Header Representation for Request and Response Messages."](#)
- **Request/Response Metadata** – Related interfaces are summarized in [Section 41.5, "Metadata and Header Representation for Request and Response Messages."](#)

39.3.2 2. Create a Directory Structure for Your Transport Project

Before developing a new transport provider, take time to set up an appropriate directory structure for your project. The recommended approach is to copy the directory structure used for the sample socket transport provider. For a detailed description of this structure, see [Section 42.2, "Sample Location and Directory Structure."](#)

39.3.3 3. Create an XML Schema File for Transport-Specific Artifacts

Create an XML schema (xsd) file for transport-specific definitions. You can base this file on the schema file developed for the sample socket transport provider: `OSB_ORACLE_HOME/samples/servicebus/sample-transport/schemas/SocketTransport.xsd`

Note: The `SocketTransport.xsd` file imports the file `TransportCommon.xsd`. This file is the base schema definition file for service endpoint configurations. This file is located in `OSB_ORACLE_HOME/lib/sb-schemas.jar`. You might want to review the contents of this file before continuing.

39.3.4 4. Define Transport-Specific Artifacts

Define XML schema for the following transport-specific artifacts in the XML schema file described in the previous section, [Section 39.3.3, "3. Create an XML Schema File for Transport-Specific Artifacts."](#)

- `EndpointConfiguration`
- `RequestMetaDataXML`
- `ResponseMetaDataXML`

Note: Only simple XML types are supported when defining transport provider-specific metadata and headers. For example, complex types with nested elements are not supported. Furthermore, an additional restriction is that there can be at most one header with a given name.

Tip: Each of these schema definitions is converted into a corresponding Java file and compiled. You will find these converted Java source files for the sample socket transport provider in the directory:
 sample-transport/build/classes/com/bea/alsb/transports/sock/im
 pl

39.3.4.1 EndPointConfiguration

EndPointConfiguration is the base type for endpoint configuration, and describes the complete set of parameters necessary for the deployment and operation of an inbound or outbound endpoint. This configuration consists of generic and provider-specific parts. For more information on the EndPointConfiguration schema definition, refer to the documentation elements in the TransportCommon.xsd file.

You need to specify a provider-specific endpoint configuration in the schema file.

[Example 39-1](#) shows an excerpt from the SocketTransport.xsd.

Example 39-1 Sample SocketEndPointConfiguration Definition

```
<xs:complexType name="SocketEndpointConfiguration">
  <xs:annotation>
    <xs:documentation>
      SocketTransport - specific configuration
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:choice>
      <xs:element name="outbound-properties"
        type="SocketOutboundPropertiesType"/>
      <xs:element name="inbound-properties"
        type="SocketInboundPropertiesType"/>
    </xs:choice>
    <xs:element name="request-response" type="xs:boolean">
      <xs:annotation>
        <xs:documentation>
          Whether the message pattern is synchronous
          request-response or one-way.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  ...
</xs:complexType>
```

39.3.4.2 RequestMetaDataXML

It is required that each transport provider store metadata (message headers) in a Plain Old Java Object (POJO) and pass that to the pipeline. Examples of information that might be transmitted in the metadata are the Content-Type header, security information, or locale information. A RequestMetaData POJO is a generic object that extends the RequestMetaData abstract class and describes the message metadata of the incoming or outgoing request. The transport provider has to deliver the message metadata to Oracle Service Bus runtime in a RequestMetaData POJO. See also [Section 39.5.3, "Request and Response Metadata Handling."](#)

RequestMetaDataXML is an XML representation of the same RequestMetaData POJO. This XML representation uses Apache XML Bean technology. It is only needed by Oracle Service Bus runtime if or when processing of the message involves any actions in the pipeline that need an XML representation of the metadata, such as setting the entire metadata to a specified XML fragment on the outbound request.

You need to specify request metadata configuration in the schema file. [Example 39–2](#) shows an excerpt from the `SocketTransport.xsd`.

Example 39–2 Sample SocketRequestMetaDataXML Definition

```
<xs:complexType name="SocketRequestMetaDataXML">
  <xs:annotation>
    <xs:documentation/>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="ts:RequestMetaDataXML">
      <xs:sequence>
        <xs:element name="client-host"
          type="xs:string" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              Client host name
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="client-port" type="xs:int" minOccurs="0">
          <xs:annotation>
            <xs:documentation>Client port</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

39.3.4.3 RequestHeadersXML

RequestHeadersXML is the base type for a set of inbound or outbound request headers. You need to specify the RequestHeadersXML configuration in the schema file. [Example 39–2](#) shows an excerpt from the `SocketTransport.xsd`.

Example 39–3 Sample SocketRequestHeadersXML Definition

```
<xs:complexType name="SocketRequestHeadersXML">
  <xs:annotation>
    <xs:documentation/>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="ts:RequestHeadersXML">
      <xs:sequence>
        <xs:element name="message-count" type="xs:long" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              Number of messages passed till now.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```


39.3.4.4 ResponseMetaDataXML

ResponseMetaDataXML is the base type for metadata for a response to an inbound or outbound message. You need to specify the ResponseMetaDataXML configuration in the schema file. [Example 39–2](#) shows an excerpt from the `SocketTransport.xsd`.

Example 39–4 Sample SocketResponseMetaDataXML Definition

```
<xs:complexType name="SocketResponseMetaDataXML">
  <xs:complexContent>
    <xs:extension base="ts:ResponseMetaDataXML">
      <xs:sequence>
        <xs:element name="service-endpoint-host"
          type="xs:string" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              Host name of the service end point connection.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element name="service-endpoint-ip"
          type="xs:string" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              IP address of the service end point connection.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

39.3.4.5 ResponseHeadersXML

ResponseHeadersXML is the base type for a set of response headers. You need to specify the ResponseHeadersXML configuration in the schema file. [Example 39–2](#) shows an excerpt from the `SocketTransport.xsd`.

Example 39–5 Sample SocketResponseHeadersXML Definition

```
<xs:complexType name="SocketResponseHeadersXML">
  <xs:annotation>
    <xs:documentation/>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="ts:ResponseHeadersXML"/>
  </xs:complexContent>
</xs:complexType>
```

39.3.5 5. Define the XMLBean TransportProviderConfiguration

To configure the TransportProviderConfiguration XML bean, edit the transport provider configuration file. This XML file is located in the config directory in the transport provider root directory. See [Section 42.2, "Sample Location and Directory Structure,"](#) for the location of this file (`SocketConfig.xml`) in the sample socket transport provider implementation.

- If proxy services can use your transport, set the `inbound-direction-supported` element to `true`.

- If business services use your transport, set the `outbound-direction-supported` element to `true`.
- If your transport is self-described, include an element `self-described` with the value set to `true`. A self-described transport is one whose services are responsible for describing their shape (schema or WSDL) based on their endpoint configuration.
- If you want to publish a tModel for your transport to UDDI, include an element `UDDI`. See [Section 39.10, "Publishing Proxy Services to a UDDI Registry"](#) for more info.

Tip: The schema for `TransportProviderConfiguration` is defined in `TransportCommon.xsd`, which is located in `OSB_ORACLE_HOME/lib/sb-schemas.jar`. Refer to the schema file for more information.

39.3.6 6. Implement the Transport Provider User Interface

When you add a business or proxy service using the Oracle Service Bus Console, you select a transport provider in the service creation wizard. The wizard includes the transport providers that are provided with Oracle Service Bus and any custom transport providers that were developed with the Transport SDK.

This section discusses the Transport SDK API components that bind your custom transport provider to the Oracle Service Bus Console user interface. You must implement these APIs to connect your provider to the user interface.

Tip: This section assumes that you are familiar with the service creation wizard. See [Section 42.5, "Configuring the Socket Transport Sample,"](#) for a detailed, illustrated example.

Creating the Custom Transport User Interface

1. After users create a new service and choose the Service Type in the service creation wizard, they must then select an appropriate transport provider for the Service Type. To validate the selection, the wizard calls the following method of the `TransportUIBinding` interface:

```
public boolean isServiceTypeSupported(BindingTypeInfo binding)
```

This method determines if the transport provider is suitable for the selected Service Type.

2. After users select a valid transport provider, they enter an endpoint URI. To validate this URI, the wizard calls the following method of the `TransportUIBinding` interface:

```
public TransportUIError[] validateMainForm(TransportEditField[] fields)
```

3. Next, the wizard displays the transport-specific configuration page. To render this page, the wizard calls the following method of the `TransportUIBinding` interface:

```
public TransportEditField[] getEditPage(EndPointConfiguration config,
BindingTypeInfo binding) throws TransportException
```

The Transport SDK offers a set of `TransportUIObjects` that represent fields in the configuration page. For example, you can add text boxes, check boxes, and other types of UI elements. Use the `TransportUIFactory` to create them. After creation

use the same factory to specify additional properties and obtain `TransportEditField` objects that can be displayed by the service creation wizard.

Tip: You can associate events with most of the UI fields. An event acts like a callback mechanism for the `TransportUIBinding` class and lets you refresh, validate, and update the configuration page. When an event is triggered, the wizard calls the method:

```
updateEditPage(TransportEditField[] fields, String name) throws
TransportException
```

4. When users finishes the transport configuration, the wizard calls the validation method:

```
TransportUIError[] validateProviderSpecificForm(TransportEditField[] fields)
```

5. After the service is saved, the transport manager calls the following method of the `TransportProvider` class:

```
void validateEndPointConfiguration(TransportValidationContext context)
```

If no error is reported, a new endpoint is created. The Transport Manager then calls the method:

```
TransportEndPoint createEndPoint(EndPointOperations.Create context) throws
TransportException
```

If this method returns successfully, the new service is listed in the Oracle Service Bus Console and the underlying transport configuration is associated with an endpoint on the `TransportProvider`.

Note: The endpoint configuration is saved in the Oracle Service Bus session and does not need to be persisted or recovered in case of a server restart by the transport provider.

6. Once the session is activated, you must deploy the endpoint to start processing requests. See [Section 39.11, "When to Implement TransportWLSArtifactDeployer"](#) and [Section 43.4, "Deploying to a Cluster,"](#) to learn more about deploying an endpoint and processing requests.

Tip: For the sample socket transport provider, you can find the implementations of these interfaces in the `sample-transport/src` directory.

39.3.7 7. Implement the Runtime Interfaces

A new custom transport provider must implement the following runtime interfaces. For a summary of the Transport SDK interfaces and related classes, see [Chapter 41, "Transport SDK Interfaces and Classes."](#) For detailed information on interfaces and classes, see the *Oracle Fusion Middleware Java API Reference for Oracle Service Bus*.

Tip: For the sample socket transport provider, you can find the implementations of these interfaces in the `sample-transport/src` directory.

Implement the Following Runtime Interfaces

- `TransportProvider`

- `TransportWLSArtifactDeployer`

Note: Only implement the `TransportWLSArtifactDeployer` interface if the transport provider needs to deploy Oracle WebLogic Server-related artifacts, such as EAR/WAR/JAR files, that go into an Oracle WebLogic Server change list at the time of endpoint creation. For more information, see [Section 39.11, "When to Implement `TransportWLSArtifactDeployer`."](#)

- `TransportEndPoint`
- `InboundTransportMessageContext`
- `OutboundTransportMessageContext`
- `Transformer`

Note: Only implement the `Transformer` interface if the transport provider needs to work with non-standard payload bindings, for example, anything other than Stream, DOM, SAX, or XMLBean. For more information, see [Section 39.6, "Transforming Messages."](#)

39.3.8 8. Deploy the Transport Provider

For detailed information on deployment, see [Chapter 43, "Deploying a Transport Provider."](#)

39.4 Important Development Topics

This section discusses several topics that you will encounter while developing a custom transport provider. These topics include:

- [Section 39.5, "Handling Messages"](#)
- [Section 39.6, "Transforming Messages"](#)
- [Section 39.7, "Working with `TransportOptions`"](#)
- [Section 39.8, "Handling Errors"](#)
- [Section 39.9, "Defining Custom Environment Value Types"](#)
- [Section 39.10, "Publishing Proxy Services to a UDDI Registry"](#)
- [Section 39.11, "When to Implement `TransportWLSArtifactDeployer`"](#)
- [Section 39.12, "Creating Help for Custom Transports"](#)

39.5 Handling Messages

This section discusses message handling in transport providers and includes these topics:

- [Section 39.5.1, "Overview"](#)
- [Section 39.5.2, "Sending and Receiving Message Data"](#)
- [Section 39.5.3, "Request and Response Metadata Handling"](#)
- [Section 39.5.4, "Character Set Encoding"](#)

- [Section 39.5.5, "Co-Located Calls"](#)
- [Section 39.5.6, "Returning Outbound Responses to Oracle Service Bus Runtime"](#)

39.5.1 Overview

The Transport SDK features a flexible representation of message payloads. All Transport SDK APIs dealing with payload use the Source interface to represent message content.

The Source-derived message types provided with the Transport SDK include:

- StreamSource
- ByteArraySource
- StringSource
- XmlObjectSource
- DOMSource
- MFLSource
- SAAJSource
- MimeSource

Note: StreamSource is a single use source; that is, it implements the marker interface SingleUseSource. With the other Sources, you can get the input stream from the source multiple times. Each time the Source object gets the input stream from the beginning. With a SingleUseSource, you can only get the input stream once. Once the input is consumed, it is gone (for example, a stream from a network socket); however, Oracle Service Bus buffers the input from a SingleUseSource, essentially keeping a copy of all of its data.

If you implement a Source class for your transport provider, you need to determine whether you can re-get the input stream from the beginning. If the nature of the input stream is that it can only be consumed once, it is recommended that your Source class implement the marker interface SingleUseStream.

The Transport SDK provides a set of Transformers to convert between Source objects. You can implement new transformations, as needed, as long as they support transformations to/from a set of canonical representations. See [Section 39.6, "Transforming Messages"](#) for more information. See also [Section 38.8, "Designing for Message Content."](#)

39.5.2 Sending and Receiving Message Data

When implementing inbound endpoints to deliver the inbound message to Oracle Service Bus runtime, you need to call `TransportManager.receiveMessage()`. The transport provider is free to expose the incoming message payload in either one of the standard Source-derived objects, such as stream, DOM or SAX, or a custom one.

If Oracle Service Bus needs to send a response message back to the client that sent the request, it will call methods `setResponseMetaData()` and `setResponsePayload()` followed by `close()` on `InboundTransportMessageContext` to indicate that the response is ready to be sent

back. When Oracle Service Bus runtime calls the inbound transport message context `close()` method, this will be done from a different thread than that on which the inbound request message was received. The transport provider should be aware of this as it may affect the semantics of transactions. Also, the transport provider cannot attempt to access the response payload and/or metadata until `close()` method has been called.

39.5.3 Request and Response Metadata Handling

It is required that each transport provider store metadata and headers in a Plain Old Java Object (POJO) and pass that to the pipeline. There are some cases where Oracle Service Bus requires an XMLBean. In these cases, you need to implement a conversion from POJO to XMLBean using the API.

The following are the methods you must provide to convert from a POJO to XML:

```
RequestHeaders.toXML()
```

```
RequestMetaData<T>.toXML()
```

```
ResponseHeaders.toXML()
```

```
ResponseMetaData<T>.toXML()
```

For the reverse direction (XML to POJO) you need to implement:

```
TransportEndPoint.createRequestMetaData(RequestMetaDataXML)
```

```
InboundTransportMessageContext.createResponseMetaData(ResponseMetaDataXML)
```

39.5.4 Character Set Encoding

Each transport provider is responsible for specifying the character set encoding of the incoming message payload to Oracle Service Bus. For outgoing messages (outbound request and inbound response), the transport provider is responsible for telling Oracle Service Bus what character set encoding to use for the outgoing payload. The character-set encoding is specified in request and response metadata.

In virtually every case, the character-set encoding that the transport is responsible for inserting into the metadata is exactly the encoding that is statically specified in the service configuration. One of the few exceptions to this is HTTP transport, which inspects Content-Type for any "charset" parameters and overrides any encoding configured in the service. This is necessary in order to conform to HTTP specifications. Other transport protocols may need to handle similar issues.

Tip: In general, the encoding for a service is fixed. If someone sends an UTF-16 encoded message to a proxy that is specified to be SHIFT_JIS, then that is generally considered to be an error. Transport providers should not need to inspect the message simply to determine encoding.

For outgoing messages, the transport provider tells Oracle Service Bus what encoding it requires for the outbound request, and Oracle Service Bus performs the conversion if necessary.

Transports should always rely on this encoding for outgoing messages and should not assume that it is the same as the encoding specified in the service configuration. If there is a discrepancy, the transport can choose to allow it, but others could consider it an error and throw an exception. Also the transport has the additional option of

leaving the encoding element blank. That leaves the pipeline free to specify the encoding (for example, via pass-through).

39.5.5 Co-Located Calls

If a given transport provider supports proxy service endpoints, it is possible to configure the request pipeline such that there is a routing step that routes to that provider's proxy service. Furthermore there could be a Publish or a Service Callout action that sends a message to a proxy service instead of a business service. This use case is referred to as co-located calls.

The transport provider needs to be aware of co-located calls, and handle them accordingly. Depending on the nature of the proxy service endpoint implementation, the transport provider may choose to optimize the invocation such that this call bypasses the entire transport communication stack and any inbound authentication/authorization, and instead is a direct call that effectively calls `TransportManager.receiveMessage()` immediately.

Tip: Oracle Service Bus has implemented this optimization with the HTTP, File, Email and FTP transport providers. The JMS provider does not use this optimization due to the desire to separate the transactional semantics of send operation versus receive operations.

If you want to use this optimization in a custom transport provider, you need to extend the `CoLocatedTransportMessageContext` class and call its `send()` method when `TransportProvider.sendMessageAsync()` is invoked.

39.5.6 Returning Outbound Responses to Oracle Service Bus Runtime

When Oracle Service Bus runtime sends a message to an outbound endpoint and there is a response message to be returned, the transport provider must return this response asynchronously. That means `TransportSendListener.onReceiveResponse()` or `TransportSendListener.onError()` methods need to be called from a different thread than the one on which `TransportProvider.sendMessageAsync()` was called.

If the transport provider has a built-in mechanism by which the response arrives asynchronously, such as responses to JMS requests or HTTP requests when the async response option is used, it happens naturally. However, if the transport provider has no built-in mechanism for retrieving responses asynchronously, it can execute the outbound request in a blocking fashion and then schedule a new worker thread using the `TransportManagerHelper.schedule()` method, in which the response is posted to the `TransportSendListener`.

39.6 Transforming Messages

When Oracle Service Bus needs to set either the request payload to an outbound message or the response payload to an inbound message, it asks the transport provider to do so through an object derived from the `Source` interface. The transport provider then needs to decide what representation the underlying transport layer requires and use the `Transformer.transform()` method to translate the `Source` object into the desired source.

Tip: For more information on message transformation, see [Section 38.8, "Designing for Message Content."](#) For a list of built-in transformations, see [Section 38.8.4, "Built-In Transformations,"](#) and [Section 41.4, "Source and Transformer Classes and Interfaces."](#)

A custom transport provider can support new kinds of transformations. Suppose a transport provider needs to work with a DOM object in order to send the outbound message. When called with `setRequestPayload(Source src)`, the transport provider needs to call the method:

```
TransportManagerHelper.getTransportManager().getTransformer().
transform(src, DOMSource.class, transformOptions)
```

The return value of the method gives a `DOMSource`, which can then be used to retrieve the DOM node.

Note: If the transport provider requires a stream, there is a shortcut: each `Source` object supports transformation to stream natively.

You can add new transformations to a custom transport provider. For example, suppose you want to add a new kind of `Source`-derived class, called `XYZSource`. For performance reasons, transport providers are encouraged to provide conversions from `XYZSource` to one of the two canonical `Source` objects, `XmlObjectSource` and `StreamSource` when applicable. Without such transformation, generic transformers will be used, which rely on the `StreamSource` representation of `XYZSource`. Of course, if `XYZSource` is a simple byte-based `Source` with no internal structure, then relying on the generic transformers is usually sufficient. Note that any custom transformer that is registered with `TransportManager` is assumed to be thread-safe and stateless.

To support attachments, the transport provider has three options:

- The `Source` returned by `TransportMessageContext` must be an instance of `MessageContextSource`. A limitation of this option is that `MessageContextSource` requires that the content has already been partitioned into a core-message `Source` and an attachments `Source`.
- The `Source` is an instance of `MimeSource` and the `Headers` objects contain a multipart `Content-Type` header.
- The `Content-Type` is a pre-defined header for the transport provider with the specific value `multipart/related`. Both HTTP(S) and Email transports rely on this third option for supporting attachments.

39.7 Working with TransportOptions

A `TransportOptions` object is used to supply options for sending or receiving a message. A `TransportOptions` object is passed from the transport provider to the transport manager on inbound messages. On outbound messages, a `TransportOptions` object is passed from the Oracle Service Bus runtime to the transport manager, and finally to the transport provider.

This section includes these topics:

- [Section 39.7.1, "Inbound Processing"](#)
- [Section 39.7.2, "Outbound Processing"](#)
- [Section 39.7.3, "Request Mode"](#)

39.7.1 Inbound Processing

The transport provider supplies these parameters to `TransportManager.receiveMessage()`:

- **QOS** – Specifies exactly-once or best-effort quality of service. Exactly-once quality of service is specified when the incoming message is transactional.
- **Throw On Error** – If this flag is set, an exception is thrown to the callee of method `TransportManager.receiveMessage()` when an error occurs during the Oracle Service Bus pipeline processing. The options for throwing the exception include: throw the exception back to the inbound message or create a response message from the error and notify the inbound message with the response message. Typically, you set **Throw On Error** to true when QOS is exactly-once (for transactional messages).

For example, JMS/XA sets this flag to true to throw the exception in the same request thread, so it can mark the exception for rollback. HTTP sets the flag to false, because there is no retry mechanism. The error is converted to a status code and a response message is returned.

- **Any transport-specific opaque data** – Opaque data can be any data that is set by the transport provider and passed through the pipeline to the outbound call. This technique provides optimized performance when the same transport is used on inbound and outbound. The opaque data is passed directly through the pipeline from the inbound transport to the outbound transport. For example, the HTTP/S transport provider can pass the username and password directly from the inbound to the outbound to efficiently support identity pass-through propagation.

39.7.2 Outbound Processing

For outbound processing, the Oracle Service Bus runtime supplies these parameters to the transport manager, which uses some of the parameters internally and propagates some parameters to `TransportProvider.sendMessageAsync()`. These parameters include:

- **QOS** – Specifies whether or not "exactly-once" quality of service can be achieved. For example, for HTTP, if quality of service is set to exactly once, the HTTP call is blocking. If it is set to best effort, it is a non-blocking HTTP call.
- **Mode** – Specifies one-way or request response. See also [Section 38.2.3, "Transport Provider Modes."](#)
- **URI, Retry Interval, and Count** – The transport provider uses the URI to initialize the outbound transport connection. For example, the HTTP transport provider uses the URI when instantiating a new `URLConnection`. The transport provider is not required to use Retry Interval and Count.
- **OperationName** – The transport provider can use `OperationName` if it needs to know what outbound Web Service is being used. The transport manager uses this parameter to keep track of monitoring statistics.
- **Any transport-specific opaque data** – An example of transport-specific opaque data is the value of the "Authorization" header for HTTP/S.

39.7.3 Request Mode

The request mode is defined as an enumeration with two values: `REQUEST_ONLY` (also called "one-way") and `REQUEST_RESPONSE`. These modes are interpreted as follows for requests and responses:

- On outbound requests, the pipeline indicates the mode through `TransportOptions` and the transport provider must honor the mode.
- On inbound requests, the pipeline knows the mode and closes the inbound request and does not send a response if it computes the mode `REQUEST_ONLY`.
- If a response is sent by the pipeline, then there is a response even if the response is empty.
- For transports that are inherently one-way, the transport must not specify response metadata.

39.8 Handling Errors

There are three different use cases to consider with respect to the effect run-time exceptions have on the transactional model. These cases include:

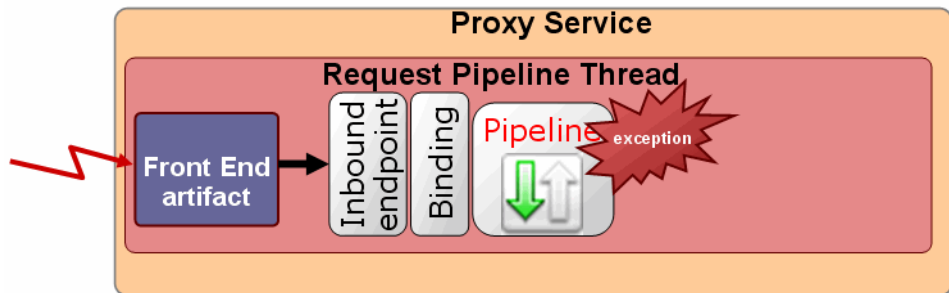
- [Section 39.8.1, "Case 1"](#): The exception occurs somewhere in the request pipeline but before the outbound call to the business service.
- [Section 39.8.2, "Case 2"](#): The exception occurs during the business service call.
- [Section 39.8.3, "Case 3"](#): The exception occurs sometime after the business service call in the response pipeline.
- [Section 39.8.4, "Catching Application Errors"](#)
- [Section 39.8.5, "Catching Connection Errors"](#)

39.8.1 Case 1

The exception occurs somewhere in the request pipeline but before the outbound call to the business service, as shown in [Figure 39–2](#). For example, executing a specific XQuery against the contents of the request message raises an exception.

If there is a user-configured error handler configured for the request pipeline, the error will be handled according to the user configuration. Otherwise, the proxy service will either catch an exception when calling `TransportManager.receiveMessage()` or will be notified in the `InboundTransportMessageContext.close()` method of the error through response metadata, based on the transport options passed as an argument to the `receiveMessage()` call. If the proxy service indicates that the exception should be thrown, surround `receiveMessage()` with a try/catch clause and mark the transaction for rollback.

Figure 39–2 Error Case 1



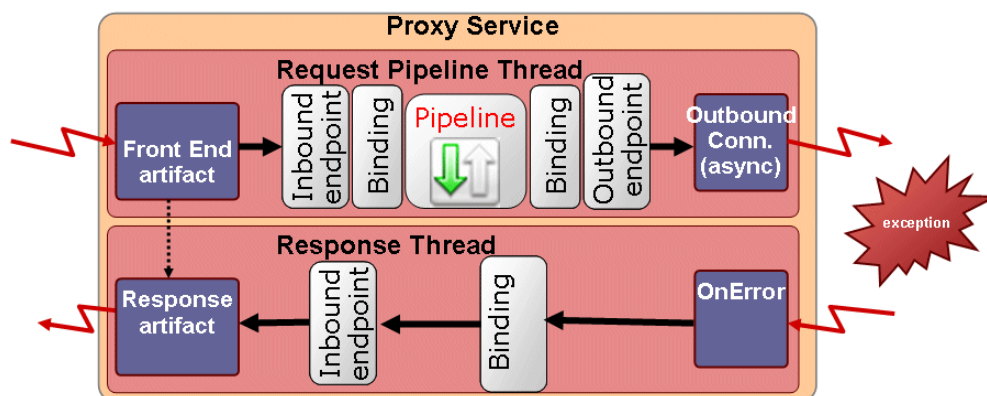
39.8.2 Case 2

The exception occurs during the business service call, as shown in [Figure 39-3](#). The outbound transport provider either:

- Throws an exception from `TransportProvider.sendMessageAsync()`. For example, the outbound provider throws an exception if there was an error while establishing a socket connection to external service. This situation could occur if the business service cannot be called because of an incorrect URL, a faulty connection, or other reasons. In these cases, the transport provider must raise an exception.
- Notifies the listener through `TransportSendListener.onError()`. For example, if the business service was called, but the call resulted in an error (such as a SOAP fault), the transport provider needs to call `TransportSendListener.onError()` instead of raising an exception.

In the first instance, the exception handling is the same as that described in [Section 39.8.1, "Case 1"](#). In the second instance, if there is an error handler configured for the response pipeline, the error is handled according to the user configuration. Otherwise, the exception is propagated back to the proxy service endpoint in `InboundTransportMessageContext.close()` through the response metadata.

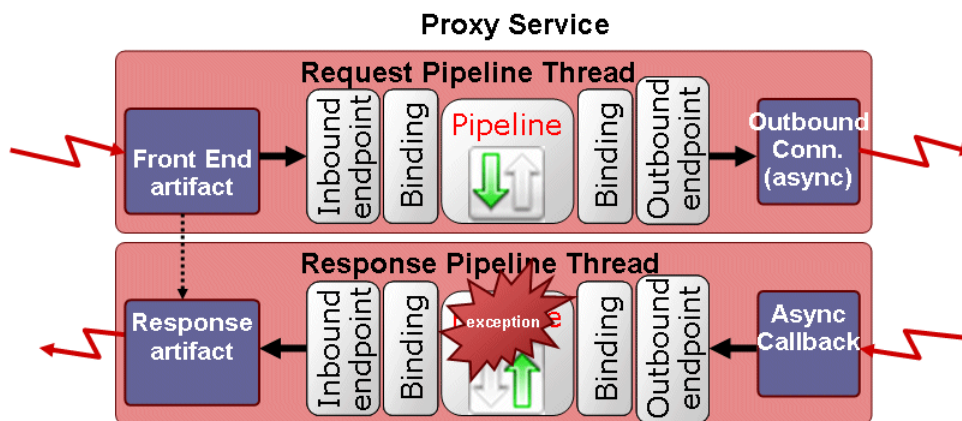
Figure 39-3 Error Case 2



39.8.3 Case 3

The exception occurs sometime after the business service call in the response pipeline, as shown in [Figure 39-4](#). Again, in the absence of a user-defined error handler for the response pipeline, the proxy service endpoint is notified of the error with the `InboundTransportMessageContext.close()` method with appropriate response metadata. If the inbound transport endpoint is a synchronous transactional endpoint, it is guaranteed that the transaction that was active at the time request was received is still active and it may be rolled back. If the inbound endpoint is not transactional or not synchronous, there is not an inbound transactional context to roll back, so some other action might need to be performed.

Figure 39–4 Error Case 3



39.8.4 Catching Application Errors

When business services try to access an external service and an error occurs in the external service application, such as a SOAP fault, subsequent retries by the services are likely to produce errors until the application is fixed.

Oracle Service Bus lets you identify application errors, giving you the option of preventing retries on application errors when your transport is used.

This section describes how to catch application errors in your transport and configure your transport to prevent application error retries.

39.8.4.1 Identifying Application Errors

In your transport provider code you must identify the conditions under which an application error occurs.

For example, in the Oracle Service Bus HTTP transport, an application error is one in which the HTTP response has a 500 status code, has a non-empty payload, and has a content type that is consistent with the service type, such as XML for SOAP.

When an error meets the application error conditions you define, return a `TRANSPORT_ERROR_APPLICATION` type using one of the following methods:

- Errors in the request – Throw a `TransportException` with the error code `TRANSPORT_ERROR_APPLICATION` in `TransportProvider.sendMessageAsync()`.
- Errors in the response – Schedule `TransportSendListener.onError()` with the error code `TRANSPORT_ERROR_APPLICATION`.

The transport SDK can then identify application errors when they occur and handle them accordingly.

The transport SDK also sends application errors to the pipeline `$fault` variable.

39.8.4.2 Configuring Application Error Retries

In your `<Transport>Config.xml` file, enter the following element as a child of the `<ProviderConfiguration>` element, according to the `TransportCommon.xsd` schema in `/osb_10.3/lib/sb-schemas.jar`:

```
<declare-application-errors>true</declare-application-errors>
```

This entry provides a Retry Application Errors option on the business service's main transport configuration page when a user selects your transport. If you do not provide this element, the default value is false, application errors are not caught, and no option is provided to retry application errors.

39.8.5 Catching Connection Errors

Oracle Service Bus lets you identify connection errors in your transport, which triggers the transport SDK to take inaccessible endpoint URIs offline automatically. For example, in a cluster with Oracle Service Bus running on managed servers, a managed server that experiences a connection error on a service request can automatically mark the endpoint URI as offline.

You can re-enable endpoint URIs in the following ways:

- On configuring the business service, you can set a Retry Count and Retry Iteration Interval to determine the frequency and number of retries after connection errors. A successful connection to the service after a retry automatically reactivates the endpoint URI.

A Retry Iteration Interval of zero (0) takes the endpoint URI offline indefinitely and requires you to manually re-enable the endpoint URI.

- You can manually re-enable offline endpoint URIs in the Oracle Service Bus console, on the Operational Settings page for the business service.

The automated connection error functionality does not apply to the following situations:

- If a service pipeline dynamically sets an endpoint URI in `$outbound/ctx:transport/ctx:uri`, the transport SDK cannot take the URI offline, because the endpoint URI is not defined in the service configuration.
- The transport SDK does not persist connection status. After a server restart, all endpoint URIs are considered online.

For more information, see "Managing Endpoint URIs for Business Services" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

39.8.5.1 Identifying Connection Errors

This section describes how to identify connection errors in your transport. Once caught, a connection error triggers the transport SDK to take the affected endpoint URI offline automatically.

In your transport provider code, you must identify the conditions under which a connection error occurs.

For example, in the Oracle Service Bus HTTP transport, a connection error is one in which the HTTP response has a 404 status code or there is an `IOException` when a connection is attempted on the endpoint URI.

When an exception meets the connection error conditions you define, return a `TRANSPORT_ERROR_CONNECTION` type using one of the following methods:

- Errors in the request – Throw a `TransportException` with the error code `TRANSPORT_ERROR_CONNECTION` in `TransportProvider.sendMessageAsync()`.
- Errors in the response – Schedule `TransportSendListener.onError()` with the error code `TRANSPORT_ERROR_CONNECTION`.

The transport SDK can then identify connection errors when they occur and handle them accordingly.

The transport SDK also sends connection errors to the pipeline \$fault variable and adds them to the response.

39.9 Defining Custom Environment Value Types

This section describes how to define custom environment value types that you want to use in your transport implementation. When you use the `TransportProvider.getEnvValues()` method to return environment values for an endpoint, you will be able to declare environment values of these custom types.

When your transport is used, custom environment value types can be used in the same ways that standard environment value types are used in Oracle Service Bus, such as for customization, find and replace, and preservation of values on configuration import. For example, you may want to be able to define and preserve references to a service account or a JMS queue in your transport configuration.

Environment value types can be any of the following categories: environment, operational, and security.

Add custom variables to your `<Transport>Config.xml` file. The schema that determines the XML structure is `TransportCommon.xsd`, located in `/osb_10.3/lib/sb-schemas.jar`.

Following is an example of a custom security variable in the JMS transport's `JmsConfig.xml`:

```
<env-value-type>
  <name>JMS Service Accounts</name>
  <localized-display-name>
    <localizer-class>com.bea.wli.sb.transports.messages.
      TransportsTextLocalizer</localizer-class>
    <message-id>JMS_SERVICE_ACCOUNTS</message-id>
  </localized-display-name>
  <localized-description>
    <localizer-class>com.bea.wli.sb.transports.messages.
      TransportsTextLocalizer</localizer-class>
    <message-id>JMS_SERVICE_ACCOUNTS</message-id>
  </localized-description>
  <simple-value>>true</simple-value>
  <category>security</category>
</env-value-type>
```

Following are descriptions of key elements for custom environment value types:

- `name` – The variable name used by the transport SDK.
- `display-name` – The name for the variable that appears in the Oracle Service Bus user interface. Following is the localization alternative:
 - `localized-display-name` – Alternative, localized version of `display-name`.
 - `localizer-class` – The localization properties text file containing the localized `display-name`. The `.properties` extension is not required.
 - `message-id` – The property in the localization properties file that provides the value of the display name.

- description – Description of the environment value type. For localization, use the localized-description element instead with the localizer-class and message-id child elements as described in display-name.
- simple-value – If the environment value type is of the category "environment," simple-value determines whether or not this type is searchable with find and replace functionality in Oracle Service Bus (value of true or false).
- category – The category of the environment value type: "environment," "security," or "operational." When the category is "security" or "operational," you can decide whether or not to preserve the environment value type during configuration import. When the category is "environment," the environment value type is available for find and replace.

39.10 Publishing Proxy Services to a UDDI Registry

Universal Description, Discovery, and Integration (UDDI) is a standard mechanism for describing and locating Web services across the internet. You might want to publish proxy services based on a custom transport provider to a UDDI registry. This allows proxy services to be imported into another Oracle Service Bus server in a different domain as the one hosting the business service.

To publish a proxy service, the transport provider needs to define a tModel that represents the transport type in the "UDDI" section of TransportProviderConfiguration XML schema definition. (For more information on the schema-generated interfaces, see [Section 41.2, "Schema-Generated Interfaces."](#))

This tModel must contain a CategoryBag with a keyedReference whose tModelKey is set to the UDDI Types Category System and keyValue is "transport." You are required to provide only the UDDI V3 tModel key for this tModel.

If UDDI already defines a tModel for this transport type, it is recommended that the definition be copied and pasted into the UDDI section.

An example of such a tModel is provided in [Example 39–6](#).

Example 39–6 Example tModel

```
<?xml version="1.0" encoding="UTF-8"?>
<ProviderConfiguration xmlns="http://www.bea.com/wli/sb/transports">
  . . .
  . . .
  <UDDI>
    <TModelDefinition>
      <tModel tModelKey="uddi:bea.uddi.org:transport:socket">
        <name>uddi-org:socket</name>
        <description>Socket transport based webservice</description>
        <overviewDoc>
          <overviewURL useType="text">
            http://www.bea.com/wli/sb/UDDIMapping#socket
          </overviewURL>
        </overviewDoc>
        <categoryBag>
          <keyedReference keyName="uddi-org:types:transport"
            keyValue="transport"
            tModelKey="uddi:uddi.org:categorization:types"/>
        </categoryBag>
      </tModel>
    </TModelDefinition>
  </UDDI>
```

```
</ProviderConfiguration>
```

If UDDI does not already define a tModel for this transport type, Oracle Service Bus can publish the tModel you define here to configured registries. When a UDDI registry is configured to Oracle Service Bus, the "Load tModels into Registry" option can be specified. That option causes all of the tModels of Oracle Service Bus, including the tModels for custom transport providers, to be published to the UDDI registry. After deploying your transport provider, you can update your UDDI registry configuration to publish your tModel.

During UDDI export,

`TransportProvider.getBusinessServicePropertiesForProxy(Ref)` is called and the resulting map is published to the UDDI registry. The provider is responsible for making sure to preserve all important properties of the business service in the map so that during the UDDI import process the business service definition can be correctly constructed without loss of information.

During UDDI import,

`TransportProvider.getProviderSpecificConfiguration(Map)` is called and the result is an `XmlObject` that conforms to the provider-specific endpoint configuration schema, which goes into the service definition.

Tip: OASIS, the Organization for the Advancement of Structured Information Standards, is responsible for creating the UDDI standard. To read more about UDDI, including the full technical specification, go to: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec

39.11 When to Implement TransportWLSArtifactDeployer

Two sets of transport provider interfaces are provided that deal with individual service registration. `TransportProvider` has methods like `create/update/delete/suspend/resume` and `TransportWLSArtifactDeployer` has the same methods. This section discusses these two implementations and explains when you need to implement `TransportWLSArtifactDeployer`.

Only implement `TransportWLSArtifactDeployer` if your provider needs to make changes to Oracle WebLogic Server artifacts in the Oracle Service Bus domain. The methods on `TransportWLSArtifactDeployer` are only called on an Administration Server. In this case, changes are made through the `DomainMBean` argument that is passed in, and then the changes are propagated to the entire cluster.

The `TransportProvider` methods are called on all servers (Administration and Managed Servers) in the domain. Because you cannot make changes to Oracle Service Bus domain artifacts on a managed server, the purpose of the method calls on `TransportProvider` is to update its internal data structures only.

When a given Transport provider implements the `TransportWLSArtifactDeployer` interface, the methods on `TransportWLSArtifactDeployer` are called before the corresponding methods on `TransportProvider`. For example, `TransportWLSArtifactDeployer.onCreate()` is called before `TransportProvider.createEndPoint()`.

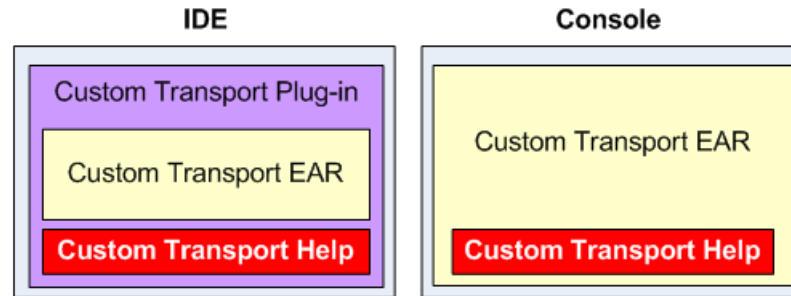
For more information on `TransportWLSArtifactDeployer`, see [Section 41.3.2, "Summary of General Interfaces."](#)

39.12 Creating Help for Custom Transports

You can provide online help for your custom transports in the design environment (Eclipse) and the run-time environment (Oracle Service Bus console). Providing help is optional, but it can be extremely useful in guiding service creators through the transport configuration process.

Figure 39–5 shows help included with a custom transport in the development and run-time environments.

Figure 39–5 Custom Transport Help in the Development and Run-Time Environments



This section includes the following topics:

- [Section 39.12.1, "Custom Transport Help Overview"](#)
- [Section 39.12.2, "Providing Custom Transport Help in Eclipse"](#)
- [Section 39.12.3, "Providing Custom Transport Help in the Oracle Service Bus Console"](#)

39.12.1 Custom Transport Help Overview

This section describes the different options available for providing custom transport help in Eclipse and the Oracle Service Bus console.

Note: Because of potential user interface and functionality differences between transport configuration in Eclipse and the Oracle Service Bus console, consider creating separate help topics for both environments.

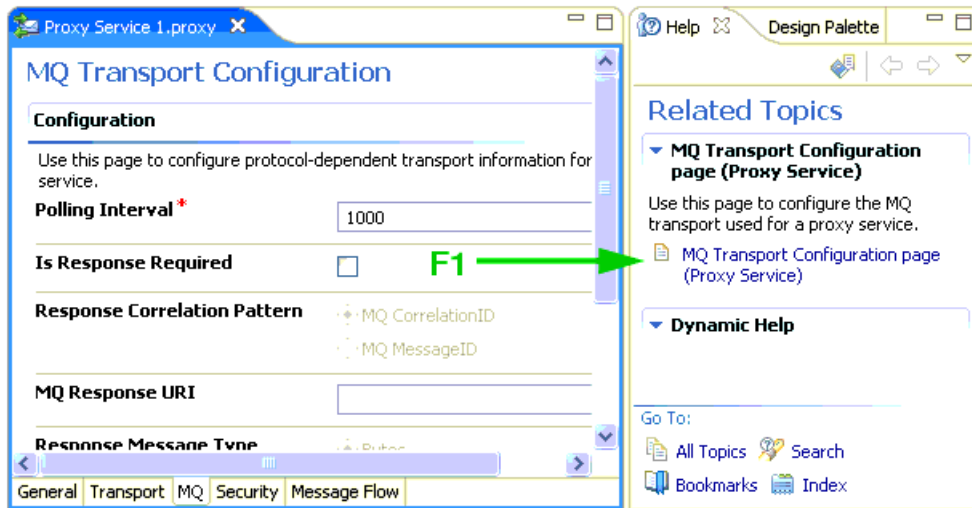
39.12.1.1 Eclipse Help

Eclipse help is based on the Eclipse help framework. You have choices for your custom transport help implementation in Eclipse.

- [Section 39.12.1.1.1, "Context-Sensitive Help \(F1\)"](#)
- [Section 39.12.1.1.2, "Eclipse Help Table of Contents"](#)

39.12.1.1.1 Context-Sensitive Help (F1) Context-sensitive help, launched by pressing F1 in Eclipse, shows help topics within Eclipse instead of launching a separate help window that displays the entire help system. Figure 39–6 shows how context-sensitive help appears in Eclipse.

Figure 39–6 Pressing F1 on a Transport Configuration Page to Display Help for the Transport



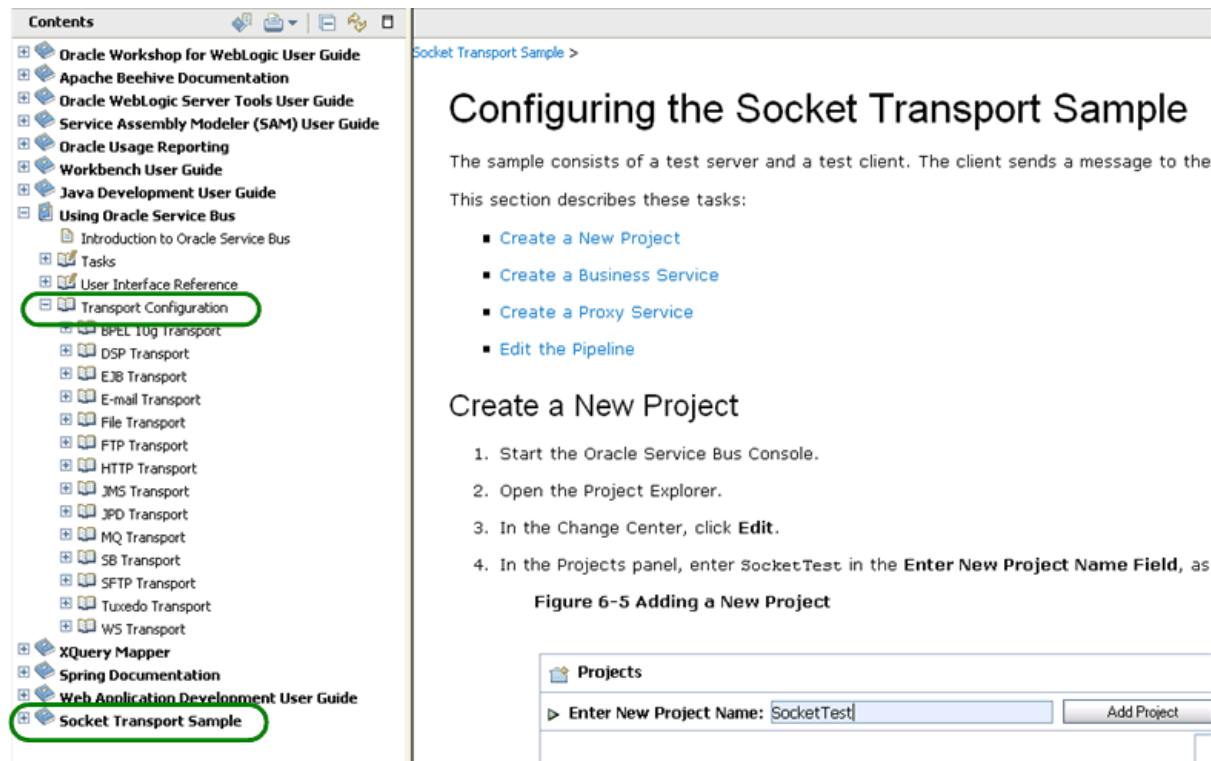
All of the native Oracle Service Bus transports provide context-sensitive help from their respective transport configuration editors.

The benefit of context-sensitive help is quick user access to targeted help topics without leaving Eclipse, which is particularly useful for help with custom transports.

39.12.1.1.2 Eclipse Help Table of Contents You can provide help content for your custom transport in the Eclipse help system, accessible from the Eclipse Help menu. When you launch Eclipse help, a separate window displays the contents of the entire help system.

Figure 39–7 shows online help for transports in the Eclipse help system.

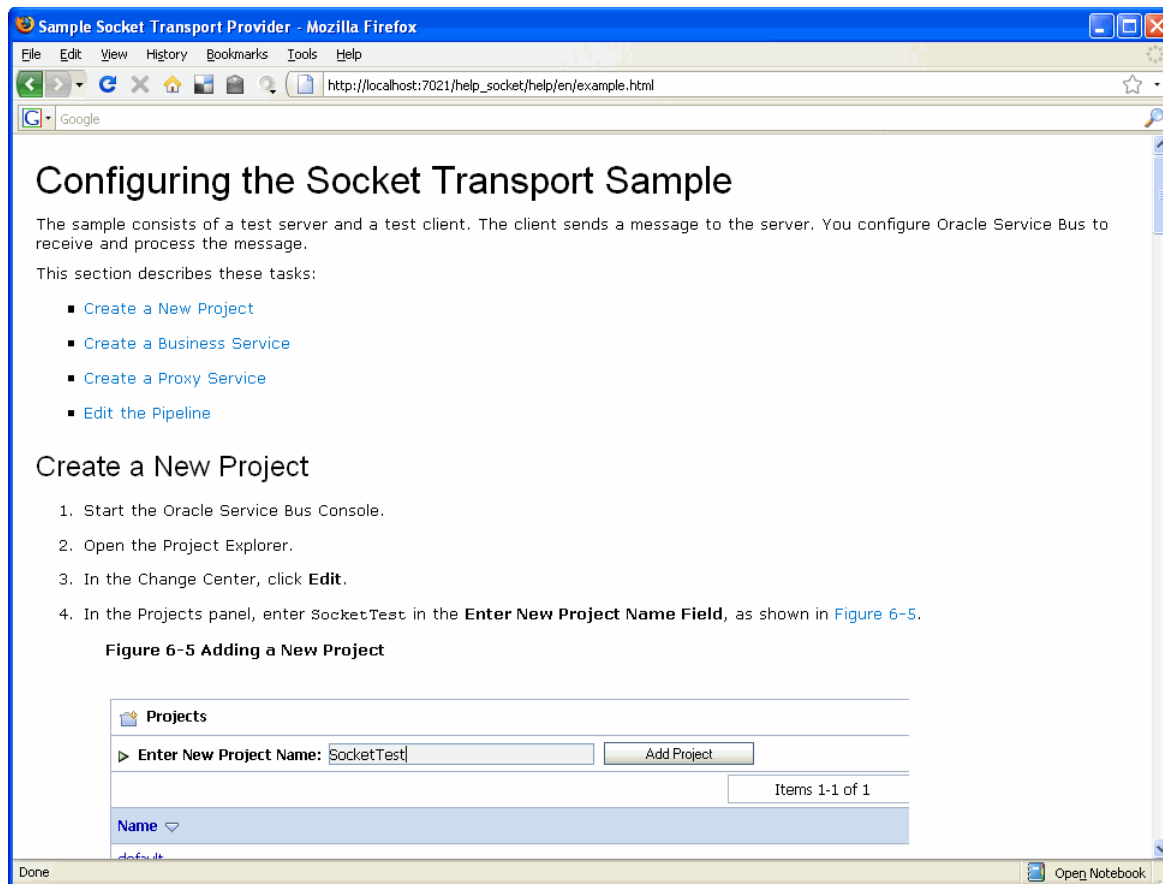
Figure 39–7 Custom Transport Help in Eclipse



Your help topic(s) can appear in different locations of the help system table of contents, as shown in Figure 39–7, depending on how you package and configure your custom transport. For example, you can merge your transport help with the transport section of the Oracle Service Bus help topics, or you can provide your transport help at the top level of the help system.

39.12.1.2 Oracle Service Bus Console Help

You can also provide transport configuration help at run time in the Oracle Service Bus console. The Oracle Service Bus console provides its own integrated help system, but Oracle Service Bus displays custom transport help stand-alone in its own browser window, as shown in Figure 39–8. Custom transport help is displayed when you click Help on the transport configuration page.

Figure 39–8 Custom Transport Help from the Oracle Service Bus Console

The following sections show you how to provide online help for your custom transport in both Eclipse and in the Oracle Service Bus console.

39.12.2 Providing Custom Transport Help in Eclipse

If you make your custom transport available for service configuration in the Eclipse, you can provide help content that appears as context-sensitive help in Eclipse, in the Eclipse help system, or both. This section shows you how.

The sample socket transport and the Oracle Service Bus native transports provide a best-practice reference implementation for Eclipse help and are used as examples in this section.

Transport help is part of the Eclipse plug-in you create for your transport. For details on plug-in creation, see [Chapter 40, "Developing Oracle Service Bus Transports for Eclipse."](#)

39.12.2.1 Providing Context-Sensitive Help in Eclipse

Providing context-sensitive help gives users information about transport configuration directly in Eclipse, where they are configuring the transport.

You can provide context-sensitive help on the transport configuration pages in the service editor in Eclipse.

The following steps are required for providing context-sensitive help:

1. In `plugin.xml`, add an extension for `org.eclipse.help.contexts` that points to a `context.xml` file. See the `org.eclipse.help.contexts` example in [Example 39-7](#).

This entry tells the plug-in where to find the `context.xml` file. The path to `context.xml` is relative to the plug-in root.

2. Create a `context.xml` file that maps the transport configuration user interface context IDs to help files. Oracle Service Bus provides context IDs for custom transports automatically. See [Section 39.12.2.3.3, "context.xml"](#).
3. Create your help topics. See [Section 39.12.2.3.4, "Help Content and Resources."](#)
4. Package all the help files. See [Section 39.12.2.4, "Packaging Help for the Transport Plug-in."](#)

39.12.2.2 Providing Help in the Eclipse Help System

You can add your transport help to the main Eclipse help system. Your help topics appear in the table of contents, as shown in [Figure 39-7](#).

1. In `plugin.xml`, add an extension for `org.eclipse.help.toc` that points to a `toc.xml` file. See the `org.eclipse.help.contexts` example in [Example 39-7](#). Use the following guidance for setting the `primary` attribute.
 - If you are packaging your plug-in as a JAR file, or if you want your transport help to appear in the top level of the help table of contents, as shown in the "Sample Transport" entry in [Figure 39-7](#), set `primary="true"`.
 - If you want your transport help to be merged with the Oracle Service Bus help topics, as shown in [Figure 39-7](#), set `primary="false"`.

To merge your transport help with the main Oracle Service Bus help, your transport plug-in must be packaged as an exploded directory.

For details on plug-in packaging, see [Chapter 40, "Developing Oracle Service Bus Transports for Eclipse."](#)

2. Create a `toc.xml` file that provides the table of contents structure for your transport help. See the examples in [Section 39.12.2.3.2, "toc.xml"](#).
3. Create your help topics. See [Section 39.12.2.3.4, "Help Content and Resources."](#)
4. Package all the help files. See [Section 39.12.2.4, "Packaging Help for the Transport Plug-in."](#)

39.12.2.3 Help Implementation Reference

Eclipse help, which is based on the Eclipse help framework, requires the resources described in this section.

Use this reference section in conjunction with the previous procedures for implementing transport help in Eclipse.

Note: Oracle Service Bus provides a sample help implementation in its sample socket transport, located at `OSB_ORACLE_HOME/samples/servicebus/sample-transport`. The sample transport is a good reference implementation for developing your own custom transports and help. The sample `plugin.xml` is in the `/eclipse` subdirectory, and the help resources are in the `/help` subdirectory.

This section describes the following Eclipse help resources:

- [Section 39.12.2.3.1, "plugin.xml"](#) – The key file that identifies the components you want to add to Eclipse; in this case, an addition to the help system.
- [Section 39.12.2.3.2, "toc.xml"](#) – The hierarchy of your help topics that appears in the help system table of contents, as shown in [Figure 39–7](#).
- [Section 39.12.2.3.3, "context.xml"](#) – Enables context-sensitive help for your transport configuration user interface.
- [Section 39.12.2.3.4, "Help Content and Resources"](#) – The HTML files, CSS file(s), images, and any other help resources you want to provide.

39.12.2.3.1 plugin.xml The plugin.xml file is the key to adding your transport and transport help files to the Eclipse environment. You must add entries in plugin.xml for your help table of contents (toc.xml) and for context-sensitive help (context.xml).

[Example 39–7](#) shows the toc.xml and context.xml (contexts_socketTransport.xml) entries in the sample socket transport's plugin.xml file, located in the `OSB_ORACLE_HOME/samples/servicebus/sample-transport/eclipse` directory.

Example 39–7 Sample transport plugin.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.2"?>
<plugin>
...
  <extension
    point="org.eclipse.help.toc">
    <toc file="/help/en/toc.xml" primary="true"/>
  </extension>
  <extension
    point="org.eclipse.help.contexts">
    <contexts
      file="/help/en/contexts_socketTransport.xml"
      plugin="Socket_Transport"/>
    </extension>
</plugin>
```

Key Parts of plugin.xml

- All paths are relative to the plug-in root directory.
- The `org.eclipse.help.toc` extension point makes the connection to the Eclipse help system left navigation area.

The `<toc file...>` entry references the toc.xml file containing the help topic hierarchy you create for your transport help.
- The `primary="true"` attribute is important. If set to true, your transport table of contents appears at the top level of the Eclipse help system. Set it to true if you are packaging your custom transport plug-in as a JAR file.

If set to false, Eclipse expects your toc.xml to be merged into an existing toc.xml hierarchy, such as the main Oracle Service Bus help system. See the following toc.xml section for more information.
- The entry for the `org.eclipse.help.contexts` lets you implement Eclipse-based context-sensitive (F1) help for your transport. Context-sensitive help topic links appear in the Related Topics area of the Help view in Eclipse.

For details on plug-in packaging, see [Chapter 40, "Developing Oracle Service Bus Transports for Eclipse."](#)

39.12.2.3.2 toc.xml The `toc.xml` file determines how your custom transport help appears in the left navigation area of the Eclipse help system. [Example 39-8](#) shows how you provide your transport help at the top level of the Eclipse help system table of contents.

Example 39-8 Sample Socket Transport `toc.xml` for a Top-Level Entry in the Eclipse Help System

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<?NLS TYPE="org.eclipse.help.toc"?>
<toc label="Socket Transport Sample">
  <topic label="Socket Transport Configuration Page (Business Services)"
    href="help/en/tpSOCKETTransportBizService.html"/>
  <topic label="Socket Transport Configuration Page (Proxy Services)"
    href="help/en/tpSOCKETTransportProxyService.html"/>
  <topic label="Configuring the Socket Transport Sample (Service Bus Console)"
    href="help/en/example.html"/>
</toc>
```

When including your transport help as a top-level entry in the Eclipse help system, be sure to set `primary="true"` in the `org.eclipse.help.toc` extension point in `plugin.xml`.

39.12.2.3.3 context.xml The `context.xml` file, shown in [Example 39-9](#), maps the context IDs of your transport editor page to help topics. When users press F1 on your transport configuration pages, Eclipse displays your help links in the Help view, as shown in [Figure 39-6](#).

Example 39-9 Sample `context.xml` (`contexts_socketTransport.xml`) for the sample socket transport

```
<?xml version="1.0" encoding="UTF-8"?>
<?NLS TYPE="org.eclipse.help.contexts"?>
<contexts>
  <!-- Default Socket Transport help -->
<context id="tpSOCKETTransportBizService"
  title="Socket Transport Configuration page (Business Service)">
  <description>The Sample socket transport illustrates Transport SDK
    concepts.</description>
  <topic href="help/en/tpSOCKETTransportBizService.html"
    label="Socket Transport Configuration Page (Business Services)"/>
</context>
<context id="tpSOCKETTransportProxyService"
  title="Socket Transport Configuration page (Proxy Service)">
  <description>The Sample socket transport illustrates Transport SDK
    concepts.</description>
  <topic href="help/en/tpSOCKETTransportProxyService.html"
    label="Socket Transport Configuration Page (Proxy Services)"/>
</context>
</contexts>
```

Key Points About `context.xml`

- The `context id` attribute value is the context ID of the editor user interface.
- The `topic href` attribute tells Eclipse which help topic to link to when the user presses F1.

- The `topic label` attribute determines the link text that appears in the Related Topics area of the Eclipse Help view.
- The `description` element provides the text above the displayed link when the user presses F1.
- Context IDs for your transport user interfaces are available automatically. Use the patterns in [Table 39–1](#) for the `context id` attribute, following the exact text and case sensitivity. If your `id` values are incorrect, context-sensitive help will not work for your transport.

Table 39–1 Context IDs for Transport Configuration Pages

User Interface Component	Value for Id Attribute
Transport configuration page in the business service editor	<code>tp<TRANSPORT_ID>TransportBizService</code> – For example: <code>tpSOCKETTransportBizService</code>
Transport configuration page in the proxy service editor	<code>tp<TRANSPORT_ID>TransportProxyService</code> – For example: <code>tpSOCKETTransportProxyService</code>

The `<TRANSPORT_ID>` value comes from your implementation of the `TransportProvider` class, where you set the String ID of the transport. Notice that the `<TRANSPORT_ID>` must be all uppercase letters even if you named the transport ID with lowercase letters.

- You can give your `context.xml` a unique name (with an `.xml` extension) and put it anywhere within the plug-in directory as long as you provide the correct path to it in `plugin.xml`.
- All paths in `context.xml` are relative to the plug-in root directory.

39.12.2.3.4 Help Content and Resources You have a lot of flexibility in deciding what type of help content to provide, from a simple page of text with no graphics to multiple pages with many graphics, PDF files, embedded video and so on.

For example, you could create a single HTML file and reference it from the `toc.xml` and `context.xml` files; or you could create separate help files that describe the transport configuration fields for business services and proxy services and also provide a high-level overview, pointing at the three help files in different combinations from `toc.xml` and `context.xml`.

You can store your help topics and resources anywhere in your transport plug-in, as long as you reference them correctly in `toc.xml` and/or `context.xml`.

Because of potential user interface and functionality differences between transport configuration in Eclipse and the Oracle Service Bus console, consider creating separate help topics for Eclipse and for the Oracle Service Bus console.

39.12.2.4 Packaging Help for the Transport Plug-in

Your transport plug-in should contain the following:

- The `plugin.xml` file
- A transport JAR containing your transport classes and supporting files
- A help directory containing the `toc.xml`, `context.xml`, and help files

Whether you package your transport plug-in as a JAR or as an exploded directory, following is a recommended packaging structure for your transport help with relation to other resources:


```
/plugin_root
plugin.xml
transport.jar
/help
/en (locale)
toc.xml
context.xml
/html
```

<help files and resources>

Notice that with the /en directory the help is packaged to support localization. To provide localization, you must create a plug-in for each locale, as described in the Eclipse documentation.

Note: You can also package your help files in a doc.zip file. For more information, see "Help server and file locations" in the *Eclipse Platform Plug-In Developer Guide* at http://help.eclipse.org/galileo/topic/org.eclipse.platform.doc.isv/guide/ua_help_content_files.htm.

39.12.2.5 Related Topics

For complete information on the Eclipse help framework, see the Eclipse help system at

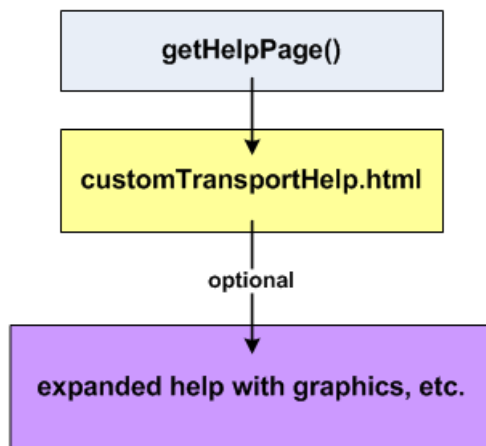
http://help.eclipse.org/galileo/topic/org.eclipse.platform.doc.isv/guide/ua_help.htm.

For information on plug-in packaging, see [Chapter 40, "Developing Oracle Service Bus Transports for Eclipse."](#)

39.12.3 Providing Custom Transport Help in the Oracle Service Bus Console

This section shows you how to provide help for your custom transport at run time in the Oracle Service Bus console. Oracle Service Bus displays custom transport help as a stand-alone help page in a browser, as shown in [Figure 39-8](#).

[Figure 39-9](#) provides a high-level view of the Oracle Service Bus console help framework for custom transports.

Figure 39–9 Oracle Service Bus Console Help Framework

By implementing a specific Oracle Service Bus interface, you use the `getHelpPage()` method to launch a single HTML page when the user clicks Help in the Oracle Service Bus console when your user interface has focus. The HTML file can contain the following:

- Text, inline CSS definitions, inline JavaScript functions
- References to graphics and other resources, as long as those resources are hosted in a Web application or an external Web site

In most situations, you should be able to provide all the help for your custom transport with text and inline formatting.

However, if you want to provide full-featured Web-based help that includes graphics and other external resources, those resources must be hosted in a Web application or an external Web site. You must either reference those external resources in the HTML file or provide a link from the HTML file to an external location. For example, the sample socket transport help provides a link from the starting HTML file to a help topic with graphics that is running in a custom Web application. Using an embedded JavaScript call, you could also set up your HTML file to automatically redirect to the expanded help URL you want.

Following are the tasks involved in creating custom transport help in the Oracle Service Bus console:

- [Section 39.12.3.1, "Implementing the CustomHelpProvider Interface"](#)
- [Section 39.12.3.2, "Creating an HTML File to Launch"](#)
- [Section 39.12.3.3, "Creating a Simple Web Application to Display Expanded Help \(Optional\)"](#)
- [Section 39.12.3.4, "Packaging Transport Help for the Oracle Service Bus Console"](#)

39.12.3.1 Implementing the CustomHelpProvider Interface

To develop the configuration user interface for your custom transport, you implement the `TransportUIBinding` interface in a custom class. To provide help for your transport configuration user interface in the Oracle Service Bus console, you must also implement the `CustomHelpProvider` interface. `CustomHelpProvider` contains the `getHelpPage()` method you need to launch help for your transport configuration page in the Oracle Service Bus console.

The sample socket transport implements CustomHelpProvider in its SocketTransportUIBinding.java class, located at `OSB_ORACLE_HOME/samples/servicebus/sample-transport/src/com/bean/alsb/transports/sock`.

[Example 39–10](#) contains snippets that illustrate the implementation of CustomHelpProvider.

Example 39–10 Implementing CustomHelpProvider to provide help for your transport in the Oracle Service Bus console

```
public class SocketTransportUIBinding
    implements TransportUIBinding, CustomHelpProvider {
    .
    .
    .
    public Reader getHelpPage() {
        String helpFile = "help/en/contexts_socketTransport.html";
        ClassLoader clLoader = Thread.currentThread().getContextClassLoader();
        InputStream is = clLoader.getResourceAsStream(helpFile);
        InputStreamReader helpReader = null;
        if(is!=null)
            helpReader = new InputStreamReader(is);
        else
            SocketTransportUtil.logger
                .warning(SocketTransportMessagesLogger.noHelpPageAvailableLoggable()
                    .getMessage(uiContext.getLocale()));
        return helpReader;
    }
}
```

In [Example 39–10](#), `Reader getHelpPage()` returns a Reader stream that the Oracle Service Bus console uses to send the HTML page to the browser. The `helpFile` path is relative to the root within the transport JAR.

If you are providing help in multiple languages, use `TransportUIContext.getLocale()` to help provide the appropriate path to the localized content; in this case, providing the locale value for `/help/<locale>/your.html`.

39.12.3.2 Creating an HTML File to Launch

Create an HTML file for the `getHelpPage()` method to launch, as illustrated by `help/en/contexts_socketTransport.html` in [Example 39–10](#).

If you want to keep your help implementation simple, create the HTML file to use text, inline CSS definitions, and inline JavaScript functions. If you do this, you do not need to create a separate Web application to host graphics or other external resources.

However, if you want to provide more expanded help with graphics and other resources, reference those external resources in your HTML file, such as

```
img src="/help_socket/help/en/wwimages/addProject.gif"
```

or

```
a href="http://www.yoursite.com"
```

You can also set the HTML file up to automatically redirect to the expanded help with an embedded JavaScript call, as shown in [Example 39–11](#), which redirects from the sample socket transport HTML page to the expanded `help_socket` Web application help content.

Example 39–11 JavaScript function that provides a redirect

```
<script language="JavaScript" type="text/javascript">
<!-- Begin
window.location="/help_socket/help/en/example.html";
// End -->
</script>
```

The sample socket transport HTML file provides a link to its expanded help. The HTML file, `contexts_socketTransport.html`, is located at `OSB_ORACLE_HOME/samples/servicebus/sample-transport/resources/help/en/`.

39.12.3.3 Creating a Simple Web Application to Display Expanded Help (Optional)

If you want to go beyond a basic text HTML file for your transport help, you can provide expanded help with graphics and other resources in various ways:

- Link from the self-contained HTML file to an existing URL; for example, if you have an existing Web site that contains your transport documentation. All that is required is that you provide a link to the URL from the self-contained HTML file. You can also insert references to graphics and other resources hosted on an external site.
- Create a Web application for the expanded help, bundle it with your transport, and link to it or reference graphics and other resources from the HTML file. This topic provides instructions on creating a Web application that is bundled in your transport EAR to display your expanded transport help.

Create the following files for your Web application:

39.12.3.3.1 META-INF/application.xml In `application.xml`, give your Web application a context root that is used for the Web application's root URL. For example, [Example 39–12](#) shows the context root for the sample socket transport Web application.

Example 39–12 application.xml for the sample socket transport Web application

```
<application>
  <display-name>Socket Transport</display-name>
  <description>Socket Transport</description>
  <module>
    <web>
      <web-uri>webapp</web-uri>
      <context-root>help_socket</context-root>
    </web>
  </module>
</application>
```

The sample socket transport `application.xml` file is located at `OSB_ORACLE_HOME/samples/servicebus/sample-transport/META-INF/`.

This entry maps the file system directory `/webapp` to an alias Web application root URL:

```
http://server:port/help_socket/
```

With your help files inside the Web application in a directory such as `/help/en/`, the full URL to your expanded help would be:

```
http://server:port/help_socket/help/en/index.html
```

But your internal links to it only need to be

/help_socket/help/en/index.html

where index.html is the landing HTML page.

39.12.3.3.2 WEB-INF/web.xml In web.xml, enter a display name and description for the Web application. This is standard deployment descriptor information. For example, [Example 39–13](#) shows the name and description of the sample socket transport Web application.

Example 39–13 web.xml for the sample socket transport Web application

```
<web-app>
  <display-name>Sample Socket Transport Help WebApp</display-name>
  <description>
    This webapp implements the help webapp for the socket transport.
  </description>
</web-app>
```

The sample socket transport web.xml file is located at `OSB_ORACLE_HOME/samples/servicebus/sample-transport/webapp/WEB-INF/`.

39.12.3.3.3 Help Content and Resources Create and package your expanded help files inside the Web application directory. In the sample socket transport, the help files are stored in `OSB_ORACLE_HOME/samples/servicebus/sample-transport/resources/help/en`.

Note: The reason the socket transport help files are not stored in the /webapp directory is because the help directory contains help files and resources for both the Eclipse plug-in and the Oracle Service Bus console. When the sample socket ANT build creates the transport JAR, transport EAR, and Eclipse plug-in, it packages the help in different ways. For the transport EAR build, it moves the help files under the /webapp directory.

Because of potential user interface and functionality differences between transport configuration in Eclipse and the Oracle Service Bus console, consider creating separate help topics for Eclipse and for the Oracle Service Bus console.

39.12.3.4 Packaging Transport Help for the Oracle Service Bus Console

Your transport EAR should contain the following:

- A transport JAR stored in APP-INF/lib containing:
 - Your transport classes and supporting files
 - The HTML file for your transport help, ideally in a directory such as help/en/ for localization support
- Optionally, a Web application containing expanded help for your transport

Developing Oracle Service Bus Transports for Eclipse

This chapter documents the best practices, design considerations, and packaging to develop transports for Oracle Service Bus design time. The Transport SDK interface provides a bridge between transport protocols and the Oracle Service Bus run time.

Tip: Before you begin this chapter, review [Chapter 38, "Design Considerations."](#)

This chapter includes the following sections:

- [Section 40.1, "Introduction"](#)
- [Section 40.2, "Services Runtime and Services Configuration"](#)
- [Section 40.3, "Packaging Transports as Eclipse Plug-Ins"](#)
- [Section 40.4, "Reference"](#)

40.1 Introduction

Oracle Service Bus transports were originally designed to be deployed on Oracle Service Bus servers and configured through the Oracle Service Bus console. With design environments like Eclipse, some modifications to the SDK and the existing transports are necessary.

This document describes the additional steps to ensure Oracle Service Bus transports design time can be used on platforms outside the Oracle Service Bus console.

The sample socket transport was ported to Eclipse and can be considered a best practice for Eclipse integration. The sample socket resources are located at `OSB_ORACLE_HOME/samples/servicebus/sample-transport/`. The Java source files are in the `/src` subdirectory. The sample also contains a build script that automatically packages the sample socket transport for both Eclipse integration and Oracle Service Bus console deployment. For information on building and deploying the sample socket transport, see [Chapter 42, "Sample Socket Transport Provider."](#)

40.2 Services Runtime and Services Configuration

When you develop a transport, you should distinguish the runtime aspects from the configuration aspects. The runtime aspects include proxy or business service deployment and service run-time invocation. The configuration aspects include proxy or service configuration and validation.

The runtime aspects do not need to change since they are always exercised in the context of a running Oracle Service Bus server. However, the configuration aspects are dependent on the design environment.

Developers should consider three different modes:

1. Online mode – The services for the custom transport are configured with the Oracle Service Bus console on a running Oracle Service Bus server.
2. Offline mode – The transport is configured with a design environment running outside the Oracle Service Bus server. No remote server is available.
3. Offline mode with remote server – The transport is configured with a design environment running outside the Oracle Service Bus server. However, a remote server is available and can be used for both validation and configuration purposes.

Transports running in Eclipse must support offline mode and, optionally, offline mode with a remote server.

This section contains the following topics:

- [Section 40.2.1, "Offline Methods"](#)
- [Section 40.2.2, "Restrictions when Working Offline"](#)
- [Section 40.2.3, "Working Offline with a Remote Server"](#)
- [Section 40.2.4, "Bootstrapping Transports in Offline Mode"](#)
- [Section 40.2.5, "Packaging Transports in Offline Mode"](#)
- [Section 40.3, "Packaging Transports as Eclipse Plug-Ins"](#)
- [Section 40.4, "Reference"](#)

40.2.1 Offline Methods

When you deploy a transport in offline mode, the configuration framework creates a single session for all the resource configurations. This session is never activated. Since proxy or business services can only be deployed on a running Oracle Service Bus server, there is no need to activate the session. However, it is still important to detect conflicts and configuration errors and the validation methods are still exercised.

Following is a list of the minimum set of classes and methods defined by the Transport SDK that must be implemented in offline mode. The exceptions were removed from the methods signature for better readability.

Note: You do not need to completely re-implement your transport for offline mode. In most cases your transport will only need a few changes to existing methods to support both online and offline modes.

Classes and Methods You Must Implement for Offline Mode

- The public interface `TransportProvider` class

```
String getId();
void validateEndPointConfiguration(TransportValidationContext context);
SchemaType getEndPointConfigurationSchemaType();
SchemaType getRequestMetaDataSchemaType();
SchemaType getRequestHeadersSchemaType();
SchemaType getResponseMetaDataSchemaType();
SchemaType getResponseHeadersSchemaType();
TransportProviderConfiguration getProviderConfiguration();
```



```

TransportUIBinding getUIBinding(TransportUIContext context);
void shutdown();
Collection<NonQualifiedEnvValue> getEnvValues(Ref ref, EndPointConfiguration epConfig);
void setEnvValues(Ref ref, EndPointConfiguration epConfig, Collection<NonQualifiedEnvValue>
envValues);
Collection<Ref> getExternalReferences(EndPointConfiguration epConfig);
void setExternalReferences(Map<Ref, Ref> mapRefs, EndPointConfiguration epConfig);
Map<String, String> getBusinessServicePropertiesForProxy(Ref ref);
XmlObject getProviderSpecificConfiguration(Ref ref, Map<String, String> props);

```

- The public interface `TransportProviderFactory` class
This interface registers transports in offline mode. See [Chapter 43, "Deploying a Transport Provider,"](#) for more details.
- The public interface `TransportUIBinding` class
You should implement all the methods in this interface and define the user interface used to configure a proxy or business service.

Helper Classes

- The public class `TransportManagerHelper` class
This class, which is typically used by `TransportProvider` developers, provides a boolean `isOffline()` method to help the provider implementor determine whether the code is running offline or not.

Some of the methods that are not valid in offline mode will throw exceptions, which are described below. Other methods are meant only for runtime or deployment, such as `public isAdmin()`.

The following methods are also available when working in offline mode with remote server:

- The public `Set<String> getDispatchPolicies(JMXConnector connector);` method
- The public `DomainRuntimeServiceMBean getDomainRuntimeServiceMBean(JMXConnector connector);` method

See [Section 40.2.3, "Working Offline with a Remote Server"](#) for more information.

Do not invoke the following methods in offline mode:

- The `isAdmin()` method throws exceptions (`public static boolean isAdmin();`). This method throws a `java.lang.IllegalStateException` message.
- The `clusterExists()` method always returns false (`public static boolean clusterExists();`). This method always returns false.

Note: The preferred method for checking run-time status is `isRuntimeEnabled()`, in conjunction with `getRuntimeServers()`.

40.2.2 Restrictions when Working Offline

When you work offline, none of Oracle WebLogic Server services running on the server are available. Do not use these services inside the methods described in [Section 40.2.1, "Offline Methods."](#)

Following are examples of restrictions for working offline:

- The WLS MBeans are not available.
- The server Java properties are not available.
- You cannot access the JNDI tree directly. However, if JNDI properties are defined in the service configuration, you can attempt to use them.
- You can not determine if the service is going to run in a cluster or a standalone server.
- You do not have access to Oracle WebLogic Server security infrastructure.
- You do not have access to any static singleton service located on the server.

Because some of the services are not available, it is necessary to evaluate how the transport user interface is affected. In general, the user interface should be more flexible to let users manually configure values instead of trying to retrieve values from the server environment.

Eclipse design time does not currently support deployment to an Oracle Service Bus clustered environment. Therefore, the user interface must be populated as if there is no cluster. If necessary, the user can use a customization file to update the configuration and force a deployment to an Oracle Service Bus cluster.

For example, some transports retrieve the list of available WorkManager items by using the TransportManagerHelper and letting the user pick one through a list. However, in offline mode, the MBeans are not available so the list cannot be populated. The transport provider has two choices:

1. Let the user type the correct WorkManager name. In that case, the user interface must be changed to be a text box and not a list when working offline.
2. Another less flexible option is to populate the list with just the default WorkManager. When the service is pushed to a running Oracle Service Bus server, the WorkManager name can be switched using an environment value substitution.

40.2.3 Working Offline with a Remote Server

When you work offline, a remote server might be available. For instance, when you configure a service on Eclipse, the user can associate a remote Oracle Service Bus server to the current project. The transport provider can take advantage of the remote server by accessing the Oracle WebLogic Server MBeans and retrieving information. This mode is similar to working online; however, some restrictions still apply since the code is not running on the server and only the MBeans are available.

When you work offline, the following statements apply:

- The server Java properties are not available.
- You cannot use many of TransportManagerHelper methods as described in [Section 40.2.1, "Offline Methods."](#)
- You cannot access the JNDI tree directly. However if JNDI properties are defined in the service configuration, you can attempt to use them.
- You do not have access to any static singleton service located on the server.

To access the MBeans, the framework provides an instance of `JMXConnector` when it requests the `TransportUI` object, or when it asks the provider to validate a configuration. The `JMXConnector` is available in the `TransportUIContext` or the `TransportValidationContext`:

```
JMXConnector connector =
(JMXConnector)uiContext.get(TransportValidationContext.JMXCONNECTOR);
```

For more information, see the sample transport in [Section 40.4, "Reference."](#)

If the connector is not present, a remote server is not available. This connector object can then be used to access the MBeans. Helper methods have been added to the `TransportManagerHelper` to retrieve the list of `WorkManager` and Oracle WebLogic Server domain MBean.

Note: This behavior is generalized for both online and offline modes. The public static `Set<String> getDispatchPolicies()` method defined in the `TransactionManagerHelper` will be deprecated and must be replaced by the same method with `JMXConnector` as a parameter. If you do not replace it, the following error appears:
`com.bea.wli.sb.transports.TransportException.`

40.2.4 Bootstrapping Transports in Offline Mode

In online mode, transports must be packaged as EAR files and deployed on an Oracle Service Bus server. When the EAR is loaded at startup, the transport registers a callback on a startup event and registers an instance of the `TransportProvider` to the `TransportManager`.

In offline mode, the SDK provides an interface called `com.bea.wli.sb.transports.TransportProviderFactory` that registers transports. A transport developer must implement this interface and must make the default constructor public. The interface is provided in [Section 40.4, "Reference,"](#) as well as a sample implementation.

If the `TransportProvideFactory` is instantiated, you can assume the transport needs to work in offline mode (with or without a remote server).

Note: You can set a boolean operator in the `TransportManagerHelper` when the constructor is invoked to determine if the transport is running in offline mode. This information can also be passed in the `TransportUIContext` and the `TransportValidationContext`. Your engineering department can assist you in making this decision.

40.2.5 Packaging Transports in Offline Mode

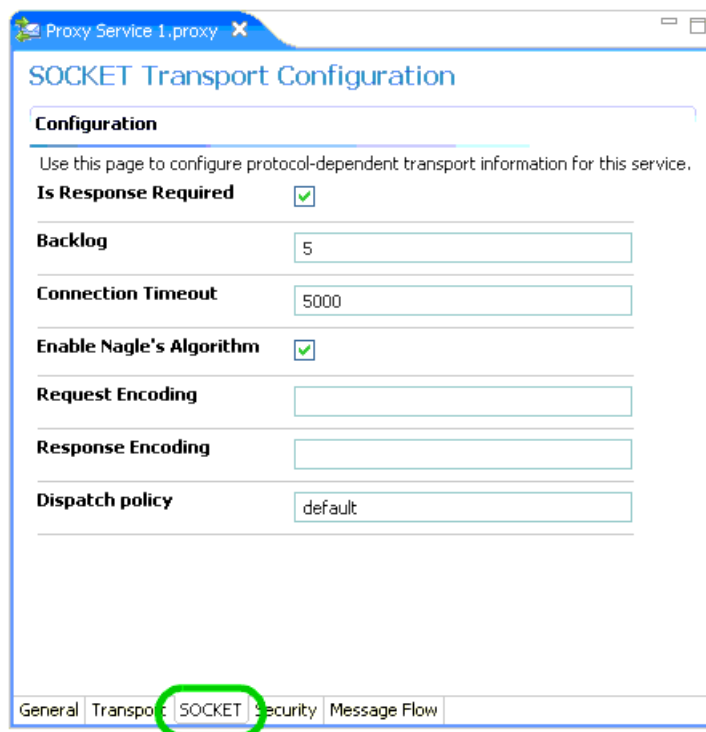
In offline mode, you can use transports in different design environments. The Eclipse environment defines specific packaging, which is described in the next section. In general, transports simply need to be available to external design time environments as a self-contained JAR file. A self-contained JAR file has the JAR file include the `transport config.xml` file, the header, metadata schemas, XBeans classes, `TransportManagerHelper` implementation, and the compiled transport classes.

40.3 Packaging Transports as Eclipse Plug-Ins

Packaging your custom transport as an Eclipse plug-in, in conjunction with your transport user interface implementation, lets service developers select and configure your transport in the development environment.

Figure 40–1 shows the service editor—after a service has been created—with a configuration page for the selected transport.

Figure 40–1 Transport Configuration Page in Eclipse



Note: Your implementation of the `TransportUIBinding` interface determines the user interface for selecting and configuring your transport, both in Eclipse and in the Oracle Service Bus console.

Oracle Service Bus provides Eclipse-based plug-ins to Eclipse. The core Oracle Service Bus plug-in, `com.bea.alsb.core`, defines an extension point (`com.bea.alsb.core.transports`) that is used to register Oracle Service Bus transports in the Eclipse environment as plug-ins.

40.3.1 Transport Plug-in Resources

Your transport plug-in must provide the following resources:

- MANIFEST.MF file – Contains key information about your transport plug-in. Use the sample socket transport MANIFEST.MF for reference. See [Section 40.4.5, "MANIFEST.MF."](#)
- In the `plugin.xml` file, provide an extension point that registers your transport with Eclipse as a plug-in. Use the sample socket transport plug-in as a reference. See [Section 40.4.4, "plugin.xml."](#)

- A transport JAR file containing your transport implementation.
- (Optional) Resources for providing online help.

40.3.2 Transport Plug-in Packaging

You can package your transport plug-in either as a JAR file or in an exploded directory. Packaging as a JAR makes the transport more portable. Packaging as an exploded directory lets you reference resources in other Oracle Service Bus plug-ins. For example, if you want to merge your transport help with the Oracle Service Bus transport topics in the Eclipse help system, you must package your transport plug-in in an exploded directory, as described in [Section 39.12.2, "Providing Custom Transport Help in Eclipse."](#)

Whether you package your plug-in in a JAR or an exploded directory, you must package your transport implementation in a JAR file.

Use the following guidance for packaging your transport as a plug-in:

- To construct the plug-in JAR or exploded directory name, append the `Bundle-Version` to the `Bundle-SymbolicName` from the MANIFEST.MF file. For example, the sample socket transport JAR is named `Socket_Transport_3.0.0.0.jar`.
- Package with the following directory structure, as shown in [Figure 40-2](#):
 - `plugin.xml` (root of the plug-in)
 - `/lib/your_transport.jar` (transport classes and resources)
 - `/META-INF/Manifest.mf`
 - `/help` – If you are providing help for your transport, include a `/help` directory for your help resources, as described in [Section 39.12.2, "Providing Custom Transport Help in Eclipse."](#)

Figure 40-2 Plug-In Packaging

Name	Path
Manifest.mf	meta-inf\
sock_transport.jar	lib\
plugin.xml	

For reference on transport plug-in packaging, build the sample socket transport, as described in [Chapter 42, "Sample Socket Transport Provider."](#) View the generated `Socket_Transport_3.0.0.0.jar`.

40.4 Reference

This section contains the following topics:

- [Section 40.4.1, "Working in Different Modes"](#)
- [Section 40.4.2, "TransportProviderFactory"](#)
- [Section 40.4.3, "Extension Point Schema"](#)
- [Section 40.4.4, "plugin.xml"](#)
- [Section 40.4.5, "MANIFEST.MF"](#)

- [Section 40.4.6, "Build.xml"](#)
- [Section 40.4.7, "TransportManagerHelper Methods"](#)

40.4.1 Working in Different Modes

Dispatch Policies are used by most transports and allow services throttling. This code distinguishes the three modes described in [Section 40.2, "Services Runtime and Services Configuration"](#):

- Online mode
- Offline mode
- Offline mode with remote server

The connection to the remote server is retrieved from the context, as shown in [Example 40–1](#).

Example 40–1 Connection to the Remote Server

```
/**
 * Builds the dispatch policies in the ui object.
 *
 * @param curDispatchPolicy
 * @return TransportEditField containing existing dispatch policies.
 */
public TransportEditField getDispatchPolicyEditField(String curDispatch
Policy) {
    TransportUIFactory.TransportUIObject uiObject = null;
    Set<String> wmSet = null;

    if (SocketTransportManagerHelper.isOffline()) { // if on
        Eclipse try to get the MBeans from the UIContext
        JMXConnector connector = (JMXConnector)uiContext.get
            (TransportValidationContext.JMXCONNECTOR);
        if (connector != null) {
            try {
                wmSet = TransportManagerHelper.getDispatchPolicies
                    (connector);
            } catch (Exception ex) {
                wmSet = null; //continue
            }
        }
    } else { // if running on the server use the helper to get the
        policies
        try {
            wmSet = TransportManagerHelper.getDispatchPolicies();
        } catch (TransportException e) {
            SocketTransportUtil.logger
                .error(SocketTransportMessagesLogger.
                    noDispatchPolicies(), e);
        }
    }

    if (wmSet == null) // if JMXConnector not available or impossible
        to connect provide a simple edit field
    {
        uiObject = TransportUIFactory.createTextBox
            (curDispatchPolicy);
    } else // create a drop down list
    {
```

```

        // adding default work manager to the list.
        wmSet.add(DEFAULT_WORK_MANAGER);
        String[] values = wmSet.toArray(new String[wmSet.size()]);
        uiObject = TransportUIFactory.createSelectObject(
            values,
            values,
            curDispatchPolicy,
            TransportUIFactory.SelectObject.DISPLAY_LIST,
            false);
    }
    return TransportUIFactory.createEditField(DISPATCH_POLICY,
        TextMessages.getMessage(TextMessages.DISPATCH_POLICY,
            locale),
        TextMessages.getMessage(TextMessages.DISPATCH_POLICY_INFO,
            locale), uiObject);
}

```

40.4.2 TransportProviderFactory

TransportProviderFactory, [Example 40–2](#), lets you provide development-time functionality in Eclipse.

Example 40–2 The TransportProviderFactory Class

```

package com.bea.wli.sb.transports;

import com.bea.wli.sb.transports.TransportException;
import com.bea.wli.sb.transports.TransportManager;

/**
 * This interface is the extension point to plug custom ALSB transports in
 * Eclipse.
 * The implementation must declare the default class constructor.
 */
public interface TransportProviderFactory {

    /**
     * Registers a new provider with the transport manager. Typically
     * called by the ALSB core eclipse plugin.
     * @param tm the transport manager to which to register
     */
    public void registerProvider(TransportManager tm) throws
        TransportException;

    /**
     * @return a unique string that identifies this provider, like "http"
     * This method must return the same ID than provider.getId()
     */
    String getId();
}

```

The code sample in [Example 40–3](#) shows how the Socket Transport implements this interface

Example 40–3 Example of the Socket Transport Implementing the Interface

```

package com.bea.alsb.transports.sock;

import com.bea.wli.sb.transports.TransportManager;
import com.bea.wli.sb.transports.TransportException;
import com.bea.wli.sb.transports.TransportProviderFactory;

```

```

public class SocketTransportProviderFactory implements
    TransportProviderFactory {

    public static boolean isOffline() {
        return isOffline;
    }

    private static boolean isOffline = false;

    public void registerProvider(TransportManager tm) throws
        TransportException {
        isOffline = true;
        SocketTransportProvider instance =
            SocketTransportProvider.getInstance();
        tm.registerProvider(instance, null);
    }

    public String getId() {
        return SocketTransportProvider.ID;
    }
}

```

40.4.3 Extension Point Schema

[Example 40–4](#) is extracted from the extension point schema that defines the transport element and transport-provider attribute for adding a transport as a Eclipse plug-in, shown in [Section 40.4.4, "plugin.xml."](#)

Example 40–4 Part of the Extension Point Schema

```

<element name="transport">
    <complexType>
        <attribute name="transport-provider" type="string" use="required">
            <annotation>
                <documentation>
                </documentation>
                <appInfo>
                    <meta.attribute kind="java" basedOn="
                        com.bea.wli.sb.transports.Transport
                        ProviderFactory"/>
                </appInfo>
            </annotation>
        </attribute>
    </complexType>
</element>

```

40.4.4 plugin.xml

[Example 40–5](#) shows the transport extension for the sample socket transport plugin.xml file. The file is located at `OSB_ORACLE_HOME/samples/servicebus/sample-transport/eclipse/`.

Example 40–5 Plugin.xml File

```

<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.2"?>
<plugin>
    <extension
        id="socket"
        name="Socket Transport"

```



```

        point="com.bea.alsb.core.transports">
        <transport transport-provider="com.bea.alsb.transports
            .sock.SocketTransportProviderFactory"/>
    </extension>
</plugin>

```

Key Points About plugin.xml

- The extension `point` attribute value is always `com.bea.alsb.core.transports` for transports.
- The `transport-provider` attribute is the fully qualified path to your `TransportProviderFactory` implementation class.

Note: Transport providers typically are not required to manage the life cycle inside Eclipse, so there is no need to extend the `org.eclipse.core.runtime.Plugin` class like a regular plug-in.

If you are providing help for your custom transport in Eclipse, you will also have help entries in `plugin.xml`. For more information, see [Section 39.12, "Creating Help for Custom Transports."](#)

40.4.5 MANIFEST.MF

[Example 40–6](#) shows the sample socket transport MANIFEST.MF file. The file is located at `OSB_ORACLE_HOME/samples/servicebus/sample-transport/eclipse/META-INF/`.

Example 40–6 Sample MANIFEST.MF File

```

Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Socket Transport Plug-in
Bundle-SymbolicName: Socket_Transport;singleton:=true
Bundle-Version: 3.0.0.0
Bundle-Localization: plugin
Bundle-ClassPath: .,
    lib/sock_transport.jar
Require-Bundle: com.bea.alsb.core

```

40.4.6 Build.xml

For an example build file for compiling and packaging a transport, see the socket sample build file `OSB_ORACLE_HOME/samples/servicebus/sample-transport/build.xml`.

For more information on building the sample socket transport, see [Chapter 42, "Sample Socket Transport Provider."](#)

40.4.7 TransportManagerHelper Methods

[Example 40–7](#) shows the `TransportManagerHelper` methods.

Example 40–7 TransportManagerHelper Methods

```

public static Set<String> getDispatchPolicies(JMXConnector connector)
    throws TransportException;

```

```
public static DomainRuntimeServiceMBean  
getDomainRuntimeService(JMXConnector connector)  
    throws TransportException;
```

Transport SDK Interfaces and Classes

This chapter lists and summarizes the classes and interfaces provided by the Transport SDK. For information on which interfaces are required to develop a custom transport provider, see [Chapter 39, "Developing a Transport Provider."](#)

This chapter includes these sections:

- [Section 41.1, "Introduction"](#)
- [Section 41.2, "Schema-Generated Interfaces"](#)
- [Section 41.3, "General Classes and Interfaces"](#)
- [Section 41.4, "Source and Transformer Classes and Interfaces"](#)
- [Section 41.5, "Metadata and Header Representation for Request and Response Messages"](#)
- [Section 41.6, "User Interface Configuration"](#)

41.1 Introduction

The classes and interfaces discussed in this chapter are located in `OSB_ORACLE_HOME/lib/sb-kernel-api.jar` unless otherwise noted.

For details on classes and methods, see the *Oracle Fusion Middleware Java API Reference for Oracle Service Bus*.

41.2 Schema-Generated Interfaces

A number of interfaces are generated from XML Schema by an XML Schema compiler tool. The source (XML Schema) for the following interfaces is provided in the file `TransportCommon.xsd`. This file is the base schema definition file for service endpoint configurations. This file is located in `OSB_ORACLE_HOME/lib/sb-schemas.jar`

where `OSB_ORACLE_HOME` is the directory in which you installed Oracle Service Bus.

Schema-Generated Interfaces

- **EndPointConfiguration** – The base type for endpoint configuration. An endpoint is an Oracle Service Bus resource where messages are originated or targeted. `EndPointConfiguration` describes the complete set of parameters necessary for the deployment and operation of an inbound or outbound endpoint.
- **RequestMetaDataXML** – The base type for the metadata of an inbound or outbound request. Metadata is not carried in the payload of the message, but

separately and is used as the "context" for processing the message. Examples of such information that might be transmitted in the metadata are the Content-Type header, security information, or locale information.

- **RequestHeadersXML** – The base type for a set of inbound or outbound request headers.
- **ResponseMetaDataXML** – The base type for response metadata for an inbound or outbound message.
- **ResponseHeadersXML** – The base type for a set of response headers.
- **TransportProviderConfiguration** – Allows you to configure (a) whether this provider generates a service description (for example, WSDL) for its endpoints; (b) whether or not this provider supports inbound (proxy) endpoints; or (c) whether or not this provider supports outbound (business service) endpoints.

41.3 General Classes and Interfaces

This section summarizes general classes and interfaces of the Transport SDK.

This section includes these topics:

- [Section 41.3.1, "Summary of General Classes"](#)
- [Section 41.3.2, "Summary of General Interfaces"](#)

Note: For detailed information on each class and interface listed in this section, refer to the *Oracle Fusion Middleware Java API Reference for Oracle Service Bus*.

41.3.1 Summary of General Classes

class TransportManagerHelper – Helper class that allows the client to execute some common tasks with respect to the transport subsystem.

class ServiceInfo – Wrapper class that describes information about a service, such as its transport configuration and its binding type.

class TransportOptions – Supplies options for sending or receiving a message. There are two styles for using TransportOptions: multiline setup, and single-line use.

class EndPointOperations – Describes different types of transport endpoint lifecycle-related events by which the transport provider is notified. Nested classes include: CommonOperation, Create, Delete, EndPointOperationTypeEnum, Resume, Suspend, and Update.

class Ref – Uniquely represents a resource, project or folder that is managed by the Configuration system. This class is located in *MW_HOME/modules/com.bea.common.configfwk_<version>.jar*.

class TransportValidationContext – Container that supplies information to transport providers that can be used when implementing validation checks of endpoint configuration.

class Diagnostics – Contains a collection of Diagnostic entries relevant to a particular resource. This class is located in *MW_HOME/modules/com.bea.common.configfwk_<version>.jar*.

class Diagnostic – Represents a particular validation message related to a resource. Diagnostic objects are generated as a result of validation that is performed when a

resource changes. Such changes in the system trigger validation for the changed resource, as well as all other resources that (transitively) depend on the changed resource. This class is located in *MW_HOME/modules/com.bea.common.configfwk_version.jar*.

class NonQualifiedEnvValue – Represents an instance of an environment-dependent value in configuration data. Environment-dependent values normally change when moving the configuration from one domain to another. For example the URI of a service could be different on test domain and production domains. This class is located in *MW_HOME/modules/com.bea.common.configfwk_version.jar*.

41.3.2 Summary of General Interfaces

interface TransportManager – A singleton object that provides the main point of centralization for managing different transport providers, endpoint registration, control, processing of inbound and outbound messages, and other points.

interface TransportProvider – Represents the central point for management of transport protocol-specific configuration and runtime properties. There is a single instance of TransportProvider for every supported protocol. For example, there is a single instance of HTTP transport provider, JMS transport provider.

interface BindingTypeInfo – Describes the binding details of the service. The implementation is a convenience wrapper class around several internal Oracle Service Bus structures. Additional methods can be added as needed by transport providers.

interface TransportWLSArtifactDeployer – The plug-in interface for modules that need to deploy/undeploy/modify WLS related artifacts along with an Oracle Service Bus deployment. For example, in certain cases, WLS queues need to be deployed in response to the creation of a service.

Tip: For more information, see [Section 39.11, "When to Implement TransportWLSArtifactDeployer."](#)

interface SelfDescribedTransportProvider – Extends TransportProvider. Those transport providers that generate a service binding type description from a given transport endpoint need to implement this interface. An example is the EJB transport provider.

interface SelfDescribedBindingTypeInfo – Extends the BindingTypeInfo interface for those services that are self-described (for example, EJB services).

interface WsdlDescription – Describes the WSDL associated with a registered Oracle Service Bus service.

interface ServiceTransportSender – Sends outbound messages to a registered service associated with a transport endpoint. `TransportProvider.sendMessageAsync()` gets an instance of ServiceTransportSender (which extends TransportSender) from which the provider can retrieve the payload and metadata for outbound requests.

interface CredentialCallback – Transport providers get an instance of this callback interface from Oracle Service Bus. The transport provider can call its methods to fetch a credential used for outbound authentication.

interface TransportEndPoint – A transport endpoint is an Oracle Service Bus entity/resource where service messages are originated or targeted.

41.4 Source and Transformer Classes and Interfaces

Following are descriptions of the base Source and Transformer interfaces, along with several concrete Sources provided with Oracle Service Bus and some supporting classes.

For more information, see [Section 38.8, "Designing for Message Content."](#)

41.4.1 Summary of Source and Transformer Interfaces

interface Source – Represents source content in some form. Sources may be transformed into other Sources through a Transformer instance. At minimum, a Source must natively support conversion to a byte-based stream via the two methods defined in this interface. Source may or may not take into account various TransformOptions (for example, character-set encoding) during serialization.

interface SingleUseSource – A marker interface indicating that a type of Source can only be consumed once. It also provides one helper method that can be used to determine if the Source is still "consumable" (valid).

If you create a Source class that implements the Source interface, Oracle Service Bus is free to call the `getInputStream()` method multiple times, each time retrieving the input stream from the beginning. If the Source class implements `SingleUseSource`, Oracle Service Bus calls `getInputStream()` only once; however, Oracle Service Bus buffers the entire message in memory in this case.

interface Transformer – Transforms one type of Source to another. The instance is responsible for indicating what types of sources it can convert between. Note that a transformer is required to support the full cross-product of transformations implied by the supported input and output sources. In other words, a transformer must support transforming any supported input source to any supported output source.

41.4.2 Summary of Source and Transformer Classes

class StreamSource – A byte-stream Source whose content comes from an `InputStream`. As a byte-stream source, the serialization methods do not heed any transformation options.

Note: Because this stream is backed by an `InputStream`, that means that this source is a single-use source. Both serialization methods pull from the same underlying `InputStream`, and once that content is consumed, it is gone. The push-based `writeTo()` method results in all data being consumed immediately, assuming no error occurs. The pull-based `getInputStream()` actually gives the underlying `InputStream` directly to the caller.

class ByteArraySource – A byte-stream Source whose content comes from a byte array. As a byte-stream source, the serialization methods do not heed any transformation options.

class StringSource – A Source that is backed by a single `String`. Serialization is simply a character-set encoded version of the character data.

class XmlObjectSource – Apache XBean Source content is represented as an Apache XBean. The XBean may be typed and so may be accompanied by a `SchemaType` object and an associated `ClassLoader`. However, both of these are entirely optional and the XBean can be untyped XML.

class DOMSource – A Source whose content comes from a DOM node. The referenced node may be a full-fledged `org.w3c.dom.Document`, but it may also be an internal node in a larger document.

class MFLSource – Represents MFL content. MFL data is essentially binary data that has some logical structure imposed on it by an MFL definition. CSV is a simple example of MFL data, but the structure can be arbitrarily complex. The logical/in-memory representation of the data is an XML document, but its serialized representation is the raw unstructured binary data.

class SAAJSource – A Source that is backed by a SAAJ SOAPMessage object. A SAAJSource is typically converted to and from MessageContextSource and MimeSource.

class MimeSource – A Source representing arbitrary content with headers. Essentially this is a Source that represents a MIME part. Headers must conform to RFC822 whereas the Source can be any type of source. The serialization format for this Source is a fully-compliant MIME package. This source is also aware of Content-Transfer-Encoding, and it will perform the proper encoding of the underlying content stream if the header is present. Note that this means that the Source provided to the constructor should be in raw form and not be already encoded.

class MessageContextSource – A Source that represents all message content. The Source for the message and attachments are left untyped to allow for deferred processing. Eventually, however, the attachments source will likely be converted into an object and the message source will likely be converted to a specific typed source such as an XmlObjectSource or a StringSource.

Note: The serialization format of a MessageContextSource is always a MIME multipart/related package, irrespective of the native serializations of the message and attachment sources. However, if this serialized object is needed more than once, it is best to transform the Source into a MimeSource.

class TransformOptions – Represents a set of transformation options. Instances of this class are used in conjunction with the Transformer class to influence how an input source is converted to an output source (for example, a change in character-set encoding from SHIFT_JIS to EUC-JP). This class is also used by the InputStream/OutputStream methods of the Source interface, since that is effectively also a transformation between the Source and the byte-level representation in the InputStream/OutputStream.

class JavaObjectSource – A JavaObjectSource represents the payload carried by Oracle Service Bus transports that provide a Java messaging type, such as the JMS transport. The objects that make up this payload are registered in the proxy service pipeline Java object repository by the binding layer, and their contents are visible in message context variables through `<ctx:java-content ref='jcid:xyz' xmlns:ctx="http://www.bea.com/wli/sb/context" />` XML elements. In this example, `ref` points to the unique ID of the object in the Java object repository.

class JavaXmlSource – A JavaXmlSource represents the payload carried by the services that supports Java objects as the arguments, such as the JEJB transport. JavaXmlSource is made up of an XML representation that defines the shape of the message body in the proxy service pipeline and a map containing Java objects against unique keys. In the XML representation, Java object arguments are substituted by `<ctx:java-content ref='jcid:xyz' xmlns:ctx="http://www.bea.com/wli/sb/context" />` elements, where the `ref` attribute equals a key in the JavaObjects map that indexes the

replaced Java object. The map contains the objects to be registered in pipeline Java object repository against the unique IDs in the XML representation.

41.5 Metadata and Header Representation for Request and Response Messages

This section lists classes and interfaces that deal with request and response message metadata representation. See also [Section 39.5, "Handling Messages,"](#) and [Section 38.8, "Designing for Message Content."](#)

This section includes these topics:

- [Section 41.5.1, "Runtime Representation of Message Contents"](#)
- [Section 41.5.2, "Interfaces"](#)

41.5.1 Runtime Representation of Message Contents

abstract class CoLocatedMessageContext – Needs to be extended by a transport provider that implements optimization for co-located outbound calls to go through a Java method invocation instead of the transport layer. For an example implementation, see the class

`com.bea.alsb.transports.sock.SocketCoLocatedMessageContext.java`, which is part of the Sample Socket Transport described in [Chapter 41, "Transport SDK Interfaces and Classes."](#) See also [Section 39.5.5, "Co-Located Calls."](#)

abstract class RequestHeaders – Represents a union of standard and user-defined headers in a given inbound or outbound request message. The set of standard headers is specific to each transport provider. This is an abstract class to be extended by each transport provider to implement its version of request headers.

abstract class RequestMetaData<T extends RequestHeaders> – Represents inbound or outbound request message metadata information (for example, headers, request character set encoding, and so on.) Transport providers provide an extension of this class that adds metadata information applicable to the transport provider. For example, HTTP transport provider adds `get/setQueryString()`, `get/setClientHost()` and other methods.

abstract class ResponseHeaders – Represents a union of standard and user-defined headers in a given inbound or outbound response message. The set of standard headers is specific to each transport provider. This is an abstract class to be extended by each transport provider to implement their version of response headers.

abstract class ResponseMetaData<T extends ResponseHeaders> – Represents inbound or outbound response message metadata information (such as headers, request character set encoding, and so on.) Transport providers provide an extension of this class that adds metadata information applicable to the transport provider. For example, HTTP transport provider adds `get/setHttpResponseCode()` and other methods.

41.5.2 Interfaces

interface TransportMessageContext – Most message-oriented middleware (MOM) products treat messages as lightweight entities that consist of a header and a payload. The header contains fields used for message routing and identification; the payload contains the application data being sent. In general, the transport-level message context consists of a message ID, RequestMetadata, request payload, ResponseMetadata, response payload and related properties.

interface InboundTransportMessageContext – Inbound Transport Message Context implements the message context abstraction for incoming messages.

interface OutboundTransportMessageContext – Outbound Transport Message Context implements the message context abstraction for outgoing messages.

interface ServiceTransportSender – Sends outbound messages to a registered service. The service is associated with a transport endpoint.

interface TransportSendListener – This is the callback object supplied to the outbound transport allowing it to signal to the system that response processing can proceed. This callback object should be invoked on a separate thread from the request message.

41.6 User Interface Configuration

This section includes these topics:

- [Section 41.6.1, "Overview"](#)
- [Section 41.6.2, "Summary of UI Interfaces"](#)
- [Section 41.6.3, "Summary of UI Classes"](#)

41.6.1 Overview

Because each transport provider can decide on a list of service endpoint specific configuration properties to persist, a flexible user interface is required that allows the user to enter provider-specific configuration properties for each new service endpoint. What follows is a set of classes and interfaces that allow each transport provider to expose its own properties for the user to enter as part of Oracle Service Bus service definition wizard in the Oracle Service Bus Console.

This section lists interfaces and classes used to develop the user interface for a new transport.

41.6.2 Summary of UI Interfaces

interface TransportUIBinding – Represents an object responsible for rendering provider-specific UI pages used during the service definition, summary, as well as validation of transport provider specific endpoint configurations.

interface CustomHelpProvider – Lets you provide context-sensitive help for functionality you add to the Oracle Service Bus console, such as custom transports. For implementation details, see [Section 39.12, "Creating Help for Custom Transports."](#)

41.6.3 Summary of UI Classes

class TransportUIContext – Supplies options for the transport provider specific user interface. It is passed by Oracle Service Bus Console to each transport provider.

class TransportUIGenericInfo – Holds transport specific UI information for the common transport page in the Oracle Service Bus Service Definition wizard.

class TransportUIFactory – Provides factory methods for creating a Transport Edit Field and different kinds of Transport UI objects associated with the field. Also provides some helper methods for accessing values in these objects.

class TransportEditField – Represents a single editable UI element in the provider-specific portion of Oracle Service Bus Console service registration wizard.

class TransportViewField – Represents a single read-only UI element in the provider-specific portion of the service summary page Oracle Service Bus Console service registration wizard.

class TransportUIError – Returns validation errors to the Oracle Service Bus Console.

Sample Socket Transport Provider

This chapter explains how to build and run the sample socket transport provider. This sample is installed along with Oracle Service Bus. The sample serves as an example implementation of a custom transport provider and the sample source code is available to you.

This chapter includes these topics:

- [Section 42.1, "Sample Socket Transport Provider Design"](#)
- [Section 42.2, "Sample Location and Directory Structure"](#)
- [Section 42.3, "Building and Deploying the Sample"](#)
- [Section 42.4, "Start and Test the Socket Server"](#)
- [Section 42.5, "Configuring the Socket Transport Sample"](#)
- [Section 42.6, "Testing the Socket Transport Provider"](#)

42.1 Sample Socket Transport Provider Design

The primary purpose of the sample socket transport provider is to serve as an example transport provider implementation. This publicly available sample demonstrates the implementation and configuration details of the Transport SDK.

This section includes these topics:

- [Section 42.1.1, "Concepts Illustrated by the Sample"](#)
- [Section 42.1.2, "Basic Architecture of the Sample"](#)
- [Section 42.1.3, "Configuration Properties"](#)

42.1.1 Concepts Illustrated by the Sample

The sample transport is designed to send and receive streamed data to and from a configured TCP socket in Oracle Service Bus. The sample transport is intended to illustrate the following Transport SDK concepts:

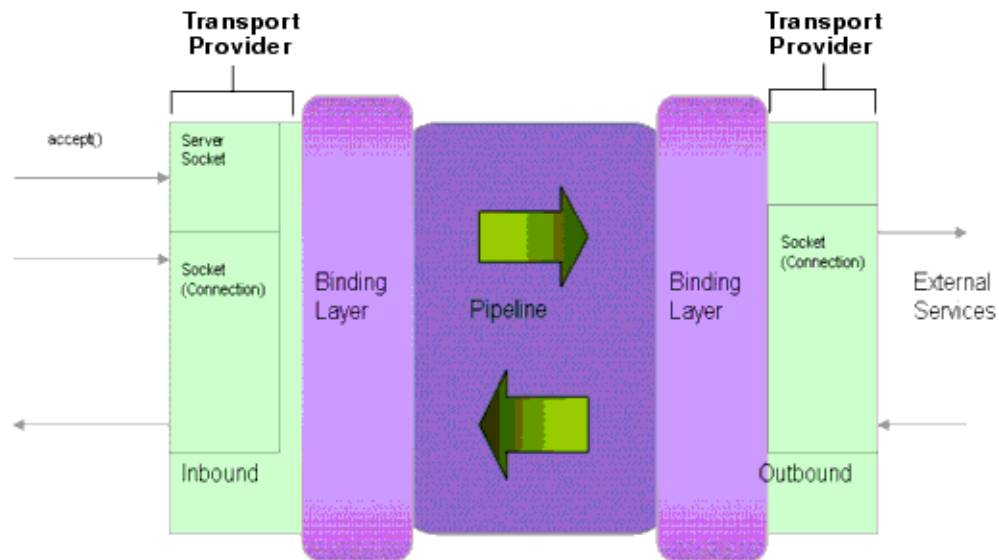
- Implementing the set of Transport SDK APIs that are required to build a custom transport.
- Performing transport endpoint validations, such as checking that no socket endpoint is listening on the configured address.
- Implementing several UI configuration options, including socket properties and message patterns.
- Implementing a one-way or synchronous request-response message pattern.

- Using POJOs (Plain Old Java Objects) for metadata and headers of endpoint requests and responses.
- Showing how streaming is used in the Oracle Service Bus pipeline.

42.1.2 Basic Architecture of the Sample

Figure 42-1 shows the basic architecture of the sample socket transport provider. Any client can connect to the server socket. Data is received at the server socket and passes through the pipeline. The response comes back through the outbound transport. The response is finally sent back to the inbound transport and back to the client.

Figure 42-1 Sample Socket Transport Architecture

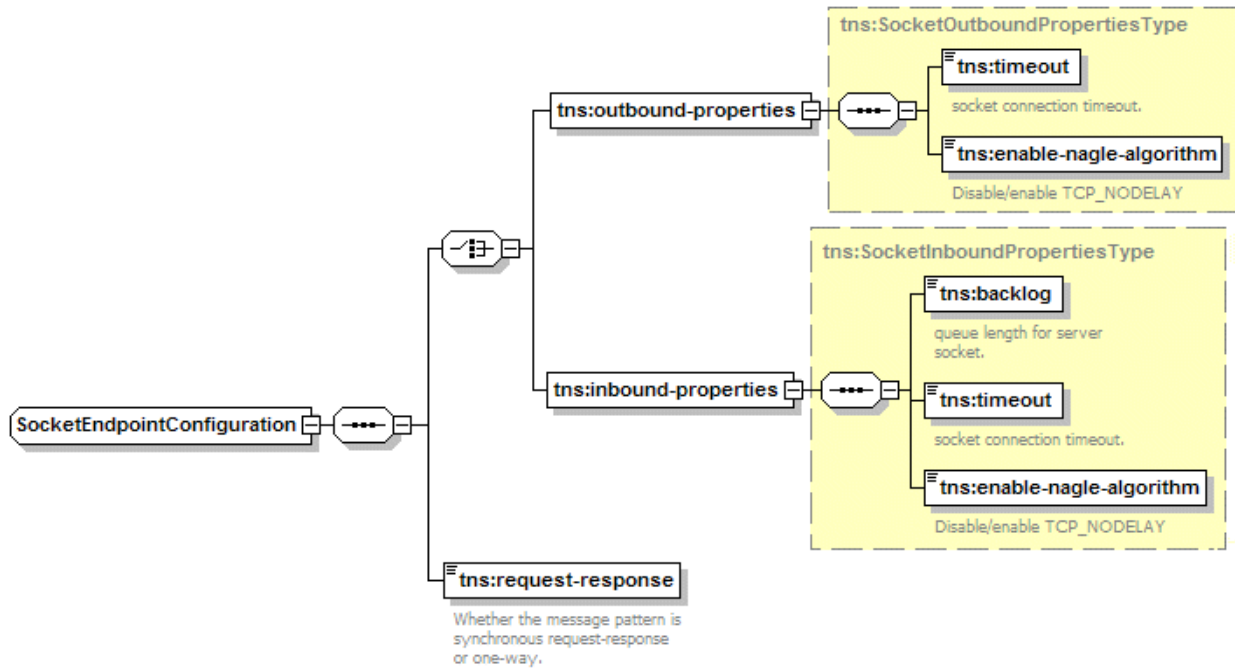


42.1.3 Configuration Properties

Figure 42-2 illustrates the configuration properties for the transport endpoint. These properties are configured in the schema file: `SocketTransport.xsd`. See [Section 42.2, "Sample Location and Directory Structure"](#) for information on the location of this file. This file allows you to extend the basic set of properties defined in the common schema provided with the SDK. Refer to the `SocketTransport.xsd` file for information on each of the properties.

Tip: See also [Section 39.3.4, "4. Define Transport-Specific Artifacts,"](#) for more information on these configuration properties.

Figure 42–2 SocketEndpointConfiguration Properties



Also in the SocketTransport.xsd file are the request/response header and metadata properties, as illustrated in Figure 42–3. Refer to the SocketTransport.xsd file for more information on these properties.

Figure 42–3 Request/Response Header and Metadata Configurations



42.2 Sample Location and Directory Structure

The sample socket transport provider is installed with Oracle Service Bus and is located in the following directory: `OSB_ORACLE_HOME/samples/servicebus/sample-transport`.

This section briefly describes some of the key folders in the sample project. You can use this directory structure as a model for developing your custom transport provider.

Table 42–1 lists and briefly describes key sample-transport directories.

Table 42–1 Key Sample Transport Provider Directories

Directory	Description
build	Created when you build the sample socket transport. Contains the built and packaged transport for use in Oracle Service Bus.
eclipse	Contains the plugin.xml file needed to add the sample transport to the Eclipse environment.
l10n	Contains Internationalization files: SocketTransportMessages.xml – Configuration file for text messages which are displayed on the Oracle Service Bus Console. SocketTransportTextMessages.xml
META-INF	Contains application deployment descriptor files: application.xml – J2EE application descriptor file weblogic-application.xml – WebLogic application descriptor file
resources	SocketConfig.xml – Socket transport provider configuration that is used by the Transport SDK. Sample help files for the transport.
schemas	Contains the relevant schemas defined for this transport: SocketTransport.xsd – Describes Socket Endpoint Request/Response Metadata/headers
src	Source tree of the sample transport
test	(not shown) Test files directory: src – Source tree for test server and client
webapp	Contains the deployment descriptors required for the sample transport help Web application.

The following Ant build files are also located in the `sample-transport` directory:

- `build.properties` – Properties file for Ant.
- `build.xml` – An Ant build file with different targets for compile, build, stage, and deploy.

42.3 Building and Deploying the Sample

This section explains how to build and deploy the sample transport provider.

42.3.1 Setting Up the Environment

Follow these steps to set the environment for building the sample.

1. Create a new domain or use one of the preconfigured domains that are installed with Oracle Service Bus.
2. Set the domain environment by running the following script:

```
DOMAIN_HOME/bin/setDomainEnv.cmd (setDomainEnv.sh on a UNIX system)
```

42.3.2 Building the Transport

To build the socket transport, do the following:

1. In a command window, go to the sample home directory:

`OSB_ORACLE_HOME/samples/servicebus/sample-transport`

2. Execute the following command: `ant build`. This command compiles the source files in `OSB_ORACLE_HOME/samples/servicebus/sample-transport/build`.
3. After a successful build, execute the following command: `ant stage`. This command does the following:
 - Copies `sock_transport.ear` `sock_transport.jar` to `OSB_ORACLE_HOME/lib/transports`.
 - Creates `OSB_ORACLE_HOME/eclipse/plugins/com.bea.alsb.transports.socket_version` to register the socket sample as a plug-in to Eclipse.

42.3.3 Deploying the Sample Transport Provider

To deploy the sample transport provider on a server, do the following:

1. Set the following variables in `sample-transport/build.properties`:

`wls.hostname`

`wls.port`

`wls.username`

`wls.password`

`wls.server.name`

2. Deploy the transport provider on the server by running the following command:

`ant deploy`

42.4 Start and Test the Socket Server

The sample project includes a simple socket server and a client to test the server. You can use this socket server to test the socket transport provider.

This section includes the following topics:

- [Section 42.4.1, "Start the Socket Server"](#)
- [Section 42.4.2, "Test the Socket Transport"](#)

42.4.1 Start the Socket Server

Run the following command to start the external service, which is a server socket that listens on a specified port and receives/sends the messages.

```
java -classpath .\test\build\test-client.jar -Dfile-encoding=utf-8
-Drequest-encoding=utf-8 com.bea.alsb.transports.sample.test.TestServer <port>
<message-file-location>
```

where:

- `port` – The port number at which `ServerSocket` is listening, which is the port number in the business service.
- `message-file-location` – (optional) The location of the message-file which will be sent as a response to the business service.

- `file-encoding` – A system property that is the encoding of the file. (default = `utf-8`)
- `request-encoding` – The encoding of the request that is sent by the socket business service. (default = `utf-8`)

42.4.2 Test the Socket Transport

Run the following command to start the service, which is a client to a configured socket proxy-service. It sends a message and receives the response from Oracle Service Bus.

```
java -classpath .\test\build\test-client.jar -Dfile-encoding=utf-8  
-Dresponse-encoding=utf-8 com.bea.alsb.transports.sample.test.TestClient  
<host-name> <port> <thread-ct> <message-file-location>
```

where:

- `host-name` – The host name of the Oracle Service Bus server.
- `port` – The port number at which the proxy service is listening.
- `thread-ct` – The number of clients that can send a message to Oracle Service Bus.
- `message-file-location` – (optional) The location of the message file that will be sent as a response to the business service.
- `file-encoding` – An optional argument specifying the encoding of the file. (default = `utf-8`)
- `response-encoding` – The encoding of the response received from the socket proxy service. (default = `utf-8`)

42.5 Configuring the Socket Transport Sample

The sample consists of a test server and a test client. The client sends a message to the server. You configure Oracle Service Bus to receive and process the message.

This section describes these tasks:

- [Section 42.5.1, "Create a New Project"](#)
- [Section 42.5.2, "Create a Business Service"](#)
- [Section 42.5.3, "Create a Proxy Service"](#)
- [Section 42.5.4, "Edit the Pipeline"](#)

42.5.1 Create a New Project

To create a new project:

1. Start the Oracle Service Bus Console.
2. Open the Project Explorer.
3. In the Change Center, click **Edit**.
4. In the Projects panel, enter `SocketTest` in the **Enter New Project Name Field**.
5. Click **Add Project**. The new project appears in the project table.

42.5.2 Create a Business Service

Create a business service to talk to the server.

1. Click the SocketTest project name in the project table. The SocketTest panel appears.
2. From the Create Service menu, select **Business Service**. The General Configuration panel appears.
3. In the General Configuration panel, enter `SocketBS` in the **Service Name** field.
4. Be sure **Any XML Service** is selected in the Service Type list, and click **Next**.
5. From the Protocol menu, select **socket**, as shown in [Figure 39.5.3](#).
6. In the Endpoint URI field, enter: `tcp://localhost:7031`, and click **Add**.
7. Click **Next**.
8. In the next panel, accept the defaults by clicking **Next**.
9. After viewing the Summary panel, click **Save**.
10. In the Change Center, click **Activate**.

42.5.3 Create a Proxy Service

In this section, you create a proxy service.

1. From the Create Resource menu, select **Proxy Service**.
2. In the General Configuration panel, enter `SocketProxy` in the **Service Name** field.
3. Be sure that **Any XML Service** is selected in the Service Type list, and click **Next**.
4. From the Protocol menu, select **socket**.
5. In the **Endpoint URI** field, enter `tcp://7032`, and click **Next**.
6. In the next panel, accept the defaults and click **Next**.
7. After viewing the Summary panel, click **Save**.
8. In the Change Center, click **Activate**.
9. Click **Submit**.

42.5.4 Edit the Pipeline

Now that the business and proxy services are defined, you can edit the pipeline to route incoming messages to the business service.

To edit the pipeline:

1. In the Change Center, click **Create**.
2. In the Resources section, click the **View Message Flow** icon in the SocketProxy row, as shown in [Figure 42-4](#).

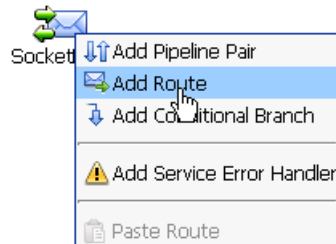
Figure 42-4 Selecting the Message Flow Icon

Name ▾	Resource Type ▲	Actions	Op
SocketBS	Business Service		a e
SocketProxy	Proxy Service		a e

Items 1-2 of 2

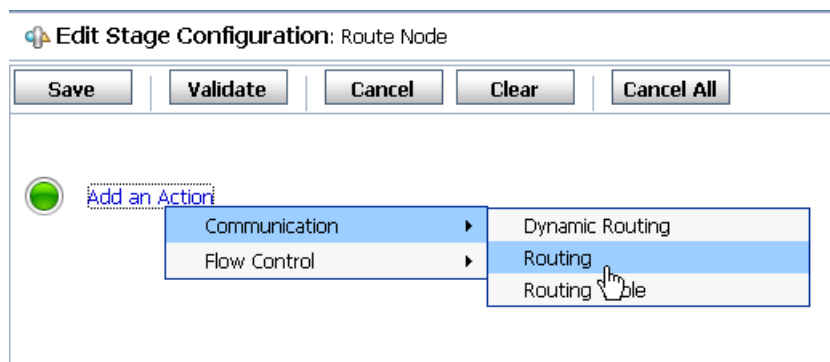
- In the Edit Message Flow window, click the **SocketProxy** icon and select **Add Route** from the menu, as shown in [Figure 42-5](#).

Figure 42-5 Editing the Message Flow



- Click the **RouteNode1** icon and select **Edit Route** from the menu.
- In the Edit Stage Configuration window, click **Add an Action**.
- In the Route Node window, click **Add an Action** and select **Communication > Routing** from the menu, as shown in [Figure 42-6](#).

Figure 42-6 Adding an Action



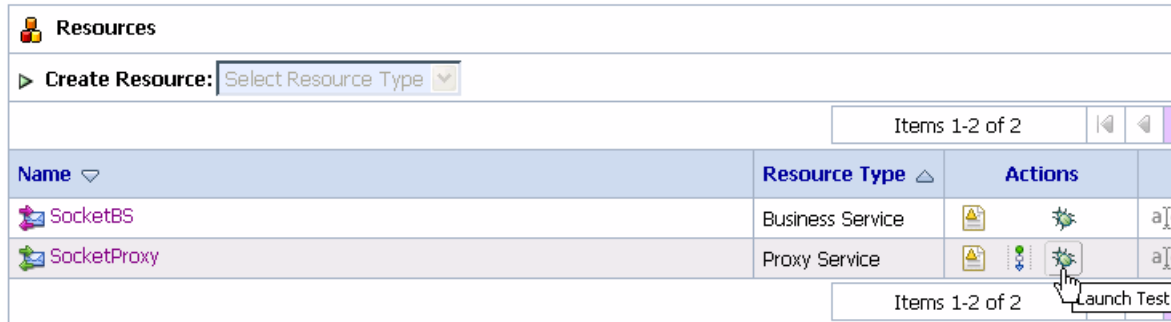
- In the next panel, select **<Service>**.
- In the Select Service window, select **SocketBS** from the list, and click **Submit**.
- In the Edit Stage Configuration window, click **Save**.
- Optionally, click the **RouteNode1** icon and change the name to **SocketBS**.
- Click **Save**.
- In the Change Center, click **Activate**, and then click **Submit**.

42.6 Testing the Socket Transport Provider

In this section you test the transport provider using Oracle Service Bus Console.

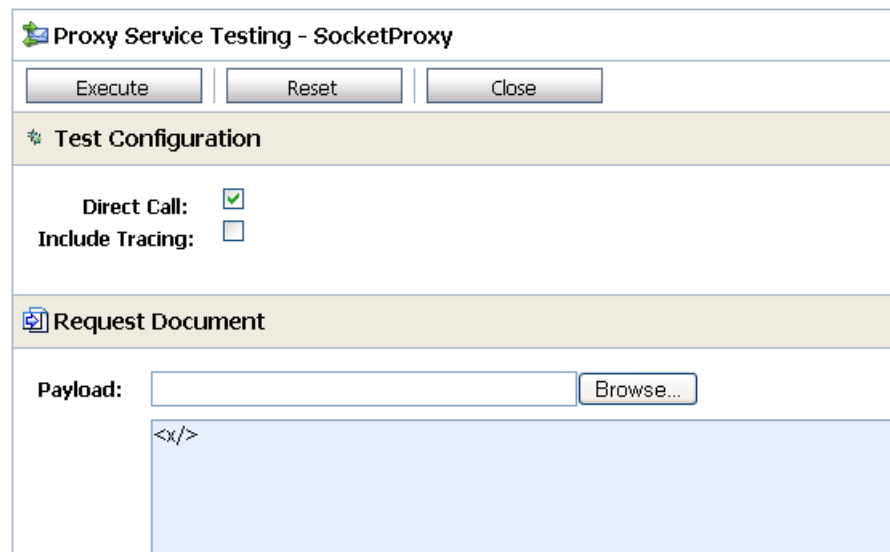
1. Start the test server, as explained previously in [Section 42.4.1, "Start the Socket Server."](#)
2. In the Project Explorer, click **SocketTest**.
3. In the SocketProxy row of the Resources table, click the **Launch Test Console** icon, as shown in [Figure 42-7](#).

Figure 42-7 Starting the Test Console



4. In the Test Console, enter any valid XML stanza in the text area, or use the **Browse** button to select a valid XML file on the local system. For example, in [Figure 42-8](#), a simple XML expression `<x/>` is entered in the text area.

Figure 42-8 Test Console



5. Click **Execute**. If the test is successful, information similar that shown in [Figure 42-9](#) appears in the Test Console. In addition, the XML text input into the Test Console is echoed in the server console.

Figure 42–9 Successful Test

The screenshot displays the 'Proxy Service Testing - SocketProxy' window. At the top, there are 'Back' and 'Close' buttons. Below this, there are sections for 'Request Document' and 'Response Document', both of which are currently empty. The 'Response Metadata' section shows the following XML content:

```
<project name="sock-transport" default="build-jar" basedir="."/>
```

The 'Invocation Trace' section shows two steps: '(receiving request)' and '(echoing request)'. At the bottom of the window, there are 'Back' and 'Close' buttons.

6. Close the Test Console.

Deploying a Transport Provider

This chapter explains how to package and deploy a custom transport provider and includes these topics:

- [Section 43.1, "Packaging the Transport Provider"](#)
- [Section 43.2, "Deploying the Transport Provider"](#)
- [Section 43.3, "Undeploying a Transport Provider"](#)
- [Section 43.4, "Deploying to a Cluster"](#)

43.1 Packaging the Transport Provider

You must package your custom transport provider as a self-contained EAR file. You can then deploy the EAR with the Oracle Service Bus Kernel EAR and other Oracle Service Bus related applications.

Tip: The sample socket transport provider example illustrates how a transport provider is organized and deployed. See [Section 42, "Sample Socket Transport Provider,"](#) for more information.

Each transport provider consists of two distinct parts:

- **Configuration** – The configuration part of a transport provider is used by Oracle Service Bus Console to register endpoints with the transport provider. This configuration behavior is provided by the implementation of the UI interfaces. [Section 41.6, "User Interface Configuration."](#)
- **Runtime** – The runtime part of a transport provider implements the business logic of sending and receiving messages.

Tip: A best practice is to package the transport provider so that the configuration and runtime parts are placed in separate deployment units. This practice makes cluster deployment simpler. See [Section 43.4, "Deploying to a Cluster"](#) for more information. See also [Section 38.4, "Transport Provider Components."](#)

43.2 Deploying the Transport Provider

This section discusses how to deploy a transport provider.

Tip: For more information on deploying applications to Oracle Service Bus, see the *Oracle Fusion Middleware Deployment Guide for Oracle Service Bus*.

After you create a deployable EAR file for your transport provider, you need to deploy it to the Oracle Service Bus domain. You can deploy the EAR by whatever method you prefer:

- Programmatically (using WebLogic Deployment Manager JSR-88 API)
- Using the Oracle WebLogic Server Administration Console
- Adding an entry similar to [Example 43-1](#) to the Oracle Service Bus domain `config.xml` file

Example 43-1 Application Deployment Entry

```
<app-deployment>
  <name>My Transport Provider</name>
  <target>AdminServer, myCluster</target>
  <module-type>ear</module-type>
  <source-path>${USER_INSTALL_DIR}/servicebus/lib/mytransport.ear</source-path>
  <deployment-order>1234</deployment-order>
</app-deployment>
```

Note: The deployment order of your transport provider EAR file should be high enough so that the entire Oracle Service Bus Kernel EAR is deployed before the transport provider.

43.2.1 Transport Registration

On server restart, you want to ensure that your deployed transport can immediately begin to handle service requests. To ensure immediate transport availability, extend the `weblogic.application.ApplicationLifecycleListener` class and use the `preStart()` method to register your transport using `TransportManager.registerProvider()`.

The sample socket transport has an `ApplicationListener` class that you can use for reference, located at `OSB_ORACLE_HOME/samples/servicebus/socket-transport/src/com/bea/alsb/transports/sock`.

When extending `ApplicationLifecycleListener`, be sure to register your extending class in `META-INF/weblogic-application.xml`. The sample socket transport provides the following entry for its `ApplicationListener` class in `OSB_ORACLE_HOME/samples/servicebus/sample-transport/META-INF/weblogic-application.xml`:

```
<weblogic-application>
  <listener>
    <!-- This class gives callbacks for the deployment lifecycle and socket
         transport is registered with ALSB whenever the application is started.
    -->
    <listener-class>com.bea.alsb.transports.sock.ApplicationListener
  </listener-class>
</listener>
</weblogic-application>
```

43.3 Undeploying a Transport Provider

Once a transport provider has been registered with Oracle Service Bus, the undeployment or unregistration of the transport provider is not supported.

43.4 Deploying to a Cluster

Your transport provider must be deployed on all the servers/clusters where Oracle Service Bus is deployed. This means that if Oracle Service Bus is deployed only on the admin server (which it always is), you must deploy the transport provider on the admin server; if Oracle Service Bus is deployed in an admin + managed server topology, you must deploy the transport provider on the admin server and that particular managed server; and if Oracle Service Bus is deployed in a cluster, you must deploy your transport provider on the admin server and the cluster. Note that Oracle Service Bus is always deployed on the admin server regardless of the domain topology.

The application code inside your transport provider EAR file needs to be aware dynamically of where the transport is being deployed (such as the administration server or a managed server) and exhibit only configuration behavior on the administration server and only run-time behavior on the managed server.

For example, in the initialization pseudo code in `some_transport.ear`, you can use this logic to decide whether or not to activate the configuration or runtime portion of the provider:

```
protected SomeTransportProvider() throws TransportException {
    . . . some other initialization code . . .
    if (!isRuntimeEnabled)
        _engine = new RuntimeEngine(. . .);
}
```

In this case, creating an instance of the `RuntimeEngine` class is run-time behavior and only needs to happen on a managed node in a multi-server domain or on the administration node in a single server domain.

Furthermore, as mentioned previously, in a cluster environment, `TransportProvider.createEndPoint()` and `deleteEndPoint()` are called on an administration server as well as managed servers in the cluster (with the exception of WLS HTTP router/front-end host). Some transport providers can choose not to do anything other than registering the fact that there is an endpoint with the given configuration, such as HTTP. In general the transport provider needs to examine whether `createEndPoint()` or `deleteEndPoint()` is called on the administration or managed server to decide the appropriate behavior.

Part VII

Security

This guide describes how to secure Oracle Service Bus and the messages it handles. Chapters include:

- [Chapter 44, "Introduction"](#)
- [Chapter 45, "Understanding Oracle Service Bus Security"](#)
- [Chapter 46, "Oracle Service Bus Security FAQ"](#)
- [Chapter 47, "Configuring Administrative Security"](#)
- [Chapter 48, "Securing Oracle Service Bus in a Production Environment"](#)
- [Chapter 49, "Configuring Transport-Level Security"](#)
- [Chapter 50, "Securing Oracle Service Bus with Oracle Web Services Manager"](#)
- [Chapter 51, "Using WS-Policy in Oracle Service Bus Proxy and Business Services"](#)
- [Chapter 52, "Configuring Message-Level Security for Web Services"](#)
- [Chapter 53, "Using SAML for Authentication"](#)
- [Chapter 54, "Configuring Custom Authentication"](#)
- [Chapter 55, "Message-Level Security with .Net 2.0"](#)

This document describes how to use standard technologies such as SSL and Web Services Security along with Oracle proprietary technologies to ensure that only authorized users can access resources in an Oracle Service Bus domain.

44.1 Document Audience

This document is intended for the following audiences:

- **Application Architects**—Architects who, in addition to setting security goals and designing the overall security architecture for their organizations, evaluate Oracle Service Bus security features and determine how to best implement them. Application Architects have in-depth knowledge of Java programming, Java security, and network security, as well as knowledge of security systems and leading-edge, security technologies and tools.
- **Security Developers**—Developers who focus on defining the system architecture and infrastructure for security products that integrate into Oracle Service Bus and on developing custom security providers for use with Oracle Service Bus. They work with Application Architects to ensure that the security architecture is implemented according to design and that no security holes are introduced, and work with Server Administrators to ensure that security is properly configured. Security Developers have a solid understanding of security concepts, including authentication, authorization, auditing (AAA), in-depth knowledge of Java (including Java Management eXtensions (JMX), and working knowledge of Oracle WebLogic Server, Oracle Service Bus, and security provider functionality.
- **Application Developers**—Developers who are Java programmers that focus on developing client applications, adding security to Web applications and Enterprise JavaBeans (EJBs), and working with other engineering, quality assurance (QA), and database teams to implement security features. Application Developers have in-depth/working knowledge of Java (including J2EE components such as servlets/JSPs and JSEE) and Java security.
- **Server Administrators**—Administrators work closely with Application Architects to design a security scheme for the server and the applications running on the server, to identify potential security risks, and to propose configurations that prevent security problems. Related responsibilities may include maintaining critical production systems, configuring and managing security realms, implementing authentication and authorization schemes for server and application resources, upgrading security features, and maintaining security provider databases. Server Administrators have in-depth knowledge of the Java security architecture, including Web services, Web application and EJB security, Public Key security, SSL, and Security Assertion Markup Language (SAML).

- **Application Administrators**—Administrators who work with Server Administrators to implement and maintain security configurations and authentication and authorization schemes, and to set up and maintain access to deployed application resources in defined security realms. Application Administrators have general knowledge of security concepts and the Java Security architecture. They understand Java, XML, deployment descriptors, and can identify security events in server and audit logs.

44.2 Related Information

Oracle Service Bus uses the WebLogic security framework as building blocks for higher level security services, including authentication, identity assertion, authorization, role mapping, auditing, and credential mapping. In addition to this document, the following documents provide information about the WebLogic Security Service:

- *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*—This document summarizes the features of the WebLogic Security Service and presents an overview of the architecture and capabilities of the WebLogic Security Service. It is the starting point for understanding the WebLogic Security Service.
- *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server*—This document highlights essential security measures for you to consider before you deploy Oracle WebLogic Server into a production environment.
- *Oracle Fusion Middleware Securing Oracle WebLogic Server*—This document explains how to configure security for Oracle WebLogic Server and how to use Compatibility security.
- *Oracle Fusion Middleware Securing Resources Using Roles and Policies for Oracle WebLogic Server*—This document introduces the various types of WebLogic resources, and provides information that allows you to secure these resources using Oracle WebLogic Server.

Understanding Oracle Service Bus Security

Oracle Service Bus supports open industry standards for ensuring the integrity and privacy of communications and to ensure that only authorized users can access resources in an Oracle Service Bus domain. It uses the underlying WebLogic security framework as building blocks for its security services.

The WebLogic security framework divides the work of securing a domain into several components (providers), such as authentication, authorization, credential mapping, and auditing. You configure only those providers that you need for a given Oracle Service Bus domain.

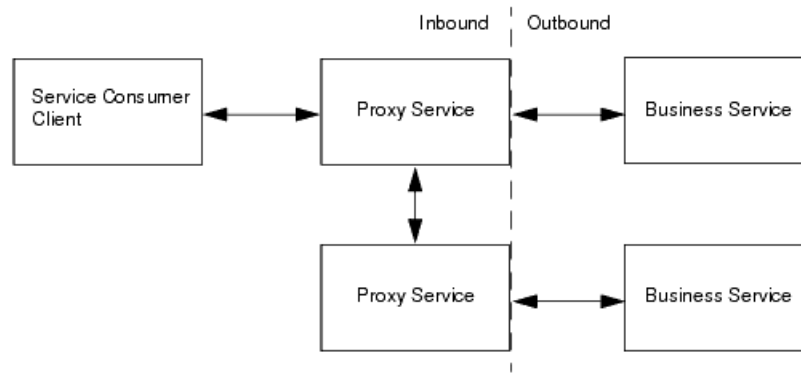
The following sections introduce the Oracle Service Bus security model and its features:

- [Section 45.1, "Inbound Security"](#)
- [Section 45.2, "Outbound Security"](#)
- [Section 45.3, "Options for Identity Propagation"](#)
- [Section 45.4, "Administrative Security"](#)
- [Section 45.7, "Configuring the Oracle WebLogic Security Framework: Main Steps"](#)
- [Section 45.8, "Context Properties Are Passed to Security Providers"](#)
- [Section 45.9, "Using Security Providers"](#)

45.1 Inbound Security

Inbound security ensures that Oracle Service Bus proxy services handle only the requests that come from authorized clients. (By default, any anonymous or authenticated user can connect to a proxy service.) It can also ensure that no unauthorized user has viewed or modified the data as it was sent from the client.

Proxy services can have two types of clients: service consumers and other proxy services. [Figure 45-1](#) illustrates that communication between proxy services and their clients is secured by inbound security, while communication between proxy services and business services is secured by outbound security.

Figure 45–1 Inbound and Outbound Security

You set up inbound security when you create proxy services and you can modify it as your needs change. For outward-facing proxy services (which receive requests from service consumers), consider setting up strict security requirements such as two-way SSL over HTTPS. For proxy services that are guaranteed to receive requests only from other Oracle Service Bus proxy services, you can use less secure protocols.

If a proxy service uses public key infrastructure (PKI) technology for digital signatures, encryption, or SSL authentication, create a **service key provider** to provide private keys paired with certificates. For more information, see "Service Key Providers" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

For each proxy service, you can configure the following inbound security checks:

- **Transport-level security** applies security checks as part of establishing a connection between a client and a proxy service. The security requirements that you can impose through transport-level security depend on the protocol that you configure the proxy service to use.

For example, for proxy services that communicate over the HTTP protocol, you can require that all clients authenticate against a database of users that you create in the Security Configuration module of the Oracle Service Bus Console. You then create an access control policy that specifies conditions under which authenticated users are authorized to access the proxy service.

Oracle Service Bus also supports client-specified custom authentication tokens for inbound transport-level requests.

For information about configuring transport-level security for each supported protocol, see [Chapter 49, "Configuring Transport-Level Security."](#)

- **Custom Authentication for message-level security.** Oracle Service Bus supports client-specified custom authentication credentials for inbound transport- and message-level requests. The custom authentication credentials can be in the form of a custom token, or a username and password.

For information on configuring custom authentication transport- and message-level security, see [Chapter 54, "Configuring Custom Authentication."](#)

- **Message-level security** (for proxy services that are Web Services) is part of the WS-Security specification. It applies security checks before processing a SOAP message or specific parts of a SOAP message.

Part of the configuration for message-level security can be embedded in the WSDL document and WS-Policy document that are associated with the Web service. These documents specify whether SOAP messages must be digitally signed and

encrypted and which Web service operations can be invoked only by authorized users.

There is an alternative way to bind WS-Policy to services. The WS-Policy console page allows you to bind policies to the service as a whole, to individual operations in the service, or to the request message or response message of individual operations.

If a proxy service or business service uses a WS-Policy statement to secure access to one or more of its operations, and if you have configured the service as an active intermediary (as opposed to a pass-through service), you use the Oracle Service Bus Console to create a message-level access control policy. The policy specifies conditions under which users, groups, or security roles are authorized to invoke the protected operations.

For more information about configuring message-level security, see [Chapter 52, "Configuring Message-Level Security for Web Services."](#)

45.2 Outbound Security

Outbound security secures communication between a proxy service and a business service. Most of the tasks that you complete for outbound security are for configuring proxy services to comply with the transport-level or message-level security requirements that business services specify.

For example, if a business service requires user name and password tokens, you create a service account, which either directly contains the user name and password, passes along the user name and password that was contained in the inbound request, or provides a user name and password that depend on the user name that was contained in the inbound request. For more information, see [Section 2.1.16, "Creating Service Account Resources."](#)

If a business service requires the use of PKI technology for digital signatures, or SSL authentication, you create a service key provider, which provides private keys paired with certificates. For more information, see "Service Key Providers" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

45.3 Options for Identity Propagation

A key group of decisions that you must make when designing security for Oracle Service Bus is how to handle (propagate) the identities that clients provide. You can configure Oracle Service Bus to do any of the following:

- Authenticate the credentials that clients provide
- Perform authorization checks
- Pass client credentials to business services unchanged
- Map client credentials to a different set of credentials that a business service can authenticate and authorize
- Bridge between security technologies

[Table 45–1](#) describes the decisions that affect how Oracle Service Bus propagates client identities to business services.

Table 45–1 Options for Identity Propagation

Decision	Description
Which type of credentials do you require clients to provide?	<p>For transport-level security, Oracle Service Bus adapts to your existing security requirements. Clients of Oracle Service Bus can supply user name and password tokens, SSL certificates, or any other type of custom authentication token that is supported by an Identity Assertion provider that you configure.</p> <p>For message-level security, Oracle Service Bus supports the Username Token, X.509 Token, any other type of custom authentication token that is supported by an Authentication or Identity Assertion provider that you configure, and SAML Token profiles (see Section 45.9, "Using Security Providers").</p> <p>If you are establishing security requirements for a new business service that uses Web Services Security, Oracle recommends that you require clients to provide SAML tokens. SAML is the emerging standard for propagating user identities within Web services. See Chapter 53, "Using SAML for Authentication."</p>
Do you require Oracle Service Bus to authenticate clients or to simply pass the client-supplied credentials to business services for authentication?	<p>When you require clients to authenticate with Oracle Service Bus, you add an additional layer of security. In general, the more security layers you add, the more secure you make a domain.</p> <p>To enable Oracle Service Bus to authenticate users, you must create user accounts in the Oracle Service Bus Console. If your set of users is very large, you must consider whether maintaining a large database of user accounts in the Oracle Service Bus Console is worth the effort.</p>
If Oracle Service Bus authenticates clients that provide X.509 tokens or SAML tokens, which Oracle Service Bus user maps to the tokens?	<p>Oracle recommends that you require clients to authenticate with Oracle Service Bus and that you modify the default access-control policies to allow (authorize) only specific, authenticated users access to your proxy services.</p> <p>To authenticate and authorize clients who supply X.509 certificates, SAML tokens, or other types of credentials other than user names and passwords, you must configure an identity assertion provider that maps the client's credential to an Oracle Service Bus user. Oracle Service Bus will use this user name to establish a security context for the client.</p>
If Oracle Service Bus authenticates clients that provide custom authentication tokens, which Oracle Service Bus user maps to the tokens?	<p>Oracle recommends that you require clients to authenticate with Oracle Service Bus and that you modify the default access-control policies to allow (authorize) only specific, authenticated users access to your proxy services.</p> <p>To authenticate and authorize clients who supply custom authentication tokens other than user names and passwords, you must configure an Identity Assertion provider that maps the client's credential to an Oracle Service Bus user. Oracle Service Bus will use this user name to establish a security context for the client.</p>

Table 45–1 (Cont.) Options for Identity Propagation

Decision	Description
<p>If Oracle Service Bus authenticates clients that provide user name and password tokens, decide whether you want to:</p> <ul style="list-style-type: none"> ■ Pass the client's user name and password to the business service ■ Map the client's user name to a new user name and password and pass the new credentials to the business service 	<p>If a custom username/password token is used, as described in Section 54.1, "What Are Custom Authentication Tokens?", then the username and password in the custom token can be used for outbound HTTP BASIC or outbound WS-Security Username Token authentication if a pass-through service account is used.</p> <p>If you pass the client-supplied user name and password to the business service, then clients are responsible for maintaining the credentials that the business service requires. If the business service changes its security requirements, then you must notify each client to make corresponding changes.</p> <p>If you expect a business service to change its requirements frequently, then consider mapping the credentials that clients supply to the credentials that the business service requires. The more clients for a business service, the more work will be required to maintain this credential mapping.</p>

[Table 45–2](#) describes all combinations of the requirements that you can impose for inbound and outbound transport-level security.

Table 45–2 Combinations of Transport-Level Security Requirements

This Inbound Requirement...	Can Be Used With This Outbound Requirement...	How to Configure
Client supplies user name and password in the HTTP header and Oracle Service Bus authenticates the client.	Pass the client's credentials in an HTTP header.	<ol style="list-style-type: none"> 1. Configure inbound HTTP security. See Section 49.2.1, "Configuring Inbound HTTP Security: Main Steps." Be sure to add the client's user name to the Oracle Service Bus Security Configuration module. 2. Configure outbound HTTP security. See Section 49.2.2, "Configuring Outbound HTTP Security: Main Steps." <p>Be sure to create a pass-through service account and attach the account to the business service.</p>
Same as previous requirement.	Map the client's credentials to a different Oracle Service Bus user and pass the new credentials in an HTTP header.	<ol style="list-style-type: none"> 1. Configure inbound HTTP security. See Section 49.2.1, "Configuring Inbound HTTP Security: Main Steps." Be sure to add the client's user name to the Oracle Service Bus Security Configuration module. 2. Configure outbound HTTP security. See Section 49.2.2, "Configuring Outbound HTTP Security: Main Steps." <p>Be sure to create a user-mapping service account and attach the account to the business service.</p>

Table 45–2 (Cont.) Combinations of Transport-Level Security Requirements

This Inbound Requirement...	Can Be Used With This Outbound Requirement...	How to Configure
Client supplies user name and password in the HTTP header and Oracle Service Bus does not authenticate the client.	Pass the client's credentials in an HTTP header.	<ol style="list-style-type: none"> 1. Configure inbound HTTP security. See Section 49.2.1, "Configuring Inbound HTTP Security: Main Steps." Be sure to configure the proxy service for HTTP, no authentication or HTTPS, one-way SSL, no authentication. 2. Configure outbound HTTP security. See Section 49.2.2, "Configuring Outbound HTTP Security: Main Steps." Be sure to configure the business service for HTTP BASIC authentication or HTTPS, one-way SSL, BASIC authentication. Also create a pass-through service account and attach the account to the business service.
Client supplies custom authentication token in the HTTP header. Oracle Service Bus authenticates the client.	Map the client's credentials to a different Oracle Service Bus user and pass the new credentials in an HTTP header.	<ol style="list-style-type: none"> 1. Configure inbound HTTP security. See Section 49.2.1, "Configuring Inbound HTTP Security: Main Steps." Be sure to add the client's user name to the Oracle Service Bus Security Configuration module. 2. Configure outbound HTTP security. See Section 49.2.2, "Configuring Outbound HTTP Security: Main Steps." Be sure to create a user-mapping service account and attach the account to the business service.
Any form of local authentication (HTTP or HTTPS BASIC, HTTPS CLIENT CERT with credential mapping)	Pass the client's credentials to an EJB over RMI. The EJB container authenticates the user.	Create a pass-through service account and attach the account to the business service. See Section 2.1.16, "Creating Service Account Resources."

Table 45–3 describes all combinations of the requirements that you can impose for inbound and outbound message-level security. In some cases, the inbound requirement for *transport-level* security affects the requirements that you can impose for outbound message-level security.

Table 45–3 Combinations of Message-Level Security Requirements

This Inbound Requirement...	Can Be Used With This Outbound Requirement...	How to Configure
Client supplies user name and password, or custom authentication token, in the HTTP header and Oracle Service Bus authenticates the client.	Pass the client's credentials in a SOAP header.	<ol style="list-style-type: none"> 1. Configure inbound HTTP security. See Section 49.2.1, "Configuring Inbound HTTP Security: Main Steps." Be sure to add the client's user name to the Oracle Service Bus Security Configuration module. 2. Create a pass-through service account and attach the account to the business service. See Section 2.1.16, "Creating Service Account Resources."
Same as previous requirement.	Map the client's credentials to a different Oracle Service Bus user and pass the new credentials in a SOAP header.	<ol style="list-style-type: none"> 1. Configure inbound HTTP security. See Section 49.2.1, "Configuring Inbound HTTP Security: Main Steps." Be sure to add the client's user name to the Oracle Service Bus Security Configuration module. 2. Create a user-mapping service account and attach the account to the business service. See Section 2.1.16, "Creating Service Account Resources."
Same as previous requirement.	Map the client credentials to a SAML token. Oracle Service Bus asserts the user identity.	<ol style="list-style-type: none"> 1. Configure inbound HTTP security. See Section 49.2.1, "Configuring Inbound HTTP Security: Main Steps." Be sure to add the client's user name to the Oracle Service Bus Security Configuration module. 2. Configure a SAML credential mapping provider. See Section 53.1, "Configuring SAML Credential Mapping: Main Steps."
Client supplies custom user name and password, or custom authentication token, in the message header or body and Oracle Service Bus authenticates the client.	Pass the client's credentials in a SOAP header.	<ol style="list-style-type: none"> 1. Configure an Authentication or Identity Assertion provider to handle the custom token or username and password. Be sure to add the client's user name to the Oracle Service Bus Security Configuration module. 2. Create a pass-through service account and attach the account to the business service. See Section 2.1.16, "Creating Service Account Resources."

Table 45–3 (Cont.) Combinations of Message-Level Security Requirements

This Inbound Requirement...	Can Be Used With This Outbound Requirement...	How to Configure
Same as previous requirement.	Map the client's credentials to a different Oracle Service Bus user and pass the new credentials in a SOAP header.	<ol style="list-style-type: none"> 1. Configure an Authentication or Identity Assertion provider to handle the custom token or username and password. Be sure to add the client's user name to the Oracle Service Bus Security Configuration module. 2. Create a user-mapping service account and attach the account to the business service. See Section 2.1.16, "Creating Service Account Resources."
Same as previous requirement.	Map the client credentials to a SAML token. Oracle Service Bus asserts the user identity.	<ol style="list-style-type: none"> 1. Configure an Authentication or Identity Assertion provider to handle the custom token or username and password. Be sure to add the client's user name to the Oracle Service Bus Security Configuration module. 2. Configure a SAML credential mapping provider. See Section 53.1, "Configuring SAML Credential Mapping: Main Steps."
Client supplies user name and password in the HTTP header and Oracle Service Bus does not authenticate the client.	Pass the client's credentials in a SOAP header.	<ol style="list-style-type: none"> 1. Configure inbound HTTP security. See Section 49.2.1, "Configuring Inbound HTTP Security: Main Steps." Be sure to configure the proxy service for HTTP, no authentication or HTTPS, one-way SSL, no authentication. 2. Configure outbound HTTP security. See Section 49.2.2, "Configuring Outbound HTTP Security: Main Steps." Be sure to configure the business service for HTTP BASIC authentication or HTTPS, one-way SSL, BASIC authentication. Also create a pass-through service account and attach the account to the business service.
Client supplies a certificate as part of HTTPS CLIENT-CERT authentication (two-way SSL) and Oracle Service Bus authenticates the client.	Map the client credentials to a SAML token. Oracle Service Bus asserts the user identity.	<ol style="list-style-type: none"> 1. Configure inbound HTTP security. See Section 49.2.1, "Configuring Inbound HTTP Security: Main Steps." 2. Configure a SAML credential mapping provider. See Section 53.1, "Configuring SAML Credential Mapping: Main Steps."

Table 45–3 (Cont.) Combinations of Message-Level Security Requirements

This Inbound Requirement...	Can Be Used With This Outbound Requirement...	How to Configure
An active intermediary proxy service enforces Web-Services Security with the User Name Token Profile.	Encode the credentials as a user name and password token in the SOAP message.	Create an active intermediary proxy service with a WS-Policy statement that requires passwords (not password digests). See Section 52.3.1, "Creating an Active Intermediary Proxy Service: Main Steps."
Same as previous requirement.	Encode the credentials as a SAML token in the SOAP message.	<ol style="list-style-type: none"> 1. Create an active intermediary proxy service with a WS-Policy statement that requires passwords. See Section 52.3.1, "Creating an Active Intermediary Proxy Service: Main Steps." 2. Configure a SAML credential mapping provider. See Section 53.1, "Configuring SAML Credential Mapping: Main Steps."
An active intermediary proxy service enforces Web-Services Security with the X.509 Token Profile.	Encode the credentials as a SAML token in the SOAP message.	<ol style="list-style-type: none"> 1. Create an active intermediary proxy service with a WS-Policy statement that requires digital signatures and optionally requires authentication with an X.509 token. See Section 52.3.1, "Creating an Active Intermediary Proxy Service: Main Steps." 2. Configure a SAML credential mapping provider. See Section 53.1, "Configuring SAML Credential Mapping: Main Steps."
An active intermediary proxy service enforces Web-Services Security with the SAML Token Profile.	Generate a new SAML token in the outbound SOAP message.	<ol style="list-style-type: none"> 1. Create an active intermediary proxy service with a WS-Policy statement that requires a SAML token. See Section 53.3, "Authenticating SAML Tokens in Proxy Service Requests." 2. Configure a SAML credential mapping provider. See Section 53.1, "Configuring SAML Credential Mapping: Main Steps."
A pass-through proxy service, which can pass user names and passwords, X.509 tokens, or SAML tokens.	A business service that uses either the User Name Token Profile, the X.509 Token Profile, or the SAML Token Profile.	<ol style="list-style-type: none"> 1. Create a pass through proxy service. See Section 52.3.1, "Creating an Active Intermediary Proxy Service: Main Steps." 2. Create a business service that enforces one of the token profiles. See Section 52.4, "Configuring Business Service Message-Level Security: Main Steps" or Section 53.2, "Configuring SAML Pass-Through Identity Propagation."

For inbound Tuxedo requests, you can configure any of the following security requirements:

- Encode the client's credentials in an outbound call to a Tuxedo service.
- Encode the client's credentials in an outbound SOAP message as either a user name token or a SAML token.
- Map the client's credentials to a different Oracle Service Bus user and pass the new credentials in an outbound HTTP header.
- Map the client's credentials to a different Oracle Service Bus user and pass the new credentials to an EJB over RMI. The EJB container authenticates the user.

For information about using Tuxedo with Oracle Service Bus, see [Chapter 35, "Tuxedo Transport."](#)

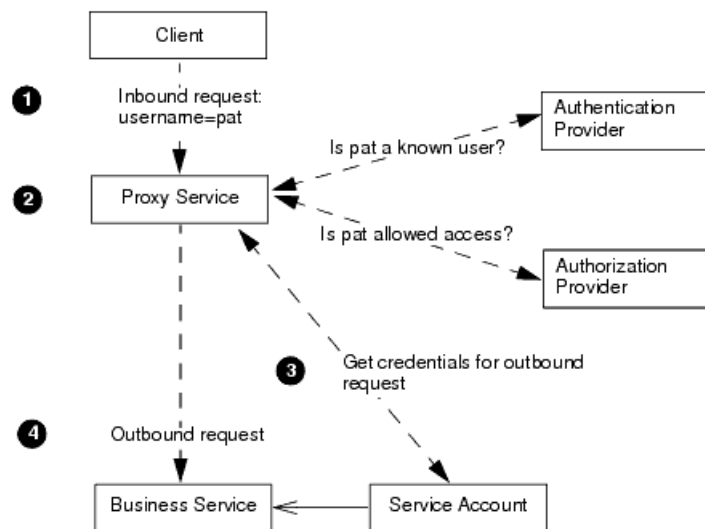
45.3.1 Example: Authentication with a User Name Token

[Figure 45–2](#) illustrates how user identities flow through Oracle Service Bus when you configure Oracle Service Bus as follows:

- Require clients to provide user names and passwords in their requests

You can require Web services clients to provide credentials at the transport level, the message level, or both. If you require clients to provide credentials at both levels, Oracle Service Bus uses the message-level credentials for identity propagation and credential mapping.
- Authenticate clients

Figure 45–2 How Service Accounts Are Used



The illustration begins with the inbound request and ends with the outbound request:

1. A client sends a request to a proxy service. The request contains the user name and password credentials.

Clients can send other types of tokens for authentication, such as an X.509 certificate or a custom authentication token. If a client sends an X.509 certificate token or a custom token, you must configure an identity assertion provider to map the identity in the token to an Oracle Service Bus security context.

2. The proxy service asks the domain's authentication provider if the user exists in the domain's authentication provider store.

If the user exists, the proxy service asks the domain's authorization provider to evaluate the access control policy that you have configured for the proxy service.

3. If the proxy service's access control policy allows the user access, the proxy service processes the message. As part of generating its outbound request to a business service, the proxy service asks the business service to supply the user name and password that the business service requires.

The business service asks its service account for the credentials. Depending on how the service account is configured, it does one of the following:

- Requires the proxy service to encode a specific (static) user name and password.
 - Requires the proxy service to pass along the user name and password that the client supplied.
 - Maps the user name that was returned from the authentication provider to some other (remote) user name, then requires the proxy service to encode the remote user name.
4. The proxy service sends its outbound request with the user name and password that was returned from the service account.

45.4 Administrative Security

To secure access to administrative functions, such as creating proxy services or business services, Oracle Service Bus provides four security roles with pre-defined access privileges:

- IntegrationAdmin
- IntegrationDeployer
- IntegrationMonitor
- IntegrationOperator

A security role is an identity that can be dynamically conferred upon a user or group at runtime. You cannot change the access privileges for these administrative security roles, but you can change the conditions under which a user or group is in one of the roles.

The Oracle Service Bus roles have permission to modify only Oracle Service Bus resources; they do not have permission to modify Oracle WebLogic Server or other resources on Oracle WebLogic Server. When assigning administrative users to roles, assign at least one user to the Oracle WebLogic Server Admin role. The Oracle WebLogic Server security roles are described in [Table 47-2](#).

For more information, see [Chapter 47, "Configuring Administrative Security."](#)

45.5 Access Control Policies

Access control determines who has access to the resources in Oracle Service Bus. An access control policy specifies conditions under which users, groups, or roles can access a proxy service. For example, you can create a policy that always allows users in the GoldCustomer role to access a proxy service and that allows users in the SilverCustomer role to access the proxy service only after 12pm on weeknights.

An access control policy is an association between a WebLogic resource and one or more users, groups, or security roles. A security policy protects the WebLogic resource against unauthorized access. Access control policies are boolean expressions assigned to specific resources. When there is an attempt to access the resource, the expression is evaluated. The expression consists of one or more conditions joined by boolean operators, such as a role (operator) and access time (8 am to 5 pm). For more information about access control policies, see "Security Fundamentals" in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*.

Oracle Service Bus relies on Oracle WebLogic Server security realms to protect its resources. Each security realm consists of a set of configured security providers, users, groups, security roles, and (access control) security policies. To access any resources belonging to a realm, a user must be assigned a security role defined in that realm, as described in [Section 47.1, "Administrative Security Roles and Privileges."](#) When a user attempts to access an Oracle Service Bus resource, Oracle WebLogic Server authenticates and authorizes the user by checking the security role assigned to the user in the relevant security realm and relevant security policy.

Note: Only a Oracle WebLogic Server administrator can define security policies or edit security roles in the Oracle Service Bus Console.

For all proxy services, you can create a transport-level policy, which applies a security check when a client attempts to establish a connection with the proxy service. Only requests from users who are listed in the transport-level policy are allowed to proceed.

For proxy services that are WS-Security active intermediaries, or that implement message-level custom authentication, you can also create a message-level policy. This type of policy applies a security check when a client attempts to invoke one of the secured operations. Only users who are listed in the message-level policy are allowed to invoke the operation.

The Oracle Service Bus Console contains a Security Configuration module for viewing and configuring users, groups, and security roles. Additionally, the Oracle Service Bus Console allows you to view and configure credentials.

45.5.1 Configuring Proxy Service Access Control

You can configure transport-level access control for all proxy services. You can also configure access control at the message-level for any WS-Security active intermediary proxy service, or for any proxy service that implements message-level custom authentication. To configure access control, you must assign an access control policy to the proxy service, either at the transport-level or message-level (or both).

The default transport-level and message-level access control policy for all proxy services is to allow access to all requests. You must assign an access control policy to the proxy service to protect it.

You configure transport-level and message-level access control policies in the Oracle Service Bus Console, as described in "Editing Transport-Level Access Policies" and "Editing Message-Level Access Policies."

45.5.2 Access Control Policy Management

Access control policies are persisted in authorization providers, and there is a reference to them in the Oracle Service Bus repository.

Access control policies are managed within an Oracle Service Bus design session and not outside the session, as was the case in releases prior to 3.0. Because the changes are made within a session, you can commit or discard the changes as with other resources.

Although ACLs can be managed from the Oracle Service Bus console, you can change policies outside Oracle Service Bus. However, changing policies outside of Oracle Service Bus can make the reference in Oracle Service Bus out-of-date and invalid.

Therefore, for consistent management, either completely manage ACLs outside of Oracle Service Bus sessions (using the authorization provider MBeans or third-party authorization provider tools) or completely manage them from within Oracle Service Bus sessions. Any combination of the two approaches can result in an inconsistent view of policies.

Oracle Service Bus manages access control policy only for proxy services. You must manage access control policy management for other server resources, such as JMS queues, JNDI entries, EJBs, applications, Oracle WebLogic Server instances, data sources, and so forth from the Oracle WebLogic Server console.

Note: When you clone a service, ACLs are also cloned in the session. If the user commits the session, ACLs on the service will be committed to the authorization provider. Therefore, when you clone a service you need to decide if you want the clone to have the same ACLs as the original. If you do not want this, then make sure to edit the ACLs of the clone.

45.5.2.1 Deleting a Proxy Service

Deleting a proxy service deletes all of the ACLs referenced by the proxy from the repository controlled by Oracle Service Bus, as well as from the appropriate authorization provider.

45.5.2.2 Deleting the Access Control Policy Assigned to a Proxy Service

You can also delete the access control policies assigned to a service without deleting the service. To do this:

1. Create a session.
2. From the **View a Proxy Service > Security tab**, use the edit Transport Access Control option to delete the policies.
3. Commit the session.

45.5.2.3 Moving or Renaming a Proxy Service

Renaming a proxy service correctly moves all of the policies. You need only rename or move the service in an Oracle Service Bus session.

45.5.2.4 Renaming a Proxy Service Operation

When an operation is renamed, the existing operation is transparently deleted and a new operation is created.

However, when an operation name is changed by changing the WSDL, Oracle Service Bus considers any policies for the old operation to be invalid, removes the reference from the Oracle Service Bus repository, and deletes the policies from the appropriate authorization provider.

In this case Oracle Service Bus does not know that the old operation is renamed to the new operation, and it does **not** add anything new for the new operation. That is, Oracle Service Bus considers that there are no policies for this new operation.

You need to add the appropriate policy to the new operation manually. You can do this in the same session as the rename of operation, after the rename is done.

45.6 Preserving Security Configuration During Import

As of this release of Oracle Service Bus, you can export or import Oracle Service Bus resources without losing any associated security configuration data.

Oracle Service Bus includes import check boxes that you can use to determine whether to preserve or overwrite the existing security configuration.

For example, assume that you want to configure your credentials in a staging area, export a project that contains these credentials, and then import the project in your production environment. When you export the project, the security configuration is included in the Oracle Service Bus configuration jar. When you then import the project on your target system, how the resources are treated depends on whether they already exist on the target system:

- New resources that exist only in the jar file always use the security configuration from the jar file.
- For resources that exist on the import target server as well as in the jar file, the new import check box allows you to decide whether to preserve the existing security configuration or to overwrite it with the configuration in the jar file.

The three import check boxes allow you to decide which, if any, aspects of the security configuration must be preserved during import:

- Preserve Security and Policy Configuration
- Preserve Credentials
- Preserve Access Control Policies

Note: These check boxes work the same way for Oracle Service Bus configuration files created for a project-level export and for an individual resource export.

These check boxes are described in more detail in the sections that follow.

45.6.1 Preserve Security and Policy Configuration Check Box

When the Preserve Security and Policy Configuration check box is set (the default), the following configuration parameters are preserved:

- Proxy service security and policy configuration:
 - A reference to the service key provider.
 - The set of WS-Policies that are bound directly to the service through the Policies tab.

Note: If the service is using WSDL-based policies, the policies are not preserved by this check box. This is because the WSDL might itself be updated and the service must reflect the WSDL.

The control also preserves the type of the WS-Policy Binding, either Custom (through the Policies tab) or WSDL-based.

- The state of the Process WS-Security Header check box.
- Message-level custom authentication configuration.
- Proxy service transport-specific security configuration:
 - For HTTP, the HTTPS flag and the authentication mode (anonymous, BASIC, client certificate, or custom token).
 - For JMS, the JMS and JNDI service accounts.
 - For email and FTP, the service account reference.
 - The SFTP authentication configuration.
- Business service security and policy configuration:
 - WS-Policy bindings
 - The Pass Caller's Subject setting.
 - A reference to the service account for outbound WS-Security.
- Business service transport-specific security configuration:
 - For HTTP, the authentication mode (anonymous, BASIC, or client certificate) and the service account reference.
 - For JMS, references to the JMS and JNDI service accounts.
 - For FTP, EJB, Tuxedo, and DSP, the service account reference.
 - The SFTP authentication configuration.

45.6.2 Preserve Credentials Check Box

When the Preserve Credentials check box is set (the default), the following credentials are preserved during the import process:

- PKI credentials in service key providers.

A PKI credential mapping provider maps Oracle Service Bus service key providers to key-pairs that can be used for digital signatures and encryption (for Web Services Security) and for outbound SSL authentication. For more information, see "Configuring a PKI Credential Mapping Provider" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.
- Username and passwords in service accounts.
- Username and password in SMTP server, JNDI provider, and UDDI registries.

45.6.3 Preserve Access Control Check Box

When the Preserve Access Control Policies check box is set (the default), all access control policies for the imported proxy services are preserved during the import process.

45.7 Configuring the Oracle WebLogic Security Framework: Main Steps

Many of the initial configuration tasks for Oracle Service Bus security require you to work in the Oracle WebLogic Server Administration Console to configure the

WebLogic security framework. After these initial tasks, you can complete most security tasks from the Oracle Service Bus Console.

To configure the WebLogic security framework for Oracle Service Bus:

1. If you plan to use SSL as part of transport-level security, do the following:
 - a. In the Oracle WebLogic Server Administration Console, configure identity and trust. See "Configuring Identity and Trust" and "Important Information Regarding Cross-Domain Security Support" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.
 - b. In the Oracle WebLogic Server Administration Console, configure SSL. See "Configuring SSL" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

Oracle recommends the following for your SSL configuration:

- If you configure two-way SSL, you must choose between two modes: *Client Certificate Requested But Not Enforced* or *Client Certificates Requested and Enforced*. Oracle recommends that whenever possible you choose *Client Certificate Requested and Enforced*. For more information, see "Secure Sockets Layer (SSL)" in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*.
 - In a production environment, make sure that Host Name Verification is enabled. See "Using Host Name Verification" in "Configuring SSL" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.
2. In the Oracle WebLogic Server Administration Console, configure authentication providers, which your proxy services use for inbound security.

[Table 45–4](#) describes the authentication providers that are commonly configured for Oracle Service Bus. For a description of all authentication providers that you can configure, see "Security Providers" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

Table 45–4 Authentication Providers

If You Require Clients to Provide...	Configure...
Simple user names and passwords	<p>The WebLogic Authentication provider and use the Oracle Service Bus Console to enter the user names and passwords of the clients that you want to allow access.</p> <p>Note: As described in Section 45.9.1, "Configuring Authentication Providers," Oracle recommends that you use the default WebLogic Authentication provider for all Oracle WebLogic Server and Oracle Service Bus administrative accounts.</p> <p>See "Adding Users" in the <i>Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus</i>.</p>
X.509 tokens for inbound HTTPS and two-way SSL authentication	<p>All of the following:</p> <ul style="list-style-type: none"> ■ The WebLogic Identity Assertion provider, which can validate X.509 tokens but does not by default. Make sure that you enable this provider to support X.509 tokens. In addition, enable this provider to use a user name mapper. See "Identity Assertion and Tokens" in <i>Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server</i>. ■ WebLogic CertPath Provider, which completes and validates certificate chains by using trusted Certificate Authority based checking.

Table 45–4 (Cont.) Authentication Providers

If You Require Clients to Provide...	Configure...
Custom authentication and username/password tokens for inbound HTTP and message-level authentication	An Identity Assertion provider, possibly user-written or from a third-party, that can validate the token type. Make sure that you enable this provider to support the token.
X.509 tokens for inbound Web Services Security X.509 Token Authentication	<p>If any of your proxy services or business services are Web services that use abstract WS-Policy statements, you must also configure the following:</p> <p>In the Web Service security configuration named <code>__SERVICE_BUS_INBOUND_WEB_SERVICE_SECURITY_MBEAN__</code> add the <code>UseX509ForIdentity</code> property and set it to <code>true</code>. See "Use X.509 Certificates to Establish Identity" in the <i>Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help</i>.</p>
SAML tokens	<p>All of the following:</p> <ul style="list-style-type: none"> ▪ WebLogic SAML Identity Assertion Provider V2, which authenticates users based on Security Assertion Markup Language 1.1 (SAML) assertions. ▪ WebLogic SAML Credential Mapping Provider V2, which maps Oracle Service Bus users to remote users.

3. If needed, in the Oracle WebLogic Server Administration Console, configure one or more Identity Assertion providers to handle the token types, such as X.509 or custom token types, for which you require support. For a description of all Identity Assertion providers that you can configure, see "Security Providers" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.
4. If you plan to create proxy services or business services that require WS-Security digital signatures on inbound requests, enable the Certificate Registry provider, which is a Certification Path provider that validates inbound certificates against a list of certificates that you register.

See "Configure Certification Path Providers" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*.

5. If you configure message-level security (in inbound requests or outbound requests) to require user name and password tokens, and if you want messages to provide a password digest instead of cleartext passwords, do the following:

- a. In the Oracle WebLogic Server Administration Console, find the two Web Service security configurations that Oracle Service Bus provides and set the value of the `UsePasswordDigest` property to `true`.

The Oracle Service Bus Web Service security configurations are named:

`__SERVICE_BUS_INBOUND_WEB_SERVICE_SECURITY_MBEAN__` and
`__SERVICE_BUS_OUTBOUND_WEB_SERVICE_SECURITY_MBEAN__`

For information on setting the values in Web Service security configurations, see "Use a Password Digest in SOAP Messages" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*.

- b. For each authentication provider that you configured, in the Oracle WebLogic Server Administration Console, select the Password Digest Enabled check box.

- c. For each identity assertion provider that you configured, in the Oracle WebLogic Server Administration Console set `wsse:PasswordDigest` as one of the active token types.
6. If you plan to create a service key provider (which passes key-certificate pairs in outbound requests), use the Oracle WebLogic Server Administration Console to configure a PKI credential mapping provider. In any Oracle WebLogic Server domain that hosts Oracle Service Bus, you can configure at most one PKI credential mapping provider.

A PKI credential mapping provider maps Oracle Service Bus service key providers to key-pairs that can be used for digital signatures and encryption (for Web Services Security) and for outbound SSL authentication. For more information, see "Configuring a PKI Credential Mapping Provider" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

You store the key-pairs that the PKI credential mapping provider uses in a keystore. You can store the PKI credential mappings in Oracle WebLogic Server's identity keystore or in a separate keystore. Configure each Oracle WebLogic Server instance to have access to its own copy of each keystore. All entries referred to by the PKI credential mapper must exist in all keystores (same entry with the same alias). For information about configuring keystores in Oracle WebLogic Server, see "Identity and Trust" in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*.

Note: When you create an Oracle Service Bus domain, by default the domain contains a user name/password credential mapping provider, which you can use if you need credential mapping for user names and passwords. In addition to this user name/password credential mapping provider, you can add one PKI credential mapping provider. An Oracle Service Bus domain can contain at most one user name/password credential mapping provider, one PKI credential mapping provider, and multiple SAML credential mapping providers.

7. If you want to enable security auditing, do the following:
 - a. In the Oracle WebLogic Server Administration Console, configure an auditing provider. See "Configuring the WebLogic Auditing Provider" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.
 - b. To enable auditing of events related to WS-Security, when you start each Oracle Service Bus server, include the following Java option in the server's startup command:

```
-Dcom.bea.wli.sb.security.AuditWebServiceSecurityErrors=true
```

Oracle Service Bus supports the auditing of security events but it does not support configuration auditing, which emits log messages and generates audit events when a user changes the configuration of any resource within a domain or invokes management operations on any resource within a domain. See "Enabling Configuration Auditing" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

8. If you have not already done so, in the Oracle WebLogic Server Administration Console, activate your changes. If you have made changes that require you to restart Oracle WebLogic Server, the Administration Console will indicate that a restart is required. If you see such a message, restart all Oracle WebLogic Server

instances that host Oracle Service Bus so your modifications to the security providers will be in effect for the remaining configuration steps.

45.8 Context Properties Are Passed to Security Providers

Context Properties provides a way (the `ContextHandler` interface) to pass additional information to the WebLogic Security Framework so that a security provider can obtain contextual information beyond what is provided by the arguments to a particular provider method. A `ContextHandler` is a high-performing WebLogic class that obtains additional context and container-specific information.

Oracle Service Bus makes use of the `ContextHandler` interface and passes several context properties to the security framework for transport-level and message-level authentication, transport-level and message-level access control, and credential mapping.

This section describes the situations in which Oracle Service Bus-specific context properties are used.

45.8.1 Context Properties for HTTP Transport-Level Authentication

When an HTTP proxy service is configured for authentication, the HTTP transport provider passes an Oracle Service Bus implementation of the Oracle WebLogic Server `ContextHandler`. There is no user configuration required for this feature.

The `ContextHandler` properties in [Table 45–5](#) are passed at runtime, under the following conditions:

- To Authentication providers, if the proxy is configured for HTTP BASIC authentication.
- To Identity Assertion providers, if the proxy is configured for CLIENT-CERT identity assertion.
- To Identity Assertion providers, if the proxy is configured for HTTP custom token identity assertion.

Table 45–5 *ContextHandler Properties for HTTP Transport Authentication*

Property Name	Type	Property Value
<code>com.bea.contextelement.alsb.service-info</code>	<code>com.bea.wli.sb.service.s.ServiceInfo</code>	An instance of <code>ServiceInfo</code> that contains information about the proxy service.
<code>com.bea.contextelement.alsb.transport.endpoint</code>	<code>com.bea.wli.sb.transports.TransportEndPoint</code>	This is the HTTP or HTTPS endpoint.
<code>com.bea.contextelement.alsb.transport.http.request</code>	<code>javax.servlet.http.HttpServletRequest</code>	This is the <code>HttpServletRequest</code> object.
<code>com.bea.contextelement.alsb.transport.http.response</code>	<code>javax.servlet.http.HttpServletResponse</code>	This is the <code>HttpServletResponse</code> object.

45.8.2 ContextHandler Properties for Access Control and Message-Level Custom Authentication

The ContextHandler properties shown in [Table 45–6](#) are passed at runtime, under the following conditions:

- To Authentication providers when performing message-level custom username/password authentication.
- To Identity Assertion providers when performing message-level custom token identity assertion.
- To Authorization providers when performing transport-level or message-level access control.

Table 45–6 ContextHandler Properties for Message-Level Custom Authentication and Access Control

Property Name	Type	Property Value
<code>com.bea.contextelement.als b.router.ProxyService</code>	<code>java.lang.String</code>	The service name (full-name; for example <code>/myproject/myfolder/svc-a</code>).
<code>com.bea.contextelement.als b.router.ServiceUri</code>	<code>java.net.URI</code>	The base URI from which the message was received.
<code>com.bea.contextelement.als b.router.inbound.Transport Provider</code>	<code>java.lang.String</code>	The Id of the transport provider that received this message.
<code>com.bea.contextelement.als b.router.inbound.request.M essageId</code>	<code>java.lang.String</code>	This is the transport provider-specific message identifier. Ideally it should uniquely identify the message among other messages going through the Oracle Service Bus runtime. However, Oracle Service Bus does not depend on the message Id being unique. The message Id is added to the message context and thus visible in the pipeline.
<code>com.bea.contextelement.als b.router.inbound.request.C haracterEncoding</code>	<code>java.lang.String</code>	Character encoding used in the message payload, or null.
<code>com.bea.contextelement.wli .Message</code>	<code>java.io.InputStream</code>	The request message as an input stream.

45.8.3 Additional Transport-Specific Context Properties

In addition to the properties in [Table 45–7](#), other transport-specific properties may be present. For each transport request-header (see the transport SDK), a property with the name

```
com.bea.contextelement.alsb.router.inbound.request.headers.<provider-id>.<header-name>
```

is present, where `provider-id` is the transport provider id, and `header-name` is one of the request-headers declared in the provider's schema file.

The type and semantics of these properties is transport-specific. For HTTP proxy services, the properties in [Table 45–3](#) are also available.

Table 45–7 Additional Message-Level Security ContextHandler Properties for HTTP Proxy Services

Property Name	Type	Property Value
com.bea.contextelement.alsb.router.inbound.request.metadata.http.relative-URI	java.lang.String	The relative URI of the request.
com.bea.contextelement.alsb.router.inbound.request.metadata.http.query-string	java.lang.String	The query string that is contained in the request URL after the path.
com.bea.contextelement.alsb.router.inbound.request.metadata.http.client-host	java.lang.String	The fully qualified name of the client that sent the request.
com.bea.contextelement.alsb.router.inbound.request.metadata.http.client-address	java.lang.String	The Internet Protocol (IP) address of the client that sent the request.

45.8.4 Administrator-Supplied Context Properties for Message-Level Authentication

Both custom username/password authentication and custom token authentication allow users (who are in the **IntegrationAdmin** or **IntegrationDeployer** roles) to pass additional context information to the security provider in the **Context Properties** field on the **Security** tab.

You can configure additional context properties by entering the **Property Name** as a literal string, and the **Value Selector** as a valid XPath expression. (XPath expressions can also be literal strings.)

The XPath expression is evaluated at runtime against the same message part that is used for the custom token or custom username/password. That is, the **Value Selector** XPath expressions are evaluated against the header for SOAP-based proxy services, and against the body for non-SOAP-based proxy services.

45.8.5 Security Provider Must Have Knowledge of the Property Name

A ContextHandler is essentially a name/value list and, as such, it requires that a security provider know what names to look for. Therefore, for both transport- and message-level custom authentication, the XPath expressions are evaluated only if an Authentication provider or Identity Assertion provider asks for the value of one of these properties.

This means that your configured Authentication or Identity Assertion provider must explicitly know which property names to request via the `ContextHandler.getValue(propertyName)` method. The only way to satisfy this requirement is for you, or a third party, to write a custom Authentication or Identity Assertion provider.

For example, [Example 45–1](#) shows how to get the `HttpServletRequest` property from a provider that you write.

Example 45–1 Getting the HttpServletRequest Property

```

:
Object requestValue = handler.getValue("com.bea.contextelement.alsb.transport.http.http-request");
if ((requestValue == null) || (!(requestValue instanceof HttpServletRequest)))
return;
HttpServletRequest request = (HttpServletRequest) requestValue;

```

```
log.println(" " + HTTP_REQUEST_ELEMENT + " method: " + request.getMethod());
log.println(" " + HTTP_REQUEST_ELEMENT + " URL: " + request.getRequestURL());
log.println(" " + HTTP_REQUEST_ELEMENT + " URI: " + request.getRequestURI());
return;
```

If the security provider does not need the value of the user-defined property, then the XPath expression is not evaluated.

45.8.6 Oracle WebLogic Server Administrative Channel is Supported

This release of Oracle Service Bus can use the Oracle WebLogic Server administrative channel.

As described in "Understanding Network Channels" in *Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server*, a Oracle WebLogic Server network channel is a configurable resource that defines the attributes of a network connection to Oracle WebLogic Server.

You can configure a particular type of network channel, called an administrative channel, to isolate "administration" and application ("business") traffic in a WebLogic domain. The administrative channel is a secured channel that accepts only SSL connections.

In Oracle Service Bus, business traffic is comprised of all messages sent to and from Oracle Service Bus proxy services and business services. SSL business traffic must use the default Oracle WebLogic Server secure network channel.

Administration traffic is comprised of all communication with the Oracle WebLogic Server Administration Console, Oracle Service Bus Administration console, internal traffic within a cluster, and traffic between administration scripts and admin or managed servers.

When an administrative channel is enabled in a domain, all of the administration traffic in that domain must go through that channel. Otherwise, the administration traffic also uses the default Oracle WebLogic Server secure network channel.

Using the Administrative Channel: Main Steps

1. Close any open browser connections to the Oracle Service Bus Administration Console for the domain.

As soon as you activate the administrative channel in Oracle WebLogic Server, the Oracle Service Bus Administration Console for the domain becomes unavailable at the current URL. The Help system also becomes unavailable.

2. Enable the domain-wide administration port in the Oracle WebLogic Server Administration Console (which configures an administrative channel on your behalf), or explicitly create an administrative channel. Both of these tasks are described in "Configuring Network Resources" in *Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server*.

The domain-wide administration port control is located on the Domain > Configuration > General page. The default administration port is 9002.

Be sure to activate the change.

3. Open a browser connection to the new URL for the Oracle Service Bus Administration Console for the domain.

The URL is `https://hostname:9002/sbconsole` if you enabled the domain-wide administration port and accepted the default port number.

4. Revise any startup scripts that refer to the old URL. If you are using the Windows graphical interface to launch the Oracle Service Bus Administration Console for the domain, revise the shortcut property to reflect the new URL.

45.9 Using Security Providers

This section provides instructions on using security providers with Oracle Service Bus.

45.9.1 Configuring Authentication Providers

Check the provided Oracle WebLogic Server Authentication providers to see if one meets your needs. Oracle WebLogic Server includes a broad array of Authentication providers, including the following:

- The WebLogic Authentication provider accesses user and group information in Oracle WebLogic Server's embedded LDAP server. This is the default out-of-the-box authentication provider. This provider is not optimized for use with very large numbers of users.
- *LDAP Authentication providers* access external LDAP stores. You can use an LDAP Authentication provider to access any LDAP server. Oracle WebLogic Server provides LDAP Authentication providers already configured for Open LDAP, Sun iPlanet, Microsoft Active Directory and Novell NDS LDAP servers.
- *RDBMS Authentication providers* access external relational databases. Oracle WebLogic Server provides three RDBMS Authentication providers: SQL Authenticator, Read-only SQL Authenticator, and Custom RDBMS Authenticator.
- The *SAML Authentication provider*, which authenticates users based on Security Assertion Markup Language 1.1 (SAML) assertions.

See "Improving the Performance of WebLogic and LDAP Authentication Providers" in *Oracle Fusion Middleware Securing Oracle WebLogic Server* for guidance on improving the performance of these authentication providers.

As described in "Why Customize the Default Security Configuration" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*, you may want to use an Authentication provider that accesses a database other than Oracle WebLogic Server's embedded LDAP server. For example, you might want to use a different authentication provider for the majority of user accounts, but continue to use the default authentication provider (embedded LDAP) for Oracle Service Bus and Oracle WebLogic Server administrative user accounts.

Using the WebLogic Authentication provider for all Oracle WebLogic Server and Oracle Service Bus administrative user accounts provides reliable access in the event of a network or database problem. Oracle recommends that you use the default WebLogic Authentication provider for all Oracle WebLogic Server and Oracle Service Bus administrative accounts for this reason.

If one of the bundled Authentication providers meets your needs, see "Configuring Authentication Providers" in *Oracle Fusion Middleware Securing Oracle WebLogic Server* for instructions on how to configure this Authentication provider in the Oracle WebLogic Server Administration Console.

If none of the Authentication providers included in Oracle WebLogic Server suits your needs, you (or a third-party) must first write a custom Authentication provider and then use the Oracle WebLogic Server Administration Console to add that provider to the security realm, as described in the following steps:

Note: Only a broad overview of the required tasks is included here. You will need to consult the Oracle WebLogic Server documentation to actually complete the tasks.

Adding a Provider to a Security Realm

1. "Create Runtime Classes Using the Appropriate SSPIs" (in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server*).
2. "Generate an MBean Type Using the WebLogic MBeanMaker"
3. "Configure the Custom Authentication Provider Using the Administration Console"

45.9.2 Using a Custom Authorization Provider to Protect Oracle Service Bus Resources

You can use Oracle Service Bus resources with custom Authorization providers, but those providers must understand the type and format of the Oracle Service Bus resources.

There are three possible resource objects for Oracle Service Bus that an Authorization provider must be able to detect and handle:

- [Section 45.9.2.2, "Oracle Service BusProxyServiceResource Object"](#)
- [Section 45.9.2.3, "ProjectResourceV2 Object"](#)
- [Section 45.9.2.4, "ConsoleResource Object"](#)

These resource objects are described in the sections that follow.

45.9.2.1 WebLogic Authorization Provider Usage Information

This section briefly describes the Oracle WebLogic Server Authorization provider SSPI. See "Authorization Providers" in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server* for complete information.

You protect resources by binding access control policies to resources via the Oracle Service Bus console, third-party tools or scripts. The Oracle WebLogic Server Security Service Provider Interface (SSPI) requires containers, such as Oracle Service Bus, to implement the Resource SPI. These implementations represent concrete resources.

The Authorization provider database contains a map from resource to policy. When an attempt is made to access a resource, the container calls the runtime SSPI to get an access control decision. The container passes a resource instance indicating which resource is being accessed.

An Authorization provider has one method, `getAccessDecision()`. The `getAccessDecision()` method obtains the implementation of the `AccessDecision` SSPI. The `AccessDecision` SSPI itself has one method, `isAccessAllowed()`. `isAccessAllowed` has five parameters, one of which is the Resource object for which access is being requested.

`isAccessAllowed` determines if the requestor should be allowed to access the named resource. To do this, the Authorization provider must find the right access control policy to evaluate. The provider must first look for a policy bound to the resource passed in. The lookup can use either the `getId()` or `toString()` method as a lookup key, as described in "Looking Up WebLogic Resources in a Security Provider's Runtime Class" in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server*. If no policy is found, the Authorization provider must then get the

parent resource and look again. This process is repeated until a policy is found or the parent is null, in which case no policy is found. When no policy is found, `isAccessAllowed` must return false.

This algorithm allows you to create coarse-grained policies that protect all proxy services in a given project or folder, all resources in a project, or all Oracle Service Bus proxy services in an Oracle Service Bus domain. More specific, finer-grained policies take precedence over coarse-grained policies.

Note: The Oracle Service Bus console user interface does not provide pages for protecting proxy services at the folder, project or domain level.

45.9.2.2 Oracle Service BusProxyServiceResource Object

The `ALSBProxyServiceResource` object is used for transport-level and message-level access control to Oracle Service Bus proxy services. The `ALSBProxyServiceResource` resource extends `weblogic.security.service.ResourceBase`, which itself implements `weblogic.security.spi.Resource`.

`ALSBProxyServiceResource` implements the following methods, as described in `weblogic.security.spi.Resource`:

getType() – Returns the type, where type is "<alsb-proxy-service>"

getKeys() – Returns up to four key-value properties: `path`, `proxy`, `action`, and `operation`. The properties are defined as follows:

- `path` is the full-name of the proxy service. For example, `path=project/folder1/folder2`
- `proxy` is the name of the proxy service. For example, `proxy=myProxy`
- `action` is one of two values, `invoke` or `wss-invoke`. For example, `action=invoke`

The `action` attribute is used to distinguish between transport-level and message-level access control. `invoke` is used for transport-level access control. `wss-invoke` is used for message-level access control; that is, access control on WS-Security active intermediaries or proxies with custom message-level authentication. The `operation` attribute is only allowed when `action` is `wss-invoke`.

- `operation` is the name of the operation to invoke, and is used only when `action` is `wss-invoke`. For example, `operation=processPO`. The `operation` attribute is only allowed when `action` is `wss-invoke`.

An `ALSBProxyServiceResource` has from 1 to 4 keys. The following table explains how the various combinations protect proxy services. The most specific policies take precedence.

If the Resource Contains These Keys	A Policy Bound to the Resource Protects:
<code>path</code>	The policy protects all proxy services in the given path
<code>path</code> and <code>proxy</code>	The policy protects all access to the given proxy service (transport-level as well as message-level)

If the Resource Contains These Keys	A Policy Bound to the Resource Protects:
path, proxy, and action	If action="invoke": The policy is the transport-level policy to the given proxy If action="wss-invoke": The policy is the message-level policy to the given proxy (for all operations)
path, proxy, action="wss-invoke", and operation	The policy is a message-level policy for the given proxy and operation

getPath() – Gets the path (project and folders) to the proxy service. This is the path where the proxy service exists within the Oracle Service Bus configuration framework.

getProxyServiceName() – Gets the name of the proxy service. For example, proxy=myProxy.

getAction() – Gets one of two values, invoke or wss-invoke. For example, action=invoke.

getOperation() – Gets the name of the operation to invoke, and is used only when action is wss-invoke. For example, operation=processPO.

makeParent() – Creates a new `ALSBProxyServiceResource` object that represents the parent of the current `ALSBProxyServiceResource` resource. `makeParent()` uses the path of the proxy service to create the parent.

45.9.2.2.1 ALSBProxyServiceResource Examples The following examples show various uses of the `ALSBProxyServiceResource` object.

- Using `ALSBProxyServiceResource` for transport-level access control for proxy project/folder/myProxy:


```
type=<alsb-proxy-service>, path=project/folder, proxy=myProxy, action=invoke
```
- Using `ALSBProxyServiceResource` for message-level access control for operation processPO on proxy project/folder/myProxy:


```
type=<alsb-proxy-service>, path=project/folder, proxy=myProxy,
action=wss-invoke, operation=processPO
```
- Using the parentage hierarchy for an `ALSBProxyServiceResource`, from fine-grained to coarse-grained:


```
type=<alsb-proxy-service>, path=myProject/f1/f2, proxy=myProxy,
action=wss-invoke, operation=foo
type=<alsb-proxy-service>, path=myProject/f1/f2, proxy=myProxy,
action=wss-invoke
type=<alsb-proxy-service>, path=myProject/f1/f2, proxy=myProxy
type=<alsb-proxy-service>, path=myProject/f1/f2
type=<alsb-proxy-service>, path=myProject/f1
type=<alsb-proxy-service>, path=myProject
type=<alsb-project>, project-name=myProject
type=<alsb-proxy-service>
```

45.9.2.3 ProjectResourceV2 Object

The `ProjectResourceV2` is the root resource for all `ALSBProxyServiceResource` objects in a given project. `ProjectResourceV2` extends `ResourceBase`.

Setting an access control policy on a `ProjectResourceV2` provides a coarse-grained access control policy for all proxy services in the given project that do not have more specific policies.

`ProjectResourceV2` has the following methods:

getType() – Returns the type, where type is "`<alsb-project>`".

getKeys() – Returns the key, where key is "`project-name`".

getName() – Gets the name of the `ProjectResourceV2` object.

makeParent() – There is no parent for an `ProjectResourceV2` object. This method therefore returns the object name that was used to create the `ProjectResourceV2` object, or null if `ProjectResourceV2` does not exist.

45.9.2.4 ConsoleResource Object

The `com.bea.wli.security.resource.ConsoleResource` object is used for access control to the Oracle Service Bus console. However, we do not recommend that you set access control policies for `ConsoleResource` objects via a custom Authorization provider. This is because these policies are subject to change in future Oracle Service Bus releases.

We instead recommended that even if you need to use a custom Authorization provider, you also continue to use the Oracle WebLogic Server XACML Authorization provider to maintain the policies for the `ConsoleResource` object. In this case of two Authorization providers, you must also configure an Adjudication provider.

Oracle Service Bus Security FAQ

This section includes frequently asked questions about Oracle Service Bus security and their answers. It includes the following questions:

- [Section 46.1, "How are Oracle Service Bus and Oracle WebLogic Server Security related?"](#)
- [Section 46.2, "What is Transport-Level Security?"](#)
- [Section 46.3, "What is Web Services Security?"](#)
- [Section 46.4, "What is Web Service Policy?"](#)
- [Section 46.5, "What are Web Service Policy assertions?"](#)
- [Section 46.6, "Are Access Control Policy and Web Service Policy the same?"](#)
- [Section 46.7, "What is Web Services Security Pass-Through?"](#)
- [Section 46.8, "What is a Web Services Security Active Intermediary?"](#)
- [Section 46.9, "What is outbound Web Services Security?"](#)
- [Section 46.10, "What is SAML?"](#)
- [Section 46.12, "What is the Certificate Lookup And Validation Framework?"](#)
- [Section 46.13, "Does Oracle Service Bus support identity propagation in a proxy service?"](#)
- [Section 46.14, "If both transport-level authentication and message-level authentication exist on inbound messages to the proxy service, which identity is propagated?"](#)
- [Section 46.11, "Is it possible to customize the format of the subject identity in a SAML assertion?"](#)
- [Section 46.15, "Is single sign-on supported in Oracle Service Bus?"](#)
- [Section 46.16, "Are security errors monitored?"](#)
- [Section 46.17, "Can I configure security for MBeans?"](#)

46.1 How are Oracle Service Bus and Oracle WebLogic Server Security related?

Oracle Service Bus leverages the WebLogic Security Framework. The details of this framework are described in "WebLogic Security Framework" in "WebLogic Security Service Architecture" in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*. Before configuring security in Oracle Service Bus, you must configure

an Oracle WebLogic Server security realm and other server configurations (such as SSL) in Oracle WebLogic Server, as described in [Section 45.7, "Configuring the Oracle WebLogic Security Framework: Main Steps."](#)

46.2 What is Transport-Level Security?

Transport-level security refers to the transport protocols that secure the connection over which messages are transported. An example of transport-level security is HTTPS (HTTP over SSL). SSL provides point-to-point security, but does not protect the message when intermediaries exist in the message path. For more information, see [Chapter 49, "Configuring Transport-Level Security."](#)

46.3 What is Web Services Security?

Web Services Security (WS-Security) is an OASIS standard that defines interoperable mechanisms to incorporate message-level security into SOAP messages. WS-Security supports message integrity and message confidentiality. It also defines an extensible model for including security tokens in a SOAP envelope and a model for referencing security tokens from within a SOAP envelope. WS-Security token profiles specify how specific token types are used within the core WS-Security specification. Message integrity is achieved through the use of XML digital signatures; message confidentiality is accomplished through the use of XML encryption. WS-Security allows you to specify which parts of a SOAP message are digitally signed or encrypted. Oracle Service Bus supports WS-Security over HTTP (including HTTPS) and JMS.

46.4 What is Web Service Policy?

The Web Services Policy Framework (WS-Policy) provides a general-purpose model and corresponding syntax to describe and communicate the policies of a Web service. WS-Policy defines a base set of constructs that can be used and extended by other Web service specifications to describe a broad range of service requirements, preferences, and capabilities. For more information, see [Chapter 51, "Using WS-Policy in Oracle Service Bus Proxy and Business Services."](#)

46.5 What are Web Service Policy assertions?

The Web Services Policy Assertions Language (WS-PolicyAssertions) specifies a set of common message policy assertions that can be specified within a security policy. The specification defines general messaging-related assertions for use with WS-Policy. Separate specifications describe the syntax and semantics of domain-specific assertions for security assertions and reliable-messaging assertions.

46.6 Are Access Control Policy and Web Service Policy the same?

No. Access control policy is a boolean expression that is evaluated to determine which requests to access a particular resource (such as a proxy service, Web application, or EJB) are granted and which should be denied access. Typically access control policies are based on the *roles* of the requestor. WS-Policy is metadata about a Web service that complements the service definition (WSDL). WS-Policy can be used to express a requirement that all service clients must satisfy, such as, all requests must be digitally signed by the client.

46.7 What is Web Services Security Pass-Through?

In a WS-Security pass-through scenario, the client applies WS-Security to the request and/or response messages. The proxy service does not process the security header, instead, it passes the secured request message untouched to a business service. Although Oracle Service Bus does not apply any WS-Security to the message, it can route the message based on values in the header. After the business service receives the message, it processes the security header and acts on the request. The business service must be configured with WS-Policy security statements. The secured response message is passed untouched back to the client. For example, the client encrypts and signs the message and sends it to the proxy service. The proxy service does not decrypt the message or verify the digital signature, it simply routes the message to the business service. The business service decrypts the messages and verifies the digital signature, and then processes the request. The response path is similar. This is sometimes called a passive intermediary.

46.8 What is a Web Services Security Active Intermediary?

In an active intermediary scenario, the client applies WS-Security to the request and/or response messages. The proxy service processes the security header and enforces the WS-Security policy. For example, the client encrypts and signs the message and sends it to the proxy service, the proxy decrypts the message and verifies the digital signature, then routes the message. Before the proxy service sends the response back to the client, the proxy signs and encrypts the message. The client decrypts the message and verifies the proxy's digital signature.

46.9 What is outbound Web Services Security?

Outbound WS-Security refers to security between Oracle Service Bus proxy services and business services. It includes both the request and response between business applications and proxy services. For more information, see [Section 52.1, "About Message-Level Security."](#)

46.10 What is SAML?

SAML (Security Assertion Markup Language) is an OASIS standards-based extensible XML framework for exchanging authentication and authorization information, allowing single sign-on capabilities in modern network environments.

46.11 Is it possible to customize the format of the subject identity in a SAML assertion?

By default, the subject identity within an outbound SAML token is the same as the inbound username. The format of the subject identity can be customized by writing a custom SAML name mapper-provider. For more information, see "Configuring a SAML Credential Mapping Provider" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

46.12 What is the Certificate Lookup And Validation Framework?

The Certificate Lookup and Validation (CLV) providers complete certificate paths and validate X509 certificate chains. The two types of CLV providers are:

CertPath Builder—receives a certificate, a certificate chain, or certificate reference (the end certificate in a chain or the Subject DN of a certificate) from a Web service or application code. The provider looks up and validates the certificates in the chain.

CertPath Validator—receives a certificate chain from the SSL protocol, a Web service, or application code and performs extra validation, such as revocation checking.

At least one CertPath Builder and one CertPath Validator must be configured in a security realm. Multiple CertPath Validators can be configured in a security realm. If multiple providers are configured, a certificate or certificate chain must pass validation with all the CertPath Validators for the certificate or certificate chain to be valid. Oracle WebLogic Server provides the functionality of the CLV providers in the WebLogic CertPath provider and the Certificate Registry. For more information see "The Certificate Lookup and Validation Process" in "WebLogic Security Service Architecture" in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*.

46.13 Does Oracle Service Bus support identity propagation in a proxy service?

Yes, Oracle Service Bus supports two methods for propagating identities:

- By generating SAML assertions in conformance with the Web Services Security. This is done by setting a SAML holder-of-key or sender-vouches WS-Policy on the business service routed to by the proxy.
- If a business service requires user name and password tokens, you can configure the business service's service account to pass through the user credentials from the original client request. See [Chapter 2.1.16, "Creating Service Account Resources."](#)

46.14 If both transport-level authentication and message-level authentication exist on inbound messages to the proxy service, which identity is propagated?

If both transport authentication and message-level authentication exist, the message-level subject identity is propagated.

46.15 Is single sign-on supported in Oracle Service Bus?

Strictly speaking single sign-on (SSO) is not applicable to Oracle Service Bus messaging scenarios for several reasons. First, Oracle Service Bus is stateless; there is no notion of a session or conversation among multiple parties. Second, Oracle Service Bus clients are typically other enterprise software applications, not users behind a Web browser. Therefore, it is acceptable to require that these clients send credentials such as username and password on every request, provided that the communication is secured by means such as SSL or WS-Security. However, SSO between the Oracle Service Bus Console and the Oracle WebLogic Server Administration Console is supported. For more information, see "Single Sign-On" in "Security Fundamentals" in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*.

46.16 Are security errors monitored?

Only WS-Security errors are monitored by the Oracle Service Bus monitoring framework. Transport-level security errors such as SSL handshake errors,

transport-level authentication and transport-level access control are not monitored. For more information, see "Monitoring Oracle Service Bus at Run Time" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*. However, it is possible to configure an Auditor provider to audit transport-level authentication and authorization.

46.17 Can I configure security for MBeans?

Oracle Service Bus includes two managed beans (MBeans) that configure such runtime behavior as which types of credentials are available to abstract WS-Policy statements. By default, only users in the Admin and Deployer security roles can modify these MBeans, however you can change these defaults. See "Create JMX Policies" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Help*.

Configuring Administrative Security

To give users access to administrative functions such as creating proxy services, you assign them to one of four security roles with pre-defined access privileges. A security role is an identity that can be dynamically conferred upon a user or group based on conditions that are evaluated at runtime. You cannot change the access privileges for the Oracle Service Bus administrative security roles, but you can change the conditions under which a user or group is in one of the roles.

The following sections describe administrative security for Oracle Service Bus:

- [Section 47.1, "Administrative Security Roles and Privileges"](#)
- [Section 47.2, "Administrative Security Groups"](#)
- [Section 47.3, "Configuring Administrative Security: Main Steps"](#)

For more information about security roles, see "Users, Groups, and Security Roles" in *Oracle Fusion Middleware Securing Resources Using Roles and Policies for Oracle WebLogic Server*.

47.1 Administrative Security Roles and Privileges

[Table 47-1](#) describes the Oracle Service Bus administrative security roles and summarizes their access privileges.

Table 47-1 Oracle Service Bus Administrative Security Roles

Role	Pre-Defined Access Privileges
IntegrationAdmin and IntegrationDeployer	<p>Has complete access to all Oracle Service Bus resources, including the ability to create, edit, or delete user names, passwords, and credential alias bindings in service accounts and service key providers. The user names and passwords that this role can create are used only by service accounts for outbound authentication; they are not used to authorize access to Oracle Service Bus resources.</p> <p>Cannot create, edit, or delete users, groups, roles, or access control policies in the Security Configuration module of the Oracle Service Bus Console.</p>

Table 47-1 (Cont.) Oracle Service Bus Administrative Security Roles

Role	Pre-Defined Access Privileges
IntegrationOperator	<p>This group has the following privileges:</p> <ul style="list-style-type: none"> ■ Has read access to all Oracle Service Bus resources. ■ Cannot export resources. ■ Has access to create, view, edit and delete alert rules. ■ Has access to session management, including create, commit, discard and undo of sessions. Cannot view all sessions. ■ Has access to create, edit, view and delete operational settings of services.
IntegrationMonitor	<ul style="list-style-type: none"> ■ Has read access to all Oracle Service Bus resources. ■ Cannot export resources.

Note: In this release, IntegrationAdministrators and IntegrationDeployers have the same privileges. This might change in future releases.

The Oracle Service Bus roles have permission to modify only Oracle Service Bus resources; they do not have permission to modify Oracle WebLogic Server or other resources on Oracle WebLogic Server. To give permission to modify Oracle WebLogic Server its other resources, add a user to one of the Oracle WebLogic Server security roles described in [Table 47-2](#). In each Oracle Service Bus domain, make sure that you add at least one user to the Admin role.

Table 47-2 Oracle WebLogic Server Security Roles

Oracle WebLogic Server Role	Default Access Privileges
Admin	Has complete access to all Oracle WebLogic Server and Oracle Service Bus objects and functions, including the ability to create, edit, or delete users, groups, roles, or access control policies.
Deployer	Has read access to all objects. Can create, delete, edit, import or export resources, services, service key providers, or projects.
Operator	Has read and export access to all objects. Can configure alerts, enable or disable metric collection, and suspend or resume services.
Monitor	Has read access to all objects. Can export any resource, service, service key provider, or project.

47.1.1 Role-Based Access in the Oracle Service Bus Console

[Table 47-3](#) through [Table 47-8](#) show the actions that each Oracle Service Bus security role can perform in the Oracle Service Bus Console.

Permission to perform an action is indicated by a (Y) in the table. Only the Oracle WebLogic Server Admin role has Security Configuration privileges.

Table 47-3 Role-Based Operations Access in Oracle Service Bus Console

Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
View Statistics	Y	Y	Y	Y
Reset Statistics	Y	Y	Y	N
View Alerts	Y	Y	Y	Y
Delete Alerts	Y	Y	Y	N
View Alert History	Y	Y	Y	Y
View Server Summary	Y	Y	Y	Y
View Dashboard Settings	Y	Y	Y	Y
Set Dashboard Settings	Y	Y	Y	Y
Set Smart Search Settings	Y	Y	Y	N
View Smart Search Settings	Y	Y	Y	Y
Set Global Settings	Y	Y	Y	N
View Global Settings	Y	Y	Y	Y
Set Tracing Settings	Y	Y	Y	N
View Tracing Settings	Y	Y	Y	N
View Message Reports	Y	Y	Y	Y
Purge Messages	Y	Y	Y	N

Table 47-4 Role-Based Resource Browser Access in Oracle Service Bus Console

Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
Create Proxy Service	Y	Y	N	N
View Proxy Service	Y	Y	Y	Y
Edit Proxy Service	Y	Y	N	N
Delete Proxy Service	Y	Y	N	N
Create Business Service	Y	Y	N	N
View Business Service	Y	Y	Y	Y

Table 47-4 (Cont.) Role-Based Resource Browser Access in Oracle Service Bus

Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
Edit Business Service	Y	Y	N	N
Delete Business Service	Y	Y	N	N
Create WSDLs	Y	Y	N	N
View WSDLs	Y	Y	Y	Y
Edit WSDLs	Y	Y	N	N
Delete WSDLs	Y	Y	N	N
Create XML Schemas	Y	Y	N	N
View XML Schemas	Y	Y	Y	Y
Edit XML Schemas	Y	Y	N	N
Delete XML Schemas	Y	Y	N	N
Create WS-Policy	Y	Y	N	N
View WS-Policy	Y	Y	Y	Y
Edit WS-Policy	Y	Y	N	N
Delete WS-Policy	Y	Y	N	N
Create XQuery	Y	Y	N	N
View XQuery	Y	Y	Y	Y
Edit XQuery	Y	Y	N	N
Delete XQuery	Y	Y	N	N
Create XSLT	Y	Y	N	N
View XSLT	Y	Y	Y	Y
Edit XSLT	Y	Y	N	N
Delete XSLT	Y	Y	N	N
Create MFL	Y	Y	N	N
View MFL	Y	Y	Y	Y
Edit MFL	Y	Y	N	N
Delete MFL	Y	Y	N	N
Create JARs	Y	Y	N	N
View JARs	Y	Y	Y	Y
Edit JARs	Y	Y	N	N
Delete JARs	Y	Y	N	N
Create Service Account	Y	Y	N	N

Table 47-4 (Cont.) Role-Based Resource Browser Access in Oracle Service Bus

Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
View Service Account	Y	Y	Y	Y
Edit Service Account	Y	Y	N	N
Delete Service Account	Y	Y	N	N
Create service key provider	Y	Y	N	N
View service key provider	Y	Y	Y	Y
Edit service key provider	Y	Y	N	N
Delete service key provider	Y	Y	N	N
Create Alert Rule	Y	Y	Y	N
View Alert Rule	Y	Y	Y	Y
Edit Alert Rule	Y	Y	Y	N
Delete Alert Rule	Y	Y	Y	N

Table 47-5 Role-Based Project Explorer Access in Oracle Service Bus Console

Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
Create Project	Y	Y	N	N
View Project	Y	Y	Y	Y
Edit Project	Y	Y	N	N
Delete Project	Y	Y	N	N
Create Folder	Y	Y	N	N
View Folder	Y	Y	Y	Y
Edit Folder	Y	Y	N	N
Delete Folder	Y	Y	N	N

Table 47-6 Role-Based Security Configuration Access in Oracle Service Bus Console

Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
Create User	N	N	N	N
View User	Y	Y	Y	Y
Edit User	N	N	N	N
Delete User	N	N	N	N
Create Group	N	N	N	N

Table 47-6 (Cont.) Role-Based Security Configuration Access in Oracle Service Bus

Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
View Group	Y	Y	Y	Y
Edit Group	N	N	N	N
Delete Group	N	N	N	N
Create Role	N	N	N	N
View Role	Y	N	N	N
Edit Role	N	N	N	N
Delete Role	N	N	N	N
Create Policy	N	N	N	N
View Policy	N	N	N	N
Edit Policy	N	N	N	N
Delete Policy	N	N	N	N

Table 47-7 Role-Based System Administration Access in Oracle Service Bus Console

Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
Import Resources	Y	Y	N	N
Export Resources	Y	Y	N	N
Create UDDI Registries	Y	Y	N	N
View UDDI Registries	Y	Y	Y	Y
Edit UDDI Registries	Y	Y	N	N
Delete UDDI Registries	Y	Y	N	N
Import from UDDI	Y	Y	N	N
Synchronize Auto-Import Status	Y	Y	Y	Y
Detach UDDI	Y	Y	N	N
Publish to UDDI	Y	Y	N	N
Auto-Publish Status	Y	Y	Y	Y
Publish Auto-Publish Status	Y	Y	N	N
Create JNDI Providers	Y	Y	N	N

Table 47-7 (Cont.) Role-Based System Administration Access in Oracle Service Bus

Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
View JNDI Providers	Y	Y	Y	Y
Edit JNDI Providers	Y	Y	N	N
Delete JNDI Providers	Y	Y	N	N
Create SMTP Servers	Y	Y	N	N
View SMTP Servers	Y	Y	Y	Y
Edit SMTP Servers	Y	Y	N	N
Delete SMTP Servers	Y	Y	N	N
Find Value (Customization)	Y	Y	N	N
Replace With (Customization)	Y	Y	N	N
Create File (Customization)	Y	Y	N	N
Select File (Customization)	Y	Y	N	N
Select Items (Customization)	Y	Y	N	N
Execute File (Customization)	Y	Y	N	N

Table 47-8 Role-Based Change Center Access in Oracle Service Bus Console

Actions	Integration Admin	Integration Deployer	Integration Operator	Integration Monitor
Edit Session	Y	Y	Y	N
View All Sessions	Y	Y	N	N
View Changes	Y	Y	Y	N
Activate Changes	Y	Y	Y	N
Discard Changes	Y	Y	Y	N
Exit Session	Y	Y	Y	N

47.2 Administrative Security Groups

To facilitate the process of assigning users to the pre-defined administrative roles, Oracle Service Bus also provides four corresponding security groups. While membership in a role is dynamic, membership in a group is static: an administrator

places a user in a group and the user remains in the group until the administrator changes the assignment.

In the simplest scenario for configuring administrative security, you create a user, add the user to one of the four administrative groups, and the user is automatically always a member of the corresponding role with all of the pre-defined access privileges.

In a more complex scenario, you might create two of your own groups, MyAdministratorsEast and MyAdministratorsWest, and assign users appropriately. You configure the pre-defined IntegrationAdmin security role so that the MyAdministratorsWest group is in the role from 8am to 8pm EST, while the MyAdministratorsEast group is in the role from 8pm to 8am EST.

[Table 47-9](#) describes the administrative groups that Oracle Service Bus provides. You can create your own groups in addition to these.

Table 47-9 Oracle Service Bus Groups

By Default, This Group...	Is Always in This Role...
IntegrationAdministrators	IntegrationAdmin. See Table 47-1 .
IntegrationDeployers	IntegrationDeployer. See Table 47-1 .
IntegrationOperators	IntegrationOperator. See Table 47-1 .
IntegrationMonitors	IntegrationMonitor. See Table 47-1 .

47.3 Configuring Administrative Security: Main Steps

You can create or modify users, groups, and roles when you are in or out of an Oracle Service Bus session. Any additions or modifications to this data take effect immediately and are available to all sessions. If you discard a session in which you added or modified the data, the security data is **not** discarded.

To configure administrative security:

1. Log in to the Oracle Service Bus Console with a user account that is in the Oracle WebLogic Server Admin role.
2. (Optional) Create your own security groups.

See "Adding Groups" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

3. Create users and assign them to one of the Oracle Service Bus groups or one of your own groups.

See "Adding Users" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

4. (Optional) Modify the conditions under which users and groups are in the pre-defined Oracle Service Bus security roles.

By default, the four default groups are always in the Oracle Service Bus security roles, but you can change this default. To more easily manage your list of users, Oracle recommends that you never add users directly to a role. Instead, add users to a group and add the group to the role.

See "Adding Roles" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

Securing Oracle Service Bus in a Production Environment

To prepare an Oracle Service Bus installation for production, you must pay special attention to your security needs. The following list outlines some of the tasks you need to perform:

- Read and follow the guidelines in *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server* in the Oracle WebLogic Server documentation.
- Create user accounts for the Oracle Service Bus administrators and assign them to one or more of the following groups as appropriate: IntegrationAdministrators, IntegrationOperators, IntegrationMonitors, and IntegrationDeployers. For more information, see "Security Configuration" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.
- In your file system, configure access control to the directory that contains Oracle Service Bus configuration data. This is the `sbconfig` directory under the domain root. For example:

```
C:\oracle\user_projects\domains\base_domain\osb\config
```
- In your file system, configure access control to the directories used by the FTP, SFTP, file, and email transports.
- If necessary, configure access control to the JMS resources used by your Oracle Service Bus installation.

48.1 Undeploying the Service Bus (SB) Resource

Oracle Service Bus provides a resource servlet (`MW_HOME/OSB_HOME/lib/sbresourceWar/sbresource.war`) that is used to expose the resources registered in Oracle Service Bus. The resources registered with Oracle Service Bus include:

- WSDL (a WSDL registered as a resource in Oracle Service Bus)
- Schema
- MFL
- WS-Policy
- WSDL (an effective WSDL with resolved policies and port information for a proxy service—this effective WSDL is available if the proxy service was created using a WSDL).

However, this servlet provides anonymous HTTP access to metadata, and as such it may be considered a security risk in some high-security environments.

If you do not want the Oracle Service Bus resources to be available anonymously via HTTP, you can set security roles on `sbresources.war` to control access to it, or completely undeploy the resource.

Note: If you undeploy the SB resource you will no longer be able to use the UDDI subsystem.

48.2 Protection of Temporary Files With Streaming body Content

As described in "The Message Context Model" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* for processing message content, you can specify that the Oracle Service Bus pipeline streams the content rather than loading it into memory. When you enable content streaming for a proxy service, you specify whether to buffer the streamed content to memory or a disk file as an intermediate step during the processing of the message.

If you use these temporary disk files, you should protect them.

To lock-down your Oracle Service Bus domain, set the `com.bea.wli.sb.context.tmpdir` java system property to specify where these temporary files will be written.

Make sure this directory exists and has the right set of access permissions.

For more information see the file access permission and file system recommendations in *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server*.

48.3 Protecting Against Denial of Service Attacks on the Oracle Service Bus Console

In a production environment, the Oracle Service Bus Console should not be accessible to users other than administrators.

A denial of service attack can take the form of a high volume of requests from a single source or new connections being made to the server once resource constraints have reached a certain point.

Following are suggestions for protecting against denial of service attacks on the Oracle Service Bus Console.

- In a production environment, make sure the administration server—the server the Oracle Service Bus Console runs on—is never made public. Only managed servers should be available to callers.
- Instead of using the default Work Manager for the Oracle Service Bus Console, configure and use a different Work Manager that sets a default limit on the number of users that can access the Oracle Service Bus Console Web application (max-threads-constraint).

For information about Work Managers, see "Using Work Managers to Optimize Scheduled Work" in *Oracle Fusion Middleware Configuring Server Environments for Oracle WebLogic Server*.

Configuring Transport-Level Security

Transport-level security applies security checks as part of establishing a connection between service consumers, proxy services, and business services. The type of security checks that Oracle Service Bus can apply depends on the protocol that the proxy service or business service uses to communicate. Some protocols can also encrypt the communication between client and endpoint to prevent snooping from third parties.

Inbound transport-level secures the communication between clients and Oracle Service Bus proxy services. **Outbound** transport security secures all three techniques of sending outbound requests from Oracle Service Bus proxy services: route actions, publish actions, and callout actions.

The following sections describe configuring transport-level security:

- [Section 49.1, "Configuring Transport-Level Security for HTTPS"](#)
- [Section 49.2, "Configuring Transport-Level Security for HTTP"](#)
- [Section 49.3, "Configuring Transport-Level Security for JMS"](#)
- [Section 49.4, "Configuring Transport-Level Security for SFTP Transport"](#)
- [Section 49.5, "Email, FTP, and File Transport-Level Security"](#)
- [Section 49.6, "Configuring Transport-Level Security for SB Transport"](#)
- [Section 49.7, "Configuring Transport-Level Security for WS Transport"](#)
- [Section 49.8, "Configuring Transport-Level Security for WebSphere Message Queue Transport"](#)
- [Section 49.9, "Transport-Level Security Elements in the Message Context"](#)

Note: Transport-level security secures only the connection itself. Even if you use the HTTPS or JMS protocols to encrypt the communication, if there is an intermediary between a Web services client and an Oracle Service Bus proxy service, such as a router, message queue or another proxy service, the intermediary gets the SOAP message in plain text. When the intermediary sends the message to the second receiver, the second receiver does not know who the original sender was. To prevent unintended intermediaries from viewing or modifying SOAP or JMS messages, configure message-level security *in addition to* transport-level security. See [Chapter 52, "Configuring Message-Level Security for Web Services."](#)

49.1 Configuring Transport-Level Security for HTTPS

Note: In previous releases of Oracle Service Bus, HTTPS was managed via the HTTPS transport. HTTPS is part of the HTTP transport.

This section has been updated to reflect the new configuration model.

The HTTPS protocol uses SSL to secure communication. SSL can be used to encrypt communication, ensure message integrity, and to require strong server and client authentication. Before you can use HTTPS, you must configure SSL in Oracle WebLogic Server, see [Section 45.7, "Configuring the Oracle WebLogic Security Framework: Main Steps."](#)

The following sections describe configuring transport-level security for the HTTPS protocol:

- [Section 49.1.1, "HTTPS Authentication Levels"](#)
- [Section 49.1.2, "Configuring Inbound HTTPS Security: Main Steps"](#)
- [Section 49.1.3, "Configuring Outbound HTTPS Security: Main Steps"](#)

49.1.1 HTTPS Authentication Levels

For each proxy service or business service that communicates over the HTTPS protocol, you can configure the service to require one of the following levels of authentication:

- One-way SSL, no authentication
This level enables encrypted communication but does not require clients to provide credentials. To establish a one-way SSL connection, the client initiates the connection and Oracle Service Bus sends its certificate to the client. In other words, the client authenticates Oracle Service Bus.
- One-way SSL, BASIC authentication
This level enables encrypted communication and requires clients to supply a user name and password after the one-way SSL connection is established. The client supplies a user name and password by encoding it in the HTTP request header (which is encrypted by SSL). When the proxy service receives the encrypted request, it passes the credentials to the domain's authentication provider, which determines whether client's credentials match a user account that you have created.
- Two-way SSL, CLIENT CERT authentication
This level enables encrypted communication and strong client authentication (two-way SSL).
To establish a two-way SSL connection, the client initiates the connection and Oracle Service Bus sends its X.509 certificate to the client. Then, the client sends its certificate to Oracle Service Bus and Oracle Service Bus authenticates the client.
To get the user name from the client's certificate, you configure an identity assertion provider, which extracts a field in the certificate to use as the client identity (X.509 token), typically the CN (common name) or E (email) of the SubjectDistinguishedName in the certificate. After extracting the X.509 token,

the token is compared to the user accounts in the Security Configuration module of the Oracle Service Bus Console.

For more information about SSL and identity assertion providers, see "Security Fundamentals" in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*.

- Transport-Level Custom Credentials.

You can authenticate client requests at the transport-level via custom authentication tokens. Transport-level custom credentials are supported only on inbound requests. You specify a custom token in an HTTP header. The HTTP-specific configuration pages of the service definition wizard allows you to configure client authentication. Custom authentication concepts are described in [Chapter 54, "Configuring Custom Authentication."](#)

49.1.2 Configuring Inbound HTTPS Security: Main Steps

To configure inbound transport-level security for a proxy service:

1. Make sure that you have configured the WebLogic security framework to support SSL, an authentication provider, and an identity assertion provider, depending on the HTTPS authentication level that you want to use:
 - For no client authentication (anonymous requests), set Client Authentication to None.
 - For basic authentication, set Client Authentication to Basic. See "Adding Users" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.
 - For SSL client authentication, set Client Authentication to Client Certificate, configure the WebLogic Identity Assertion provider and the WebLogic CertPath Provider.
 - For custom authentication token, set Client Authentication to Custom Authentication. The custom authentication token can be any active token type previously configured for an Identity Assertion provider that is carried in an HTTPS header. Custom authentication concepts are described in [Chapter 54, "Configuring Custom Authentication."](#)

Note: You must first configure, or create and configure, an Oracle WebLogic Server Identity Assertion provider as described in [Section 54.5, "Configuring Identity Assertion Providers for Custom Tokens,"](#) and add the user names and passwords of the clients that you want to allow access to the Security Configuration module of the Oracle Service Bus Console.

See [Section 45.7, "Configuring the Oracle WebLogic Security Framework: Main Steps."](#)

2. When you create a proxy service in the Oracle Service Bus Console, on the **Transport Configuration** page select HTTP.
3. On the HTTP Transport Configuration page, click the HTTPS check box.
4. Choose an authentication level, as described in [Section 49.1.1, "HTTPS Authentication Levels."](#)

5. Make your Dispatch Policy, Request Encoding, and Response Encoding choices, as described in [Section 4.3, "Proxy Service Configuration."](#)
6. If the service you are creating has operations, make your selections on the Operation Selection Configuration page. Determine whether to enforce WS-I compliance (for SOAP 1.1 services only) and select the selection algorithm to use to determine the operation called by this proxy service. This option is available only for SOAP or XML services defined from a WSDL.

49.1.3 Configuring Outbound HTTPS Security: Main Steps

In outbound transport-level security, a proxy service is the client that opens a connection with a business service.

To configure outbound transport-level security:

1. If you are configuring transport-level security for a production environment (as opposed to a development or testing environment), make sure that Host Name Verification is enabled. See "Using Host Name Verification" in "Configuring SSL" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.
2. When you create a business service, on the Transport Configuration page, select HTTP.

Follow the prompts to choose an authentication level.

If you configured the proxy service such that Oracle Service Bus does not authenticate clients, configure the enterprise system to authenticate clients by selecting an authentication level of one-way SSL, BASIC authentication.

3. The URI determines whether HTTP or HTTPS is used. HTTP business services can combine HTTP and HTTPS URLs unless the authentication method is Client Certificate, in which case all URLs must be HTTPS.
4. If the business service uses HTTPS with BASIC authentication, create a service account to provide the user name and password that the business service requires.

You can add a user name and password directly to the service account, or configure the service account to pass through the credentials that it received from its client's request, or you can map a client user name to an Oracle Service Bus user. If you configured the proxy service so that Oracle Service Bus does not authenticate clients, create a service account that passes through the credentials.

5. If the business service uses Client Certificate authentication, do the following:
 - a. Create a service key provider to provide the key-pair that proxy services use for SSL client authentication with the business service. See "Service Key Providers" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.
 - b. Create a proxy service or edit an existing proxy service so that it specifies the service key provider. See [Section 2.3, "Working with Proxy Services."](#)

49.2 Configuring Transport-Level Security for HTTP

The HTTP protocol does **not** encrypt communication between clients and proxy services or business services, but it does support BASIC authentication in which clients send user names and passwords in requests. HTTP also supports custom token authentication.

Caution: Unless you have configured strong network security, Oracle recommends that you do not use BASIC authentication with HTTP in production environments because the password is sent in clear text. Instead, use BASIC authentication with HTTPS.

The following sections describe configuring transport-level security for the HTTP protocol:

- [Section 49.2.1, "Configuring Inbound HTTP Security: Main Steps"](#)
- [Section 49.2.2, "Configuring Outbound HTTP Security: Main Steps"](#)

49.2.1 Configuring Inbound HTTP Security: Main Steps

To configure inbound transport-level security for a proxy service:

1. When you create a proxy service in the Oracle Service Bus Console, on the **Transport Configuration** page select HTTP. Choose the Client Authentication option None, Basic, or Custom Authentication. If you choose Custom Authentication, you must also specify the HTTP header that is to carry the token and the token type.

The steps for configuring transport-level custom credentials are described in [Section 2.3, "Working with Proxy Services."](#) Custom authentication concepts are described in [Chapter 54, "Configuring Custom Authentication."](#)

The custom authentication token can be any active token type, previously configured for an Identity Assertion provider, that is carried in an HTTP header.

Note: To use custom authentication you must first configure, or create and configure, an Oracle WebLogic Server Identity Assertion provider as described in [Section 54.5, "Configuring Identity Assertion Providers for Custom Tokens."](#)

If you want Oracle Service Bus to authenticate clients (Basic or Custom Authentication) you must create user accounts for the clients. See [Section 47.3, "Configuring Administrative Security: Main Steps."](#)

2. Modify the proxy service's default transport-level access control policy, which specifies conditions under which users, groups, or roles can access a proxy service. See "Editing Transport-Level Access Policies" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

49.2.2 Configuring Outbound HTTP Security: Main Steps

In outbound transport-level security, a proxy service is the client that opens a connection with a business service.

To configure outbound transport-level security:

1. When you create a business service, on the Transport Configuration page select HTTP. When prompted, select **Basic Authentication Required**.
See [Section 2.2, "Working with Business Services."](#)
2. Create a service account to provide the user name and password that the business service requires. See [Section 2.1.16, "Creating Service Account Resources."](#)

You can add a user name and password directly to the service account, or configure the service account to pass through the credentials that it received from its client's request, or you can map a client user name to an Oracle Service Bus user. If you configured the proxy service so that Oracle Service Bus does not authenticate clients, create a service account that passes through the credentials.

3. Create a proxy service or edit an existing proxy service so that it specifies the service account.

49.3 Configuring Transport-Level Security for JMS

While transport-level security for JMS does not provide end-to-end security for JMS messaging, it does provide the following:

- The option to use a secure SSL channel for communication between Oracle Service Bus and a JMS server for sending or receiving JMS messages.

Oracle Service Bus can communicate with local JMS servers or foreign JMS servers. The connection to JMS servers can be secured using the T3S protocol (T3 over SSL). T3 and T3S are proprietary Oracle protocols.

- The ability to specify the username and password that Oracle Service Bus proxy services use to authenticate while establishing a connection to a JMS server and/or while looking up JMS destinations in the JNDI tree.

Note: JMS administrators use the Oracle WebLogic Server Administration Console to create access control policies that restrict access to WebLogic JMS servers and destinations in the JNDI tree. For more information, see *Oracle Fusion Middleware Securing Resources Using Roles and Policies for Oracle WebLogic Server* and "Methods for Configuring JMS System Resources" in *Oracle Fusion Middleware Configuring and Managing JMS for Oracle WebLogic Server*.

If a JMS administrator configures or changes an access control policy for a JMS destination, Oracle WebLogic Server can take up to 60 seconds to recognize the changes.

By default, Oracle WebLogic Server JMS checks the policy for each JMS destination every 60 seconds. To change this behavior, modify the Oracle WebLogic Server startup command so that it sets the following system property to the frequency (in seconds) that you want Oracle WebLogic Server JMS to check access control policies:

```
weblogic.jms.securityCheckInterval
```

A value of 0 (zero) for this property ensures that an authorization check is performed for every `send`, `receive`, and `getEnumeration` action on a JMS resource.

The following sections describe configuring JMS transport-level security:

- [Section 49.3.1, "Configuring Inbound JMS Transport-Level Security: Main Steps"](#)
- [Section 49.3.2, "Configuring Outbound JMS Transport-Level Security: Main Steps"](#)

49.3.1 Configuring Inbound JMS Transport-Level Security: Main Steps

To configure inbound JMS transport-level security:

1. When you create or edit a JMS proxy service, on the Transport Configuration page, under Advanced Settings, select the **Use SSL** check box.

Oracle Service Bus configures the JMS proxy service to use the T3S protocol.

2. If the JMS administrator created access control policies that restrict access to a JMS connection pool, configure the proxy service to authenticate when it connects to the JMS server:

- a. Create a service account to provide the user name and password that the JMS server requires. See [Section 2.1.16, "Creating Service Account Resources."](#)

The JMS service account for the proxy service is used not only for the JMS object access, but also for the JNDI lookup.

You must add a user name and password directly in the service account. JMS cannot use a service account that passes through the credentials that it received from its client's request or that maps a client user name to an Oracle Service Bus user.

- b. When you create or edit the proxy service in the Oracle Service Bus Console, on the Transport Configuration page, under Advanced Settings, click the **Browse** button next to JMS Service Account. Select the service account that you created in the previous step.

49.3.2 Configuring Outbound JMS Transport-Level Security: Main Steps

To configure outbound JMS transport-level security:

1. When you create or edit a JMS business service in the Oracle Service Bus Console, on the Transport Configuration page, under Advanced Settings, select the **Use SSL** check box.

Oracle Service Bus configures the JMS business service to use the T3S protocol.

2. If the JMS administrator created access control policies that restrict access to a JMS connection pool, configure the business service to authenticate when it connects to the JMS server:

- a. Create a service account to provide the user name and password that the JMS server requires. See [Section 2.1.16, "Creating Service Account Resources."](#)

The JMS service account for the proxy service is used not only for the JMS object access, but also for the JNDI lookup.

You must add a user name and password directly in the service account. JMS cannot use a service account that passes through the credentials that it received from its client's request or that maps a client user name to an Oracle Service Bus user.

- b. When you create or edit the business service in the Oracle Service Bus Console, on the Transport Configuration page, under Advanced Settings, click the **Browse** button next to JMS Service Account. Select the business account that you created in the previous step.
3. Select the **Pass Caller's Subject** check box to have Oracle Service Bus pass the authenticated subject when sending a message.

49.4 Configuring Transport-Level Security for SFTP Transport

As described in [Section 26.5, "SFTP Transport,"](#) Oracle Service Bus supports the SFTP transport for inbound and outbound transport-level security. The SFTP transport uses Secure Shell (SSH) version 2 to transfer files.

49.4.1 How Two-Way Authentication is Performed

The SFTP authentication is two-way: both the SFTP server and SFTP client (Oracle Service Bus service) authenticate each other, via different mechanisms:

- The SFTP server uses the authentication method you specified in the **Transport Configuration** page to authenticate the SFTP client (the Oracle Service Bus service): Username Password, Host Based, or Public Key.
- The SFTP client (the Oracle Service Bus service) uses a `known_hosts` file to authenticate the SFTP server. The `known_hosts` file on the Oracle Service Bus proxy service (inbound requests) or business service (outbound requests) system must have the hostname, IP address, and public key of the remote SFTP servers to which the proxy service or business service can connect. SSH version 2 uses this public key to authenticate the connection.

The SFTP client (the Oracle Service Bus service) always uses the `known_hosts` file to determine whether to connect to an SFTP server, no matter which of the three authentication methods is chosen in the **Transport Configuration** page. That is, in all cases the SFTP server is authenticated by the Oracle Service Bus service using the information present in this file. This ensures that the Oracle Service Bus service is connecting to a known server.

For example, in case of Username Password authentication, the SFTP Client (Oracle Service Bus Service) authenticates the SFTP server against the SFTP server's public key in the `known_hosts` file. The SFTP server authenticates the client (Oracle Service Bus service) with the username and password from the service account.

49.4.2 Use of the `known_hosts` File

No matter which authentication method you choose in the **Transport Configuration** page, a `known_hosts` file on the Oracle Service Bus proxy service (inbound requests) or business service (outbound requests) system must have the hostname, IP address, and public key of the remote SFTP servers to which the proxy service or business service can connect.

The Oracle Service Bus service authenticates the SFTP server with the public-key/host/IP combination present in the `known_hosts` file.

Note: This SSH authentication mechanism is outside of the typical Oracle Service Bus service key provider/PKI credential mapper process.

The `known_hosts` file requirement must be satisfied during authentication. SFTP servers not listed in the `known_hosts` file are not authenticated.

Creating the `known_hosts` File

1. Use the editor of your choice to create a `known_hosts` text file.

The format for `known_hosts` is as follows:

```
Hostname,IP algorithm public-key
```


where `Hostname`, `IP`, and `public_key` identify the SFTP server.

The algorithms supported are RSA (entered only as `ssh-rsa`) and DSA (entered only as `ssh-dsa` or `ssh-dss`).

The public key format for this file is "OpenSSH public key format."

For example:

```
getafix,172.22.52.130 ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAtR+M3Z9HFxnKZTx66fZdnQqAHQcF1vQe1+EjJ/HWYtgAnqsn0hM
JzqWMatb/u9yFwUpZBirjm3g2I9Qd8VocmeHwoGPhDGFQ5LQ/PPo3esE+CGwdnCOyRcktNHeuKxo4ki
CCJ/bph5dRpghCQIvsQvRE3sks+XwQ7Wuswz8pv58=
```

Multiple entries are supported, one entry per line.

2. Move the `known_hosts` file to the

`MW_HOME/user_projects/domains/osb_domain/osb/transports/sftp`

directory. The directories `/transports/sftp` are not created automatically. You must create them.

49.4.3 SFTP Transport Authentication Process

The following general principles apply to the SFTP authentication process for both a proxy service and business service.

- **Connection:** The Oracle Service Bus service (proxy and business) always acts as the SFTP client and connects to the SFTP server.
- **Authentication by the SFTP Server:** For Public Key and Host Based authentication, the SFTP server authenticates the connection with the public key of the Oracle Service Bus service. For Username Password, the SFTP server authenticates the connection with the username and password.
- **Authentication by the SFTP Client:** The Oracle Service Bus service always authenticates the SFTP server with the public-key/host/IP combination present in the `known_hosts` file.
- **Connection established:** If both the server and client authentications are successful, only then is the connection established and ready for transfer.
- **Transfer:** The file (message) is downloaded in case of the proxy service and uploaded in the case of the business service.

The SFTP authentication process is as follows:

- Inbound one-way download to the proxy service:
 1. The proxy service, which is the SFTP client, attempts to connect to the SFTP server.
 2. The proxy service is authenticated by the SFTP server via the authentication mechanism selected on the **Transport Configuration** page.

For Host Based and Public Key authentication, the remote SFTP server uses the host name and public key of the proxy service to authenticate the Oracle Service Bus system. For Username Password authentication, the SFTP server uses the username and password supplied by the proxy service (via the service account) to authenticate the Oracle Service Bus system.

3. A `known_hosts` file (on the Oracle Service Bus proxy service system) keeps the information of the remote SFTP servers to which the Oracle Service Bus proxy service can connect.

Specifically, this file contains the host name, IP address, and public key of the accepted remote servers.

SSH version 2 uses this public key to authenticate the connection. SFTP servers not listed in the `known_hosts` file are not authenticated.

4. If authentication is successful, the proxy service is the SFTP client connected to the remote SFTP server.
5. If allowed by the SFTP server, the proxy service (the SFTP client) polls a remote directory on the SFTP server and downloads any files (messages) present in the remote directory.

The proxy service configuration determines which remote directory to poll, how often to poll it, and what to do with any files (messages) that it downloads.

- Outbound one-way upload from the business service:
 1. The business service (which is the SFTP client) attempts to connect to the SFTP server.
 2. The business service is authenticated by the SFTP server via the authentication mechanism selected on the **Transport Configuration** page.

For Host Based and Public Key authentication, the SFTP server uses the host name and public key of the business service to authenticate the Oracle Service Bus system. For Username Password authentication, the SFTP server uses the username and password (from the service account) to authenticate the Oracle Service Bus system.

3. A `known_hosts` file (on the Oracle Service Bus business service system) keeps the information of the SFTP servers to which the Oracle Service Bus business service can connect.

Specifically, this file contains the host name, IP address, and public key of the accepted servers.

SSH version 2 uses this public key to authenticate the connection. SFTP servers not listed in the `known_hosts` file are not authenticated.

4. If authentication is successful, the business service is the SFTP client connected to the remote SFTP server.
5. If allowed by the SFTP server, the business service (the SFTP client) uploads files to the remote directory on the SFTP server.

The business service configuration determines in which remote directory to upload the file, how often to retry the upload, and any file prefix or suffix to add to the uploaded file name.

49.4.4 Configuring Inbound SFTP Transport-Level Security: Main Steps

To configure inbound transport-level security for a proxy service:

1. Create a `known_hosts` file, as described in [Section 49.4.2, "Use of the known_hosts File,"](#) on the Oracle Service Bus proxy service system.

`known_hosts` keeps the information of the remote SFTP servers to which the Oracle Service Bus proxy service can connect. Specifically, this file contains the host name, IP address, and public key of the accepted remote servers.

SSH version 2 uses this public key to authenticate the connection. SFTP servers not listed in the `known_hosts` file are not authenticated.

2. When you create a proxy service in the Oracle Service Bus Console, on the **Transport Configuration** page select SFTP.
3. Specify the end point URI in `sftp://hostname:port/directory` format, where:
 - `hostname` is the host name or IP address of the SFTP server.
 - `port` is the port on which SFTP server is listening. Default port for SFTP is 22.
 - `directory` is the location that is periodically polled for files. This directory is relative to the home directory of the user.
4. On the **SFTP Transport Configuration** page, select either Username Password, Host Based, or Public Key authentication.

The authentication choices are summarized here. See [Section 26.5, "SFTP Transport"](#) for complete information.

- Username/Password authentication specifies that a static service account (using user credentials on the SFTP server) is associated with this authentication method. The service account provides a user name and password that the proxy service uses for authentication to the SFTP server. The SFTP client is authenticated using the provided credentials. Only the static service account type is supported.
- Host Based Authentication specifies that only connections from identified, known hosts are allowed. This authentication method requires a username and a service key provider.

The SFTP Server authenticates the proxy service with the public key of the proxy service.

Note: The Oracle Service Bus proxy service does not itself use the service key provider to authenticate any credentials from the SFTP server. It uses only the `known_hosts` file to authenticate the SFTP server.

The public key of the proxy service is present in the key-pair referred by the service key provider. You need to extract this key when you set up the service key provider, and then configure the SFTP server to use the public key.

For example, with SFTP server on Linux, you need to:

- Edit the `/etc/ssh/shosts.equiv` file and add the host name or IP address of the machine on which Oracle Service Bus domain is running.
- Edit the `/etc/ssh/ssh_known_hosts` file and add the host name or IP address of the machine on which Oracle Service Bus domain is running, followed by space and the public key.

The username is used to determine which directory on the SFTP server to poll.

- Public Key specifies a username and service key provider are required to use this authentication method. Every user has their own private key.

The SFTP Server authenticates the proxy service with the public key.

Note: The Oracle Service Bus proxy service does not itself use the service key provider to authenticate any credentials from the SFTP server. It uses only the `known_hosts` file to authenticate the SFTP server.

The public key of the proxy service is present in the key-pair referred by the service key provider. You need to extract this key when you set up the service key provider, and then configure the SFTP server to use the public key.

For example, to allow access to a system for a given identity with an SFTP server on Linux, place the public key in a `$HOME/.ssh/authorized_keys` file on that system. All keys listed in that file are allowed access.

The username is used to determine which directory on the SFTP server to poll. It is also used to identify the location of the public key on the SFTP server.

5. If allowed by the remote SFTP server, the proxy service (SFTP client) polls a remote directory on the SFTP server and downloads any files present in the remote directory.

The proxy service configuration determines which remote directory to poll, how often to poll it, and what to do with any files (messages) that it downloads.

The directory to be polled is an absolute path.

49.4.5 Configuring Outbound SFTP Transport-Level Security: Main Steps

To configure outbound transport-level security for a business service:

1. Create a `known_hosts` file, as described in [Section 49.4.2, "Use of the known_hosts File,"](#) on the Oracle Service Bus business service system.

`known_hosts` keeps the information of the remote SFTP servers to which the Oracle Service Bus business service can connect. Specifically, this file contains the host name, IP address, and public key of the accepted remote servers.

SSH version 2 uses this public key to authenticate the connection. SFTP servers not listed in the `known_hosts` file are not authenticated.

2. When you create a business service in the Oracle Service Bus Console, on the **Transport Configuration** page select SFTP.
3. Specify the end point URI in `sftp://hostname:port/directory` format, where:
 - `hostname` is the host name or IP address of the SFTP server.
 - `port` is the port on which SFTP server is listening. Default port for SFTP is 22.
 - `directory` is the location to which files are uploaded. This directory is relative to the home directory of the user.
4. On the **SFTP Transport Configuration** page, select either Username Password, Host Based, or Public Key authentication.

The authentication choices are summarized here. See [Section 26.5, "SFTP Transport"](#) for complete information.

- Username/Password authentication specifies that a static service account (using user credentials on the SFTP server) is associated with this

authentication method. The service account provides a user name and password that the business service uses for authentication to the SFTP server. The SFTP client is authenticated using the provided credentials. Only the static service account type is supported.

- Host Based Authentication specifies that only connections from identified, known hosts are allowed. This authentication method requires a username and a service key provider.

The SFTP Server authenticates the business service with the public key of the business service.

Note: The Oracle Service Bus business service does not itself use the service key provider to authenticate any credentials from the SFTP server. It uses only the `known_hosts` file to authenticate the SFTP server.

The public key of the business service is present in the key-pair referred by the service key provider. You need to extract this key when you set up the service key provider, and then configure the SFTP server to use the public key.

For example, with SFTP server on Linux, you need to:

- Edit the `/etc/ssh/shosts.equiv` file and add the host name or IP address of the machine on which Oracle Service Bus domain is running.
- Edit the `/etc/ssh/ssh_known_hosts` file and add the host name or IP address of the machine on which Oracle Service Bus domain is running, followed by space and the public key.

The username is used to determine the upload directory on the SFTP server.

- Public Key specifies a username and service key provider are required to use this authentication method. Every user has their own private key.

The SFTP Server authenticates the business service with the public key.

Note: The Oracle Service Bus business service does not itself use the service key provider to authenticate any credentials from the SFTP server. It uses only the `known_hosts` file to authenticate the SFTP server.

The public key of the business service is present in the key-pair referred by the service key provider. You need to extract this key when you set up the service key provider, and then configure the SFTP server to use the public key.

For example, to allow access to a system for a given identity with an SFTP server on Linux, place the public key in a `$HOME/.ssh/authorized_keys` file on that system. All keys listed in that file are allowed access.

The username is used to determine the upload directory on the SFTP server and for identifying the location of the public key on the SFTP server.

5. If allowed by the remote SFTP server, the business service (SFTP client) uploads files to the remote directory on the SFTP server.

The business service configuration determines in which remote directory to upload the file, how often to retry the upload, and any file prefix or suffix to add to the uploaded file name.

The upload directory is an absolute path and is automatically created.

49.4.6 SFTP Security Attributes Preserved During Import

The following security attributes are preserved when [Section 45.6.1, "Preserve Security and Policy Configuration Check Box"](#) is turned on during import:

- Client authentication method
- Reference to the service account (in case of Username Password authentication)
- Reference to the service key provider (in case of Host Based and Public Key authentication)
- Username (in case of Host Based and Public Key authentication)

49.4.7 SFTP Credential Life Cycle

Whenever the username/password or public key credential changes, the SFTP transport drops all idle connections made with the previous credential and attempts to reconnect. For active connections, the SFTP transport drops the connection after the current operation is finished.

49.5 Email, FTP, and File Transport-Level Security

The following sections describe the security measures that are available for communication over the email, FTP, and file protocols:

- [Section 49.5.1, "Email and FTP Transport-Level Security"](#)
- [Section 49.5.2, "File Transport Security"](#)

49.5.1 Email and FTP Transport-Level Security

Email and FTP are not secure protocols. They support weak authentication, typically over insecure channels. The supported security method for email or FTP transport is the username and password needed to connect to the email or FTP server.

To secure email, you must designate a service account as an alias for the username and password in the Oracle Service Bus Console. The service will use the username and password to authenticate to the SMTP server.

To secure the FTP transport, in the Oracle Service Bus Console, select `external_user` and designate a service account as an alias for the username and password. The service will use the username and password to authenticate to the FTP server.

For information about how to add security to email and FTP transport, see [Section 2.2, "Working with Business Services"](#)

49.5.2 File Transport Security

The supported security method for file transport is the user login to the computer on which the files are located.

The SFTP transport, described in [Section 49.4, "Configuring Transport-Level Security for SFTP Transport,"](#) is the preferred mechanism to secure FTP.

49.6 Configuring Transport-Level Security for SB Transport

The Service Bus (SB) transport allows client Oracle Service Bus servers to invoke an Oracle Service Bus proxy service synchronously via RMI. RMI is the only mechanism by which client Oracle Service Bus servers can access the SB transport. In this release of Oracle Service Bus the associated API is for internal user only and is not documented.

The SB proxy service is accessed in one of two ways:

- By a client Oracle Service Bus server that uses an SB business service to connect to the Oracle Service Bus server of the proxy service by using the JNDI context and the proxy service URI.
- By products such as Oracle WebLogic Integration and Oracle Data Service Integrator that use proprietary artifacts to access SB proxy services. These artifacts are unique to those products and are not described here.

The SB business service can send messages only to SB proxy services. A JNDI provider, which is specified in the endpoint URI of the business service, is used to do a JNDI lookup on the remote Oracle Service Bus server. Specifically, the JNDI provider points to the Oracle Service Bus server where the service is deployed to retrieve the RMI stubs corresponding to the SB proxy service.

For example, the endpoint URI you specify in the business service could be `sb://some_secured_jndi_provider/some_remote_sb_proxy`.

A secure JNDI provider should have a provider URL with a secure protocol. In the SB business service case, you can use the HTTPS or t3s protocols.

The service account (of the business service) specifies the user credentials that should be used for invoking the remote SB proxy service. If no service account is specified, the user credentials of the inbound proxy service (the inbound client) of this business service are used for security context propagation.

The SB transport can use SSL to require strong server and client authentication. Before you can use the SB transport with SSL, you must configure SSL in Oracle WebLogic Server. See [Section 45.7, "Configuring the Oracle WebLogic Security Framework: Main Steps."](#)

Caution: When set, the Use SSL flag means that request must be sent over an SSL connection. However, the SB transport does not forbid unsecured connections. The proxy service will be advertised (via the effective WSDL or UDDI) with a secured URI (indicated by `sbs`), but secured access is not enforced.

The Oracle Service Bus server administrator must close all unsecured protocols on the server (`t3`, `http`, and so forth) to strictly enforce secured-client connections.

49.6.1 Configuring SAML Authentication With Service Bus (SB) Transport

If you are using SAML-based authentication with the SB transport, be sure to follow these configuration requirements:

- On the SB client side, configure a SAML Credential mapper provider and create a SAML relying party for each SB proxy service you plan to invoke from this client. In the target URL field enter `http://openuri.org/<OSBProxyServiceURI>`, where `OSBProxyServiceURI` is the service URI of the SB proxy service.

- On the Oracle Service Bus side (where the SB proxy service resides), configure a SAML Identity Assertion provider and create a SAML asserting party. In the target URL field enter the service URI of the SB proxy service. Do not include the SB protocol or host/port information. For example, `/<OSBProxyServiceURI>`.

49.7 Configuring Transport-Level Security for WS Transport

Web service reliable messaging (WS-RM) functionality is available in Oracle Service Bus as the WS transport. Oracle Service Bus supports the specification submitted in February 2005. For more information about the specification, see Web Services Reliable Messaging Protocol (WS-ReliableMessaging) at <http://schemas.xmlsoap.org/ws/2005/02/rm/>.

The WS transport has both proxy service (inbound) and business service (outbound) components that are based on SOAP1.1- and SOAP1.2-based WSDLs, along with WS-RM policy. It supports both one-way and request-response patterns, but response is unreliable.

49.7.1 Reliable Web Services Messaging Defined

As described in "Overview of Web Service Reliable Messaging" in *Oracle Fusion Middleware Programming Advanced Features of JAX-RPC Web Services for Oracle WebLogic Server*, WS-RM is a framework in which an application running in one application server can reliably invoke a web service running on another application server, assuming that both servers implement the WS-ReliableMessaging specification. "Reliable" is defined as the ability to guarantee message delivery between the two web services. In particular, the specification describes an interoperable protocol in which a message sent from a source endpoint (or client web service) to a destination endpoint (or web service whose operations can be invoked reliably) is guaranteed either to be delivered, according to one or more delivery assurances, or to raise an error.

49.7.2 WS Transport Resources Visible in WLS Console

WS proxy services are visible from the WLS console, but attempts to assign policies from WLS are ignored.

Specifically, administrators can navigate to the **Home > Summary of Security Realms > myrealm > Realm Roles** pages in the WLS console and seemingly edit the security policy for the WS proxy service.

However, this policy will have no effect and it will not be evaluated at runtime.

The EAR application is auto-generated and deployed by Oracle Service Bus when you activate the session. This is one EAR file for each WS proxy service.

49.7.3 Use of WS-Policy Files for Web Service Reliable Messaging Configuration

You configure WS transport security through WS-Policy files, either from a WSDL or bound directly to the service.

Oracle Service Bus use WS-Policy files to enable a destination endpoint to describe and advertise its WS-RM capabilities and requirements. The WS-Policy specification provides a general purpose model and syntax to describe and communicate the policies of a web service.

These WS-Policy files are XML files that describe features such as the version of the supported WS-ReliableMessaging specification, the source endpoint's retransmission interval, the destination endpoint's acknowledgment interval, and so on.

WS-Policy with RM assertions and WSSP transport-level security assertions are supported for the WS transport only.

49.7.3.1 Preconfigured WS-RM Policy Files

Oracle Service Bus includes two simple WS-RM WS-Policy files that you can specify if you do not want to create your own WS-Policy files:

- `DefaultReliability.xml` – Specifies typical values for the reliable messaging policy assertions, such as inactivity timeout of 10 minutes, acknowledgement interval of 200 milliseconds, and base re-transmission interval of 3 seconds.
- `LongRunningReliability.xml` – Similar to the default reliable messaging WS-Policy file, except that it specifies a much longer activity timeout interval (24 hours.)

You cannot change these pre-packaged files. If their values do not suit your needs you must create your own WS-Policy file.

For example, the complete `LongRunningReliability.xml` file (as extracted from `weblogic.jar`) is shown in [Example 49-1](#):

Example 49-1 *LongRunningReliability.xml File*

```
<?xml version="1.0"?>
<wsp:Policy
  xmlns:wsmr="http://schemas.xmlsoap.org/ws/2005/02/rm/policy"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:beapolicy="http://www.bea.com/wsmr/policy"
  >
  <wsmr:RMAssertion >
    <wsmr:InactivityTimeout
      Milliseconds="86400000" />
    <wsmr:BaseRetransmissionInterval
      Milliseconds="3000" />
    <wsmr:ExponentialBackoff />
    <wsmr:AcknowledgementInterval
      Milliseconds="200" />
    <beapolicy:Expires Expires="P1M" optional="true"/>
  </wsmr:RMAssertion>
</wsp:Policy>
```

49.7.4 RM WS-Policy Required Prior to Activation

A proxy or business service that uses the WS transport must have a WS-Policy with RM assertions, either from a WSDL or bound directly to the service. Services that use any other transport must not have a WS-Policy with RM assertions.

You can bind RM assertions only at the service level and not at the operation or request/response levels.

49.7.5 Async Responses

WS-RM supports two messaging patterns: one way, and request/response. The WS transport supports both patterns, but does not support reliable response. That is, the response is not sent reliably but the request is always reliable.

Async responses from a proxy service using the WS transport to an RM client connect to the `AcksTo` or `ReplyTo` endpoint references specified by the RM client. The RM client is free to use an HTTP or HTTPS URL. When using HTTPS, the RM client is free

to request a client certificate during the SSL handshake. The WS transport will use the SSL key-pair of the service key provider upon request.

49.7.6 Proxy Service Authentication

The WS transport supports the following HTTPS security modes via WS-Policy files:

- HTTPS – no client authentication
- HTTPS with BASIC authentication
- HTTPS with client-certificate authentication (2-way SSL)

[Table 49–1](#) shows the preconfigured security policies that implement these modes and indicates when you might use them.

Table 49–1 WS Transport Authentication Matrix

HTTPS Required	Authentication Required	Preconfigured Transport Security Policy
Yes	None	Wssp1.2-Https.xml
Yes	BASIC	Wssp1.2-HttpsBasic.xml
Yes	Client-certificate	Wssp1.2-HttpsClientCert.xml

WS proxy services support both basic and client-certificate (2-way SSL) authentication, as determined by the WSSP 1.2 transport-level security assertions in the WS-Policy.

Consider the example of the HTTPS token and the Basic256 algorithm as extracted from the packaged Wssp1.2-Https.xml policy, as shown in [Example 49–2](#).

When basic authentication is specified in the WS-policy, all HTTPS requests (including RM protocol messages to the WS proxy service) must have a valid username and password.

Example 49–2 Wssp1.2-Https.xml File (Partial)

```

:
<sp:TransportBinding>
  <wsp:Policy >
    <sp:TransportToken>
      <wsp:Policy>
        <sp:HttpsToken />
      </wsp:Policy>
    </sp:TransportToken>
    <sp:AlgorithmSuite>
      <wsp:Policy>
        <sp:Basic256/>
      </wsp:Policy>
    </sp:AlgorithmSuite>
    <sp:Layout>
      <wsp:Policy>
        <sp:Lax/>
      </wsp:Policy>
    </sp:Layout>
    <sp:IncludeTimestamp/>
  </wsp:Policy>
</sp:TransportBinding>
</wsp:Policy>

```

Proxy service authentication is supported as follows:

- Outbound client-certificate authentication using the SSL key-pair assigned to the service key provider configured for the proxy service.

If you plan to create a service key provider (which passes key-certificate pairs in outbound requests), use the Oracle WebLogic Server Administration Console to configure a PKI credential mapping provider. In any Oracle WebLogic Server domain that hosts Oracle Service Bus, you can configure at most one PKI credential mapping provider.

- Username/password identity propagation through a WS proxy service (with basic authentication) to any other outbound transport, or outbound WSS username token.

If a business service requires user name and password tokens, you can configure the business service's service account to pass through the user credentials from the original client request. See [Section 2.1.16, "Creating Service Account Resources."](#)

- Credential mapping between WS proxy service (with basic or 2-way SSL authentication) and any other transport.
- Sending (non-reliable) asynchronous responses from a WS proxy service to an RM client via HTTP or HTTPS. The default protocol used by proxy and business services is HTTP.

Asynchronous responses from a WS proxy service to an RM client connect to the `AcksTo` or `ReplyTo` endpoint references specified by the RM client. The RM client can use either HTTP or HTTPS URL. If the RM client uses HTTPS, the RM client can request a client certificate during the SSL handshake. The WS transport uses the SSL key-pair of the service key provider upon request.

49.7.7 Preserving Security Configuration on Import

If the `Preserve Security and Policy Configuration` flag is set, the WS transport provider preserves the following security configuration: The reference to the service account (WS business services only)

49.7.8 Configuring Inbound and Outbound WS Transport-Level Security

You configure WS transport security through WS-Policy, either from a WSDL or bound directly to the service.

49.8 Configuring Transport-Level Security for WebSphere Message Queue Transport

Oracle Service Bus provides support for a native Message Queue (MQ) transport that can send messages to and from WebSphere MQ. In this context, the MQ transport is a client that connects to an MQ Server using MQ libraries.

You configure the security-related properties for the transport when you create an MQ Connection resource. These properties are then used by the MQ proxy or business service.

Note: Make sure that you add the MQ client libraries to your environment, as described in [Section 33.8.1, "Adding MQ Client Libraries to Your Environment."](#)

The MQ Connection resource has two modes:

binding mode – You use the binding mode to connect to the MQ Queue Manager located on the same machine as Oracle Service Bus. In this mode, the service calls directly into the existing queue manager API rather than communicating over the network. This mode provides a fast path to connect to local queue managers.

TCP mode – You use the `tcp` mode when the MQ Queue Manager is not available on the same machine as Oracle Service Bus.

49.8.1 Configuring Inbound MQ Transport-Level Security: Main Steps

To configure inbound transport-level security for a proxy service:

1. Before you create a proxy service that uses the MQ transport, create an MQ Connection resource for the transport to use. Choose from the following security configuration settings:
 - **SSL Required.** Select the check box to use HTTPS for sending messages. Only server-side SSL (server authenticates to client) is supported when the 2-way SSL Required option is not selected.
 - **Cipher Suite.** This option is available only when the SSL Required check box is selected. Select the Cipher Suite algorithm to be used by SSL.

A cipher suite is an SSL encryption method that includes the key exchange algorithm, the symmetric encryption algorithm, and the secure hash algorithm. A cipher suite is used to protect the integrity of a communication.

The Cipher Suite algorithm is used to encrypt and decrypt message communications between the WebSphere MQ server and the MQ Transport.
 - **2-way SSL Required.** This option is available only when the SSL Required check box is selected. Select the check box to force the use of both client-side and server-side SSL authentication.
 - **Reference to the Service Key Provider.** If you select 2-way SSL Required, you must provide a reference to the service key provider for obtaining the appropriate key manager for client-side SSL.

Enter the path (project/folder) and name of a service key provider, or click **Browse** to select one from the **Select Service Key Provider** page.
 - **Reference to the Static Service Account.** Required for user name and password authentication. Enter the path (project/folder) and name of a static service account, or click **Browse** to select a service account.
2. When you create a proxy service in the Oracle Service Bus Console, on the **Transport Configuration** page select `mq`.

49.8.2 Configuring Outbound MQ Transport-Level Security: Main Steps

To configure outbound transport-level security for a business service:

1. Before you create a proxy service that uses the MQ transport, create a MQ Connection resource for the transport to use. Choose from the following security configuration settings:
 - **SSL Required.** Select the check box to use HTTPS for sending messages. Only server-side SSL (server authenticates to client) is supported when the 2-way SSL Required option is not selected.
 - **Cipher Suite.** This option is available only when the SSL Required check box is selected. Select the Cipher Suite algorithm to be used by SSL.

A cipher suite is an SSL encryption method that includes the key exchange algorithm, the symmetric encryption algorithm, and the secure hash algorithm. A cipher suite is used to protect the integrity of a communication.

The Cipher Suite algorithm is used to encrypt and decrypt message communications between the WebSphere MQ server and the MQ Transport.

- 2-way SSL Required. This option is available only when the SSL Required check box is selected. Select the check box to force the use of both client-side and server-side SSL authentication.
- Reference to the Service Key Provider. If you select 2-way SSL Required, you must provide a reference to the service key provider for obtaining the appropriate key manager for client-side SSL.

Enter the path (project/folder) and name of a service key provider, or click Browse to select one from the **Select Service Key Provider** page.

- Reference to the Static Service Account. Required for user name and password authentication. Enter the path (project/folder) and name of a static service account, or click Browse to select a service account.

2. When you create a business service in the Oracle Service Bus Console, on the **Transport Configuration** page select `mq`.

49.9 Transport-Level Security Elements in the Message Context

If you configure a proxy service to authenticate clients, then you can access the client's identity and the security groups to which the client belongs from the proxy service's pipeline. The identity and group information is located in the message context at

```
$inbound/ctx:security/ctx:transportClient/ctx:username
```

and

```
$inbound/ctx:security/ctx:transportClient/ctx:principals/ctx:group
```

(the message context contains one `ctx:group` element for each group the user belongs to)

If a proxy service does not authenticate clients, then the value of `$inbound/ctx:security/ctx:transportClient/ctx:username` is `<anonymous>` and there will not be any `ctx:group` elements.

For more information, see "Inbound and Outbound Variables" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus* and [Section 2.4, "Working with Proxy Service Message Flows."](#)

Securing Oracle Service Bus with Oracle Web Services Manager

Using Oracle Service Bus in conjunction with Oracle Web Services Manager provides scalable, standards-based, centrally managed approach to securing your SOA environment with WS-Security policies while leveraging your existing security providers.

Oracle Web Services Manager is a run-time framework for security policy creation, management, and governance. You create policies, attach them to services in Oracle Service Bus, and enforce those policies at various points in the messaging life cycle with Oracle Web Service Manager agents.

Note: In future releases of Oracle Service Bus, Oracle Web Services Manager policies will replace and enhance the functionality of WLS 9.2 security policies. While this version of Oracle Service Bus continues to support WLS 9.2 policies, you should consider using Oracle Web Services Manager policies for new service creation to prepare for the eventual deprecation of WLS 9.2 policy support.

This section includes the following topics:

- [Section 50.1, "About Oracle Web Services Manager Integration with Oracle Service Bus"](#)
- [Section 50.2, "Setting Up and Using Oracle Web Services Manager with Oracle Service Bus"](#)
- [Section 50.3, "Use Cases: Oracle Service Bus and WLS 9.2 Policies with Oracle Web Services Manager"](#)

For more information about Oracle Web Services Manager, see the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

50.1 About Oracle Web Services Manager Integration with Oracle Service Bus

Oracle Web Services Manager is a component of the Oracle Enterprise Manager Fusion Middleware Control, a run-time framework that provides centralized management and governance of Oracle SOA Suite environments and applications. You create and configure Oracle Web Services Manager policies in Oracle Enterprise Manager, and those policies are persisted in a policy store (a database is recommended). Oracle Web Services Manager lets you define policies against an LDAP directory and generate

standard security tokens (such as SAML tokens) to propagate identities across multiple Web services used in a single transaction.

In Oracle Service Bus, when defining a business or proxy service that lets you attach security policies, you can attach available "OWSM" policies.

Because Oracle Web Services Manager is a run-time component, attaching policies to Oracle Service Bus services requires a connection to an Oracle Service Bus domain that has Oracle Web Services Manager enabled. For example, when creating and managing services that use Oracle Web Services Manager policies in Eclipse, your Oracle Service Bus configuration must be deployed on an Oracle Web Services Manager-enabled domain to attach the policies. With no run-time connection to Oracle Web Services Manager from the development environment, you can only view or remove policies previously attached to services.

Oracle Web Services Manager support in Oracle Service Bus is not available automatically. Enable Oracle Web Services Manager support in Oracle Service Bus by selecting the "Oracle Service Bus OWSM Extension" template when you create or extend an Oracle Service Bus domain. Once Oracle Web Services Manager support is enabled in an Oracle Service Bus domain, you cannot disable it. See [Section 50.2.1, "Adding Oracle Web Services Manager and Oracle Enterprise Manager to an Oracle Service Bus Domain."](#)

50.1.1 Security Providers

This section describes the security services that Oracle Service Bus and Oracle Web Services Manager use for authentication and authorization.

Oracle Web Service Manager uses Java Platform Security (JPS), so Oracle Service Bus uses JPS providers for Oracle Web Services Manager policies. Oracle Service Bus also uses Oracle Common Security Services (CSS) for other aspects of message security.

The following points describe which security providers Oracle Service Bus and Oracle Web Services Manager use for different security areas.

50.1.1.1 JPS Providers

When using Oracle Web Services Manager policies:

- Oracle Web Services Manager policies use SAML providers from JPS and not from Oracle WebLogic Server. For information on configuring SAML for Oracle Web Services Manager policies, see [Section 50.2.3, "Configuring SAML."](#)
- For authentication, Oracle Web Services Manager uses the JPS Login Module, which in turn calls authentication providers configured on Oracle WebLogic Server.
- For Oracle Web Services Manager policies, a best practice is to configure the keystore on JPS, with both the Oracle WebLogic Server and the JPS keystore file referencing the same JKS file. For example, when a proxy service with WLS 9.2 policies routes to a business service that has Oracle Web Services Manager policies, the same keystore file should be referenced. For more information, see [Section 50.2.1, "Adding Oracle Web Services Manager and Oracle Enterprise Manager to an Oracle Service Bus Domain."](#)
- A JPS keystore serves as both a keystore and a truststore for Oracle Web Services Manager policies.

50.1.1.2 CSS Providers

Oracle Service Bus uses CSS providers in the following ways:

- To enforce WLS 9.2 policies
- To enforce transport security
- Oracle WebLogic Server authorization providers for authorization policies
- Custom Oracle WebLogic Server authentication providers and identity asserters for custom authentication policies
- Oracle WebLogic Server credential providers and mappers
- Oracle WebLogic Server keystore and truststore for WLS 9.2 policies
- Authentication and identity assertion through Oracle Web Services Manager agents

50.2 Setting Up and Using Oracle Web Services Manager with Oracle Service Bus

This section includes the following topics:

- [Section 50.2.1, "Adding Oracle Web Services Manager and Oracle Enterprise Manager to an Oracle Service Bus Domain"](#)
- [Section 50.2.2, "Attaching Oracle Web Services Manager Policies to Oracle Service Bus Services"](#)
- [Section 50.2.3, "Configuring SAML"](#)
- [Section 50.2.4, "Deployment Considerations"](#)
- [Section 50.2.5, "Auditing"](#)
- [Section 50.2.6, "Monitoring Statistics"](#)
- [Section 50.2.7, "Unsupported Assertions and Seed Policies"](#)

50.2.1 Adding Oracle Web Services Manager and Oracle Enterprise Manager to an Oracle Service Bus Domain

To use Oracle Web Services Manager policies in Oracle Service Bus, you must create the proper database schemas for the Oracle Web Services Manager policy store, then extend an Oracle Service Bus domain to include Oracle Web Services Manager.

Note: After you add Oracle Web Services Manager to an Oracle Service Bus domain, you cannot disable Oracle Web Services Manager in the domain.

1. Use the Oracle Repository Creation Utility (RCU) to create the Oracle Web Services Manager database schemas in a supported database. Select the following schemas to create:
 - **SOA Infrastructure**
 - **Metadata Services** and **AS Common Schemas** are automatically selected when you select SOA Infrastructure

Note: After you select SOA Infrastructure, you can optionally deselect **Business Activity Monitoring** and **User Messaging Services** if you do not want to enable reporting.

For more information on running RCU, see the *Oracle Fusion Middleware Repository Creation Utility User's Guide*.

2. Extend your Oracle Service Bus domain with Oracle Web Services Manager and Oracle Enterprise Manager. Select the following domain templates when running the Oracle Fusion Middleware Configuration Wizard:
 - **Oracle Service Bus OWSM Extension**
 - **Oracle WSM Policy Manager** (automatically selected when you select the OWSM Extension)
 - **Oracle Enterprise Manager** (optional, needed for creating and managing Oracle Web Services Manager policies)

As part of the domain extension, the Oracle Configuration Wizard creates an **OWSM MDS Schema** in the JDBC configuration window. Select the schema and set the database information based on the RCU settings used to create the Oracle Web Services Manager schemas in the previous step.

For more information, see "Creating a Domain" in the *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle SOA Suite*.

3. As a best practice, configure the keystore for Oracle Web Services Manager on JPS, with both the Oracle WebLogic Server and the JPS keystore file referencing the same JKS file. For example, when a proxy service with WLS 9.2 policies routes to a business service that has Oracle Web Services Manager policies, the same keystore file should be referenced.

For information on creating the keystore, see "Setting up the Keystore for Message Protection" in the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

After successful extension of the domain and creation of the keystore for Oracle Web Services Manager, you can create Oracle Web Services Manager policies using the Oracle Enterprise Manager Fusion Middleware Control and attach policies to services in Oracle Service Bus. Oracle Web Services Manager automatically provides commonly used policies.

With the domain running, you can access Oracle Enterprise Manager with the following URL:

http://host:port/em

For more information on managing Oracle Web Services Manager policies, see "Managing Web Service Policies" in the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

50.2.2 Attaching Oracle Web Services Manager Policies to Oracle Service Bus Services

After you extend your Oracle Service Bus domain to include Oracle Web Services Manager and create policies using Oracle Enterprise Manager, you can attach those policies to the following types of proxy and business services in Oracle Service Bus on the Policies page:

- WSDL

- Any SOAP

You can attach Oracle Web Services Manager policies only at the service level, and you cannot embed them in service WSDLs. For a given service, you must use either Oracle Web Services Manager policies or WLS 9.2 policies, but not both. You can, however, use one type of policy in a proxy service and another type in a corresponding business service.

In Eclipse, when adding Oracle Web Services Manager policies to services, you must be connected to a running domain that has Oracle Web Services Manager enabled. If you are not connected to a running server in the development environment, you can only view and remove previously added Oracle Web Services Manager policies, and Oracle Service Bus shows a warning that the Oracle Web Services Manager policies will be validated only on publish.

Note: When working with multiple servers in Eclipse, Eclipse chooses the first valid Oracle Service Bus server in the list of servers for retrieval of Oracle Web Services Manager policies.

When attaching policies in the development environment, keep in mind that services in the development environment can be out of sync with services in the Oracle Service Bus console, so take care when updating services from Eclipse to the console.

If you copy a service to create a same type of service (for example, copy a business service to create a new business service), be sure to review your Oracle Web Services Manager policies in the new service and make any necessary adjustments.

50.2.2.1 Policy Overrides

After adding Oracle Web Services Manager policies to a service, you can provide policy overrides on the Security page.

For the policies used, the user interface displays the override keys (properties) and their default values. The key names come from the policy binding. If allowed, a text box appears next to a key's default value where you can provide an override value.

Oracle Service Bus does not provide well-known keys for override, such as sign key alias or CSF key, which points to user credentials in a CSF store. (Oracle Service Bus provides user credentials in the service account.)

Override keys you provide are passed to the Oracle Web Service Manager agent during invocation.

50.2.3 Configuring SAML

Configuring SAML is different between WLS 9.2 policies and Oracle Web Services Manager policies. For information on configuring SAML with Oracle Web Services Manager, see "Configuring SAML" in the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

50.2.4 Deployment Considerations

When you export Oracle Service Bus configurations that contain services with Oracle Web Services Manager policy references, the references are maintained. You must ensure that the referenced policies also exist in the target environment. If the target environment is the IDE, warnings are displayed saying that policies will be validated on publish.

50.2.5 Auditing

To audit policy events in Oracle Enterprise Manager, you must set up an audit data repository and set up event collection. For more information, see the following topics in the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*:

- "Managing Audit Data Collection and Storage"
- "Viewing Audit Reports" – Pre-defined audit reports for Oracle Web Services Manager in Oracle Business Intelligence Publisher include statistics for Oracle Service Bus.

You can audit the following policy-level events:

- Policy creation, deletion, or modification
- Assertion template creation, deletion, or modification

50.2.6 Monitoring Statistics

For this release, Oracle Enterprise Manager policy monitoring statistics and usage/impact analysis for Oracle Service Bus are not available. Therefore, you are not able to see the impact of policy modifications on the services those policies are attached to. However, Oracle Service Bus collects WS-Security error statistics for Oracle Web Services Manager policy enforcement errors as it does for WLS 9.2 policies, and those statistics are available in the Oracle Service Bus service monitoring dashboard.

50.2.7 Unsupported Assertions and Seed Policies

Table 50–1 and Table 50–2 list the Oracle Web Services Manager assertions and seed policies that Oracle Service Bus does *not* support. Any assertions or seed policies not listed in tables is automatically supported, including user-defined assertions. The assertion or policy enabled/disabled option does not determine whether or not an assertion or policy is supported in Oracle Service Bus.

Table 50–1 Unsupported Oracle Web Services Manager Assertions

Assertion	Assertion Types
ExactlyOnce	N/A
binding-authorization	Authorization
binding-permission-authorization	Authorization
http-security	N/A
OptimizedMimeSerialization	MTOM
RMAssertion	Reliable Messaging
sca-component-authorization	Authorization
sca-component-permission-authorization	Authorization
UsingAddressing	N/A
wss-saml-token-bearer-over-ssl	Authentication
wss-saml-token-over-ssl	Authentication
wss-username-token-over-ssl	Authentication

Table 50–2 *Unsupported Oracle Web Services Manager Seed Policies*

Policy	Policy Type
component_authorization_denyall_policy	Security/Authorization
component_authorization_permitall_policy	Security/Authorization
component_permission_authorization_policy	Security/Authorization
wsaddr_policy	Addressing
wsmtom_policy	MTOM
warm10_policy	Reliable Messaging
warm11_policy	Reliable Messaging
Wss_oam_token_client_policy	Security/Authentication
Wss_oam_token_service_policy	Security/Authentication

Note: In the development environment, when using policies that are not one of the supported seed policies:

- An effective WSDL generated in the development environment will skip non-seed policies.
 - Validation is performed on service publish.
-

50.3 Use Cases: Oracle Service Bus and WLS 9.2 Policies with Oracle Web Services Manager

This section provides use cases that highlight the interaction between Oracle Service Bus services using WLS 9.2 policies and Oracle Web Services Manager features in providing security throughout the service pipeline.

When using Oracle Web Services Manager with WLS 9.2 policies in Oracle Service Bus services, no configuration is required, and you do not have to extend an Oracle Service Bus domain with Oracle Web Services Manager. You implement Oracle Web Services Manager features at the desired client and service locations, and the interaction and enforcement occurs automatically.

Note: In future releases of Oracle Service Bus, Oracle Web Services Manager policies will replace and enhance the functionality of WLS 9.2 security policies. While this version of Oracle Service Bus continues to support WLS 9.2 policies, you should consider using Oracle Web Services Manager policies for new service creation to prepare for the eventual deprecation of WLS 9.2 policy support.

For more information about Oracle Web Services Manager, see:

- *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*
Oracle Web Services Manager 10.1.3 documentation at
http://download.oracle.com/docs/cd/E10291_01/index.htm
- "Overview of Interoperability with Oracle Service Bus 10g Security Environments" in the *Oracle Fusion Middleware Interoperability Guide for Oracle Web Services Manager*

This document describes the following security use cases with Oracle Web Services Manager:

- [Section 50.3.1, "Message Protection"](#)
- [Section 50.3.2, "Authentication"](#)
- [Section 50.3.3, "Perimeter Security"](#)
- [Section 50.3.4, "Identity Propagation"](#)

Note: There is no equivalent of Gateway in Oracle Web Services Manager 11.1.1.

50.3.1 Message Protection

This section describes the following use cases:

- [Section 50.3.1.1, "Message Protection with Client Agent"](#)
- [Section 50.3.1.4, "Message Protection with Gateway"](#)
- [Section 50.3.1.2, "Message Protection with Server Agent"](#)
- [Section 50.3.1.3, "Message Protection with Client and Server Agents"](#)

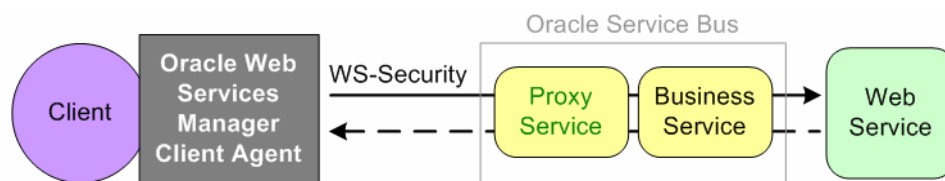
50.3.1.1 Message Protection with Client Agent

You can implement this use case with the following versions of Oracle Web Services Manager:

- 11.1.1.x
- 10.1.3.x

[Figure 50–1](#) illustrates using the Oracle Web Services Manager Client Agent for message protection.

Figure 50–1 Message Protection With an Oracle Web Services Manager Client Agent



The proxy service has an inbound message protection policy. The Oracle Web Services Manager Client Agent sends a signed and encrypted request to the proxy service. The proxy service receives the secured request and, acting as an active intermediary, decrypts and verifies signature and routes the request to the business service. The business service invokes the Web service, gets the response back, and sends it to the proxy service. The proxy service signs and encrypts the response and sends it to the Oracle Web Services Manager Client Agent. The Client Agent receives the secure response, decrypts and verifies the signature, and passes the response to the client.

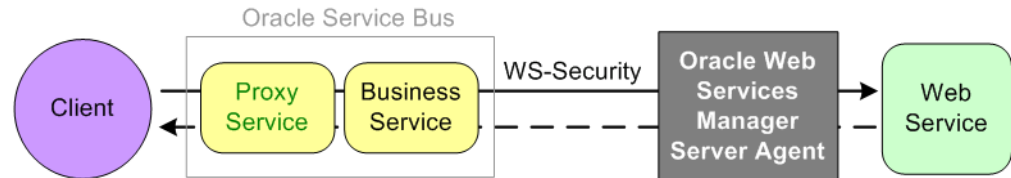
50.3.1.2 Message Protection with Server Agent

You can implement this use case with the following versions of Oracle Web Services Manager:

- 11.1.1.x
- 10.1.3.x

Figure 50–2 illustrates using the Oracle Web Services Manager Server Agent for message protection.

Figure 50–2 Message Protection With an Oracle Web Services Manager Server Agent



The client sends a plain request through the proxy and business services in Oracle Service Bus. The business service signs and encrypts the request and sends the message to the Oracle Web Services Manager Server Agent. The Server Agent decrypts and verifies the request. The plain message response is passed back to the client.

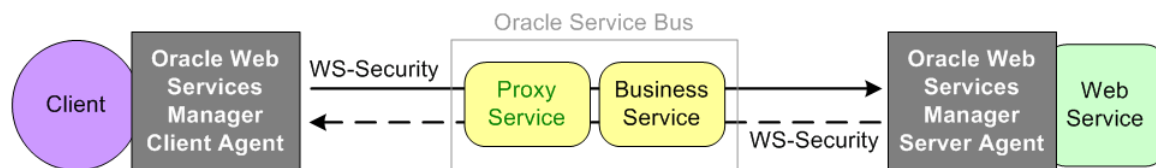
50.3.1.3 Message Protection with Client and Server Agents

You can implement this use case with the following versions of Oracle Web Services Manager:

- 11.1.1.x
- 10.1.3.x

Figure 50–3 illustrates using the Oracle Web Services Manager Client and Server Agents for message protection.

Figure 50–3 Message Protection With an Oracle Web Services Manager Client and Server Agents



The Oracle Web Services Manager Client Agent signs and encrypts a client request and sends the request through to the proxy service. The proxy service decrypts and verifies the signature and passes the request to the business service, which signs and encrypts the request. The Web service has a Server Agent injected in it. The Server Agent has an inbound message protection policy that decrypts and verifies the signature, then signs and encrypts the response. The response is sent back to the business service, which verifies the message and passes the response to the proxy service. The proxy service generates a signed and encrypted response and sends it to the Client Agent. The Client Agent decrypts and verifies the response, then returns the plain response to the client.

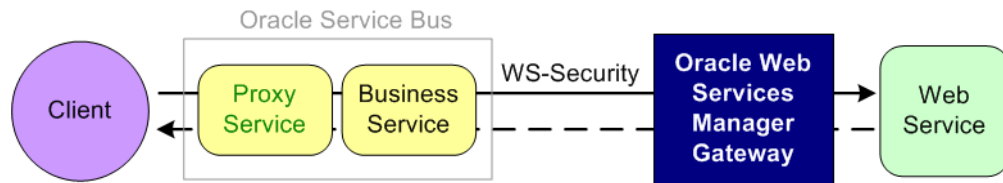
50.3.1.4 Message Protection with Gateway

You can implement this use case with the following versions of Oracle Web Services Manager:

- 10.1.3.x

Figure 50–4 illustrates using the Oracle Web Services Manager Gateway for message protection.

Figure 50–4 Message Protection With an Oracle Web Services Manager Gateway



The client sends a plain request through the proxy and business services in Oracle Service Bus. The business service signs and encrypts the request and sends the message to the Oracle Web Services Manager Gateway. The Gateway decrypts and verifies the request. The plain message response is passed back to the client.

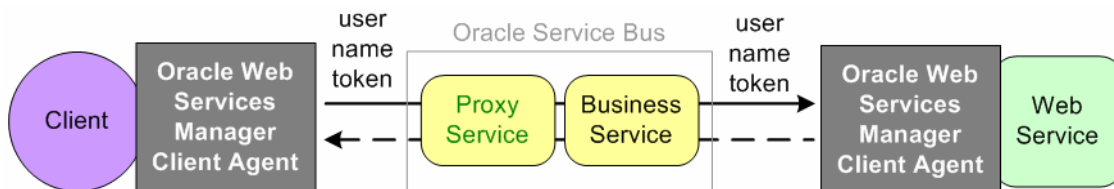
50.3.2 Authentication

You can implement this use case with the following versions of Oracle Web Services Manager:

- 11.1.1.x
- 10.1.3.x

Figure 50–5 illustrates using the Oracle Web Services Manager Client Agent for authentication.

Figure 50–5 Authentication with an Oracle Web Services Manager Client Agent



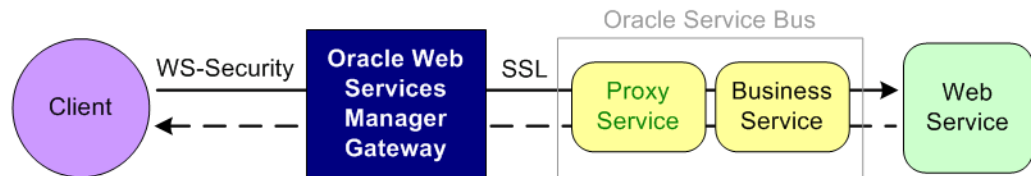
The proxy service has a user name token policy. The client, through Oracle Web Services Manager Client Agent, sends a request to the proxy service with user credentials at the message level in a user name token. The proxy service maps the user credential from the user name token using credential mapping and sends it through the business service to the Web service for authentication. The Web service is protected using an Oracle Web Services Manager service agent with an inbound user name token policy. The Oracle Web Services Manager Service Client Agent extracts and authenticates the user credentials. The response is then sent back through the business service and the proxy service to the client.

50.3.3 Perimeter Security

You can implement this use case with the following versions of Oracle Web Services Manager:

- 10.1.3.x

Figure 50–6 illustrates using Oracle Web Services Manager Gateway for enforcing perimeter security.

Figure 50–6 Perimeter Security with Oracle Web Services Manager Gateway

Oracle Web Services Manager Gateway virtualizes the service exposed by the Oracle Service Bus proxy service. The inbound request to the Oracle Web Services Manager Gateway has a message protection policy. The client sends a secure request to the Oracle Web Services Manager Gateway virtualized service, which is signed and encrypted.

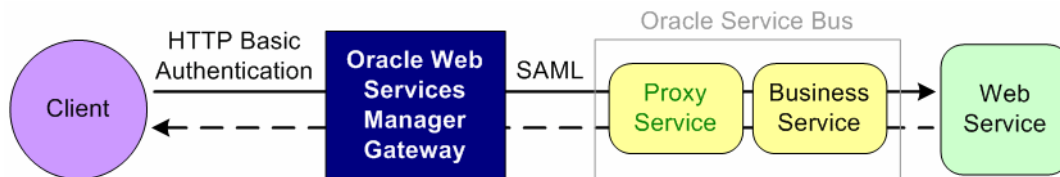
The Oracle Web Services Manager Gateway acts as a security enforcement point and decrypts and verifies the signature. Oracle Web Services Manager Gateway then routes the plain request to the proxy service over SSL. The proxy service forwards the request to the business service, which invokes the Web service and gets the plain response back. The response moves back through the proxy service and Oracle Web Services Manager Gateway to the client.

50.3.4 Identity Propagation

You can implement this use case with the following versions of Oracle Web Services Manager:

- 10.1.3.x

[Figure 50–7](#) illustrates using the Oracle Web Services Manager Gateway for identity propagation using SAML (Security Assertion Markup Language) token version 1.1.

Figure 50–7 Identity Propagation with Oracle Web Services Manager Gateway

The client sends a basic HTTP authentication request to the Oracle Web Services Manager Gateway. Oracle Web Services Manager Gateway authenticates the user using the user name and password from the HTTP header. Oracle Web Services Manager Gateway generates a SAML sender voucher assertion with the authenticated user identity (token mediation), inserts the SAML token, and sends the assertion to the proxy service. The proxy service receives the SAML assertion with the user identity and, acting as an active intermediary, verifies the user identity. The proxy service then passes the request to the business service. The response travels back through the business service, proxy service, and Oracle Web Services Manager Gateway to the client.

Using WS-Policy in Oracle Service Bus Proxy and Business Services

Note: This chapter applies only to WLS 9.2 policies and not Oracle Web Services Manager policies. In future releases of Oracle Service Bus, Oracle Web Services Manager policies will replace and enhance the functionality of WLS 9.2 security policies. While this version of Oracle Service Bus continues to support WLS 9.2 policies, you should consider using Oracle Web Services Manager policies for new service creation to prepare for the eventual deprecation of WLS 9.2 policy support.

To express the message-level security requirements for a proxy service or business service that is a Web service, you use the Web Services Policy (WS-Policy) framework.

This chapter describes conceptual information that you will need in the next chapter, [Chapter 52, "Configuring Message-Level Security for Web Services."](#)

The following sections describe configuring WS-Policy for proxy services and business services:

- [Section 51.1, "About Web Services Policy"](#)
- [Section 51.2, "Oracle Service Bus WS-Policy Files"](#)
- [Section 51.3, "Creating and Using Custom WS-Policy Statements"](#)
- [Section 51.4, "Attaching WS-Policy Statements to WSDL Documents"](#)
- [Section 51.5, "Oracle-Proprietary Security Policy Best Practices"](#)
- [Section 51.6, "Policy Subjects and Effective Policy"](#)

51.1 About Web Services Policy

Web Services Policy (WS-Policy) is a standards-based framework for defining a Web service's constraints and requirements. It expresses constraints and requirements in a collection of XML statements called policies, each of which contains one or more assertions.

In Oracle Service Bus, WS-Policy assertions are used to specify a Web service's requirements for digital signatures and encryption, along with the security algorithms and authentication mechanisms that it requires.

The WS-Policy framework allows other specifications to declare "policy assertions." These are domain-specific XML elements that appear inside a `<policy>` element.

Policy assertions specifications describe the syntax and semantics of these domain-specific assertions.

WS-SecurityPolicy is one example of a domain-specific assertion language. The WS-SecurityPolicy specification defines a set of security policy assertions for use with the WS-Policy framework.

WS-ReliableMessaging is another example of a domain-specific assertion language; it defines assertions for declaring reliable-messaging policy.

51.1.1 Relationship Between WS-Security and WS-Policy

Web Services Security (WS-Security) works in conjunction with the Web Services Policy Framework (WS-Policy), and it is important that you understand what these terms mean and how they relate:

- Web Services Security (WS-Security) is an OASIS standard that defines interoperable mechanisms to incorporate message-level security into SOAP messages. WS-Security determines "how" message-level security is incorporated into SOAP messages.

WS-Security supports message integrity and message confidentiality. It also defines an extensible model for including security tokens in a SOAP envelope and a model for referencing security tokens from within a SOAP envelope. WS-Security allows you to specify which parts of a SOAP message are digitally signed or encrypted.

- The Web Services Policy Framework (WS-Policy) provides a general-purpose model and corresponding syntax to describe and communicate the policies of a Web service. WS-Policy is an abstract XML framework. The interesting aspects of a WS-Policy are defined in child elements called policy "assertions."
- WS-SecurityPolicy defines assertions for specifying the security aspects of a WS-Policy. WS-SecurityPolicy determines "what" message-level security is required of SOAP messages.

The policies can determine which operations are secured and which security measures a Web services client must apply.

When you configure the WS-Policy of a proxy or business service, if the WS-Policy contains one or more security policy assertions, then the proxy service or business service is considered to be WS-Security enabled.

51.1.2 WS-Policies Can be Bound Directly to Service

As in prior releases of Oracle Service Bus, WS-Policy policies can be included directly in a WSDL document or included by reference, and a WSDL document may import other WSDL documents that contain or refer to WS-Policy policies. An XML file that contains these policies can be used by multiple proxy services or business services.

In addition, there is an alternative way to bind WS-Policy to services. The Policies console page allows you to bind policies directly to a service. Policies can be bound to different scopes:

- The entire service
- A service operation
- The request message of a service operation
- The response message of a service operation

If a policy is bound to the entire service, it applies to all operations in the service and all request and response messages of all operations. If a policy is bound to an operation, the policy applies to the request and response message of that operation.

Any number of policies can be bound on any given scope.

For the purpose of example, assume there is a service *S* with operations *A*, *B*, *C* and *D*, where *A*, *B* and *C* are request/response operations and *D* is a request-only operation. An administrator can configure the following WS-Policy bindings:

- Policy *X* bound to the entire service *S*,
- Policies *Y* and *Z* on operation *A*
- Policies *Y* and *Z* on operation *B*
- Policy *P* on the request message of operation *C*
- Policy *Q* on the response message of operation *C*
- Policy *R* on the request message of operation *D*

In this example:

- The effective policy of the request/response messages of operations *A* and *B* is the union of policies *X*, *Y* and *Z*.
- The effective policy on the request message of operation *C* is the union of *X* and *P*. The effective policy on the response message of operation *C* is the union of *X* and *Q*.
- The effective policy on the request message of operation *D* is the union of *X* and *R*.

51.1.3 Abstract and Concrete WS-Policy Statements

For security policy assertions written under the WS-Policy specification (using the proprietary Oracle schema for security policy), the WebLogic Web Services runtime environment recognizes two types of WS-Policy statements:

- **Concrete** WS-Policy statements specify the security tokens that are used for authentication, encryption, and digital signatures. A concrete encryption policy always has the server's encryption certificate embedded in the form of a base-64 encoded certificate in an X.509 binary security token.

You can create concrete WS-Policy statements if you know at design time the type of authentication (such as using X.509 or SAML tokens) that you want to require.

- **Abstract** WS-Policy statements do not specify security tokens. Specifically, this means the `<Identity>` and `<Integrity>` elements (or assertions) of the WS-Policy files do not contain a `<SupportedTokens><SecurityToken>` child element, and the `<Confidentiality>` element WS-Policy file does not contain a `<KeyInfo><SecurityToken>` child element.

The Oracle Service Bus runtime environment determines which security token types an abstract policy will accept.

51.2 Oracle Service Bus WS-Policy Files

Oracle Service Bus includes a set of out-of-the-box WS-Policy files that you can use. (The Oracle Service Bus policy files are a subset of the policy files that Oracle WebLogic Server provides.)

The policy statements are of three types:

- WS-Security Policy assertions
- Oracle security policy assertions
- Reliable-messaging assertions

The predefined policy files are described in the sections that follow.

51.2.1 Predefined Oracle Proprietary Policy Files

The following Oracle proprietary predefined policy files are available:

- `Auth.xml`—contains a policy that requires Web service clients to authenticate. Oracle recommends that you do not use the `Auth.xml` policy file: use the `Sign.xml` and `Encrypt.xml` policies whenever possible.
- `Encrypt.xml`—contains a policy that requires clients to encrypt the SOAP body with 3DES-CBC. The key wrapping algorithm is RSA 1.5. A symmetric key for Triple DES (Data Encryption Standard) is generated by the client and encrypted for the recipient with RSA 1.5.

You cannot use this policy with a business service. Instead, create your own concrete encryption policy. See [Section 51.3, "Creating and Using Custom WS-Policy Statements."](#)

- `Sign.xml`—contains a policy that requires clients to sign the SOAP body. It also requires that the WS-Security engine on the client add a signed timestamp to the `wsse:Security` header—which prevents certain replay attacks. All system headers are also signed. The digital signature algorithm is RSA-SHA1. Exclusive XML canonicalization is used.

The system headers are:

- `wsrn:SequenceAcknowledgement`
- `wsrn:AckRequested`
- `wsrn:Sequence`
- `wsa:Action`
- `wsa:From`
- `wsa:To`
- `wsa:FaultTo`
- `wsa:MessageID`
- `wsa:RelatesTo`
- `wsa:ReplyTo`
- `wsu:Timestamp`
- `wsax:SetCookie`

The name space prefixes correspond to the name spaces in the following table:

Prefix	Name Space
wsrn	<code>http://schemas.xmlsoap.org/ws/2005/02/rm</code>
wsa	<code>http://schemas.xmlsoap.org/ws/2004/08/addressing</code>
wsu	<code>http://schemas.xmlsoap.org/ws/2002/07/utility</code>

Prefix	Name Space
wsax	http://schemas.xmlsoap.org/ws/2004/01/addressingx

51.2.2 Predefined Reliable Messaging Policy Files

WebLogic Web Services use WS-Policy files to enable a destination endpoint to describe and advertise its Web Service reliable messaging capabilities and requirements. These WS-Policy files are XML files that describe features such as the version of the supported WS-ReliableMessaging specification, the source endpoint's retransmission interval, the destination endpoint's acknowledgment interval, and so on.

Oracle Service Bus includes two simple reliable messaging WS-Policy files that you can use (only with the WS-RM transport) if you do not want to create your own WS-Policy files:

- `DefaultReliability.xml`—Specifies typical values for the reliable messaging policy assertions, such as inactivity timeout of 10 minutes, acknowledgement interval of 200 milliseconds, and base re-transmission interval of 3 seconds. See `DefaultReliability.xml` WS-Policy File for the actual WS-Policy file.
- `LongRunningReliability.xml`—Similar to the preceding default reliable messaging WS-Policy file, except that it specifies a much longer activity timeout interval (24 hours.)

51.2.3 When to use the Predefined Policy Files

Oracle recommends that you use these pre-packaged policies whenever possible. However, you cannot use them under the following conditions:

- Use transport-level policies only where message-level security is not required.
- If you need to specify that particular parts of the body of a SOAP message are encrypted or digitally signed, rather than the entire body, you cannot use the Oracle Service Bus WS-Policy statements.

Instead, create custom WS-Policy statements. See [Section 52.5.1, "Example: Encrypting Part of the SOAP Body and Header."](#)

- If you require clients to provide SAML tokens, you cannot use the Oracle Service Bus WS-Policy statements. WS-Policy statements that require SAML tokens must specify the `confirmationMethod` and therefore must be concrete.
- If you want a **business service** to require encryption, you cannot use the Oracle Service Bus `Encrypt.xml` policy. Business services require concrete encryption policies (the certificate must be embedded in the policy).

For information on using these policies in your proxy services or business services, see [Section 51.4, "Attaching WS-Policy Statements to WSDL Documents."](#)

51.3 Creating and Using Custom WS-Policy Statements

If the Oracle Service Bus WS-Policy packaged policy files do not meet your security needs, you can write your own WS-Policy statements. You cannot modify the Oracle Service Bus WS-Policy statements.

You can write custom WS-Policy statements directly in your Web service's WSDL document. Or, if you want to reuse your statements in multiple Web services, write them in a separate XML file and then:

- Import them to Oracle Service Bus and refer to them from the WSDL documents.
- Directly bind them to a service

Note the following restrictions for WS-Policy statements in Oracle Service Bus:

- Security policy files written under the WS-Policy specification using the proprietary Oracle schema for security policy are required to have an `Id` attribute from the following name space:
<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd>
 The value of this attribute must be unique across all WS-Policy statements in the Oracle Service Bus domain. This attribute is optional in the WS-Policy schema but required in an Oracle Service Bus Web service.
- If you create a confidentiality assertion in a proxy service, it must be abstract (the certificate must not be embedded in the policy). You will get error messages while creating a proxy service that contains a concrete confidentiality assertion.
- If you create a confidentiality assertion in a business service, it must be concrete (the certificate must be embedded in the policy) and it must be located directly in the WSDL document. You cannot attach such a policy by reference. See [Section 52.5.1, "Example: Encrypting Part of the SOAP Body and Header."](#)

51.4 Attaching WS-Policy Statements to WSDL Documents

Oracle Service Bus implements the WS-Policy Attachment specification (<http://www.w3.org/Submission/WS-PolicyAttachment/>), which defines the mechanisms for associating WS-Policy statements with Web services.

To attach WS-Policy statements to a WSDL document for a Web service:

1. If you created a custom WS-Policy in a separate XML file, add the custom WS-Policy file as a resource in the Oracle Service Bus domain. See "Adding Custom WS-Policies" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.
2. In the `<definitions>` element of the WSDL document, add the following child element:

```
<wsp:UsingPolicy
  wsdl:Required="true"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" />
```

The `wsdl:required="true"` attribute ensures that proxy services and business services are capable of processing the policy attachments.

If you do not add this element, Oracle Service Bus ignores any WS-Policy statements in the WSDL.

3. Within each element in the WSDL document that you want to secure:
 - a. Determine the URI of the WS-Policy statements that you want to use. See [Section 51.4.1, "Determining the URI of a WS-Policy Statement."](#)
 - b. Specify the URI in the WSDL document. See [Section 51.4.2, "Specifying the URI of a WS-Policy Statement in a WSDL Document."](#)

51.4.1 Determining the URI of a WS-Policy Statement

For the Oracle Service Bus WS-Policy statements, the URIs are always as follows:

- policy:Auth.xml
- policy:Encrypt.xml
- policy:Sign.xml

For WS-Policy statements that are located directly in the WSDL document, the URI is as follows: `#policy-ID` where `policy-ID` is the value of the policy's `wsu:ID` attribute. See [Example 51-2](#).

For WS-Policy statements that you created in a separate XML file and added as resources to Oracle Service Bus, the URI is as follows: `policy:policy-ID` where `policy-ID` is the value of the policy's `wsu:ID` attribute (which you specified in the policy's XML file).

You can also use UDDI to attach WS-Policy statements to a WSDL document, in which case the URI is expressed differently. For more information, see the WS-Policy Attachment specification

(<http://www.w3.org/Submission/WS-PolicyAttachment/>).

51.4.2 Specifying the URI of a WS-Policy Statement in a WSDL Document

Use one of the following techniques to specify the URI in a WSDL document:

- `PolicyURIs` attribute

If the WSDL schema (described in <http://www.w3.org/TR/wsdl>) allows attribute extensibility for the element that you want secure, add the `PolicyURIs` global attribute to the element.

For the value of this element, specify a list of URIs, each of which refers to a single policy.

For example:

```
<input message="tns:foo" wsp:PolicyURIs="policy:Sign.xml"/>
```

- Nested `<Policy>` element

If the WSDL schema allows element extensibility for the element that you want to secure, add `<Policy>` as a global child element. For each WS-Policy that you want to use, add one `<PolicyReference>` element as a child of the `<Policy>` element.

For each `<PolicyReference>` element, include a URI attribute that refers to a single policy. You can also include a digest and digest algorithm in the element.

For example:

```
<wsp:Policy>
  <wsp:PolicyReference URI="policy:Sign.xml"/>
</wsp:Policy>
```

[Table 51-1](#) lists the XPath name of WSDL elements and the technique that you use to specify the URI of the WS-Policy statement. The table also indicates the WSDL elements for which Oracle Service Bus does not support the attachment of WS-Policy statements.

Table 51-1 WSDL Elements That Can Be Protected in Oracle Service Bus

To Attach a Policy to This WSDL Element...	Use This Technique...
/definitions/message	Nested <code><Policy></code> element

Table 51–1 (Cont.) WSDL Elements That Can Be Protected in Oracle Service Bus

To Attach a Policy to This WSDL Element...	Use This Technique...
/definitions/message/part	PolicyURIs attribute
/definitions/portType	PolicyURIs attribute
/definitions/portType/operation	Nested <Policy> element
/definitions/portType/operation/input	PolicyURIs attribute
/definitions/portType/operation/output	PolicyURIs attribute
/definitions/portType/operation/fault	Oracle Service Bus does not support attaching WS-Policy statements to this element
/definitions/binding	Nested <Policy> element
/definitions/binding/operation	Nested <Policy> element
/definitions/binding/operation/input	Nested <Policy> element
/definitions/binding/operation/output	Nested <Policy> element
/definitions/binding/operation/fault	Oracle Service Bus does not support attaching WS-Policy statements to this element
/definitions/binding/service	Oracle Service Bus does not support attaching WS-Policy statements to this element
/definitions/service/port	Nested <Policy> element

51.4.3 Best Practices: Attaching WS-Policy Statements

Oracle recommends that you attach WS-Policy statements to any of the following elements or its descendants:

- portType
- binding

Oracle recommends that you do not attach WS-Policy statements to the following elements:

- service
- port
- message or message/part

51.4.4 Example: Requiring X.509 Credentials for Identity and Confidentiality

If a WS-Policy statement requires an X.509 token for authentication, it must also require a digital signature. An X.509 token cannot satisfy an identity assertion unless the client also signs some content with the corresponding private key.

To create a proxy service that requires clients to use X.509 certificates for authentication and digital signatures, you can do the following:

1. In the WSDL document that you will use to create a proxy service, attach the Oracle Service Bus policies that are in the `Sign.xml` and `Auth.xml` files. See [Example 51–1](#).

2. Configure the proxy service to use a service key provider that contains an X.509 certificate for digital signatures. See "Service Key Providers" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

Because the Oracle Service Bus `Sign.xml` and `Auth.xml` policies are abstract, they will require the client to provide the credentials that are specified in the service key provider that is associated with the proxy service.

[Example 51–1](#) shows a WSDL with references to the Oracle Service Bus `Sign.xml` and `Auth.xml` policies.

Example 51–1 WSDL with Policy References to Oracle Service Bus WS-Policies

```
<definitions
  ...
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401
    -wss-wssecurity-utility-1.0.xsd">
  <wsp:UsingPolicy
    wsdl:Required="true"
    xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/" />
  ...
  <portType name="Sample">
    <operation name="doFoo" parameterOrder="data">
      <input message="tns:foo" wsp:PolicyURIs="policy:Sign.xml" />
      <output message="tns:fooResponse" />
    </operation>
  </portType>
  <binding name="SampleBinding" type="tns:Sample">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    <operation name="doFoo">
      <wsp:Policy>
        <wsp:PolicyReference URI="policy:Sign.xml" />
        <wsp:PolicyReference URI="policy:Auth.xml" />
      </wsp:Policy>
      ...
    </operation>
  </binding>
  ...
</definitions>
```

51.4.5 Example: Attaching Custom Inline WS-Policy Statements to a WSDL Document

[Example 51–2](#) shows a WSDL with two custom WS-Policy policies, `wsu:Id="policy1"` and `wsu:Id="policy2"`. The policies are located in the WSDL document; therefore the URIs that refer to these policies use XML fragments.

Example 51–2 WSDL with Policy References to a Custom Inline Policy

```
<definitions
  ...
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
    200401-wss-wssecurity-utility-1.0.xsd">
  <wsp:UsingPolicy
    wsdl:Required="true"
    xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/" />
  <wsp:Policy wsu:Id="policy1">...</wsp:Policy>
  <wsp:Policy wsu:Id="policy2">...</wsp:Policy>
  ...
</definitions>
```

```
<portType name="Sample">
  <operation name="doFoo" parameterOrder="data">
    <input message="tns:foo" wsp:PolicyURIs="#policy1"/>
    <output message="tns:fooResponse"/>
  </operation>
</portType>
<binding name="SampleBinding" type="tns:Sample">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="doFoo">
    <wsp:Policy>
      <wsp:PolicyReference URI="#policy2"/>
    </wsp:Policy>
    <soap:operation
      soapAction="http://com.bea.samples/sample/doFoo"
      style="document"/>
    <input>
      <soap:body namespace="http://com.bea.samples/sample"
        use="literal"/>
    </input>
    <output>
      <soap:body namespace="http://com.bea.samples/sample"
        use="literal"/>
    </output>
  </operation>
</binding>
...
</definitions>
```

51.5 Oracle-Proprietary Security Policy Best Practices

This section describes best practices you should follow when using security policy assertions written under the WS-Policy specification, using the proprietary Oracle schema for security policy.

Note: Carefully analyze your security requirements before you design your WS-SecurityPolicy. These best practices may or may not apply to your specific business security needs.

- Make sure you do not use Identity assertions on an operation's response policy. As a corollary, do not use the predefined `Auth.xml` policy in a response policy.
When using WS-Security username tokens on inbound to an active intermediary proxy service, if you want to pass the username/password to a back-end service (username/password pass-through), the username token must include the password in clear-text.
- Whenever using WS-Security username tokens with clear-text passwords, it is strongly recommended that you protect the confidentiality of the username token, either by encrypting the entire token (with WS-Security) or by sending the message over SSL.
- Whenever using an Identity assertion, you may also want to use an Integrity assertion to digitally sign the authentication token (username, X.509 or SAML token) together with sensitive message content (SOAP body and/or SOAP header parts). The digital signature protects the integrity of the signed content and binds together the authentication token and message content. This is important to prevent someone from copying the authentication token into an arbitrary SOAP

envelope, thus forging a message. (You can also send the message over SSL instead of using an integrity assertion.)

- When using an Integrity assertion, it is recommended that you also use a MessageAge assertion. Furthermore, it is recommended that you include the signing token (that is, the verification certificate) in the wsse:Security header and that the digital signature covers the signing token and the timestamp, in addition to whatever SOAP body and/or SOAP header parts you wish to sign. The message age assertion guarantees a timestamp will be included in the security header. The timestamp is used to prevent some replay attacks. The predefined `Sign.xml` policy follows this best practice.
- When using timestamps over JMS (MessageAge assertions), make sure you set the age of the MessageAge assertion appropriately. If the value is too low, the message may expire while on the queue.
- Whenever an Identity assertion includes X.509 tokens in the supported token list, your policy must also have an Integrity assertion. The server will not accept X.509 tokens as proof of authentication unless the token is also used in a digital signature.

If the Identity assertion accepts other token types, you may use the `X509AuthConditional` attribute of the Integrity assertion to specify that the digital signature is required only when the actual authentication token is an X.509 token. Remember that abstract Identity assertions are pre-processed at deploy time and converted into concrete assertions by inserting a list of all token types supported by your runtime environment.

- Oracle recommends that you do not use abstract Identity assertions in your policy. It is preferable instead to directly specify exactly which token types are supported for authentication. Furthermore, Oracle recommends that your Identity assertion supports only one token type.

Note: This makes the `X509AuthConditional` attribute of Integrity assertions unnecessary, as there is no ambiguity as to which token types are supported.

As a corollary, Oracle recommends that you do not use the `Auth.xml` policy file; use the `Sign.xml` and `Encrypt.xml` policies whenever possible.

- Whenever an Oracle Service Bus proxy processes digital signatures (on inbound request messages or back-end response messages), it is strongly recommended that you configure a certificate registry in your security realm and import your trading partner certificates in the registry.

51.6 Policy Subjects and Effective Policy

A **policy subject** is an entity, such as service, endpoint, operation, or message, with which a policy can be associated. You can associate a single WS-Policy statement with multiple policy subjects; conversely, multiple WS-Policy statements can be associated with a single policy subject. A **policy scope** is the collection of policy subjects to which a policy applies. For example, the policy scope implied by a policy attached to `wsdl:binding/wsdl:operation/wsdl:input` is the input message, the operation, the endpoint, and the service.

The **effective policy** for a given policy subject is the merge of all policies whose scopes contain that policy subject. For example, the effective policy of the input message of a binding operation is the merge of all policies attached to the following:

- The input message of the binding operation
- The binding operation
- The binding
- The input message of the port-type operation
- The port-type operation
- The port-type
- The service

The Oracle Service Bus Console displays the effective policy (read only) when configuring a business or proxy service with WS-Policy statements.

Configuring Message-Level Security for Web Services

Message-level security applies security checks to a SOAP message after a Web services client establishes a connection with an Oracle Service Bus proxy service or business service and before the proxy service or business service processes the message.

Message-level security is categorized as follows:

- **Inbound** message-level security applies to messages between clients and Oracle Service Bus proxy services. It applies security to both the request from the client and the response message back to the client.

You can think of this as proxy service security.

- **Outbound** message-level security applies to messages between Oracle Service Bus proxy services and SOAP-HTTP or SOAP-JMS business services. It applies security to both the request and the response.

You can think of this as business service security.

The following sections describe configuring message-level security for a proxy service or a business service:

- [Section 52.1, "About Message-Level Security"](#)
- [Section 52.2, "Message-Level Access Control Policies for Proxy Services"](#)
- [Section 52.3, "Configuring Proxy Service Message-Level Security"](#)
- [Section 52.4, "Configuring Business Service Message-Level Security: Main Steps"](#)
- [Section 52.5, "Examples of Custom WS-Policy Statements"](#)
- [Section 52.6, "Disabling Business Service Message-Level Security"](#)

Note: The implementation of message-level security includes proxy services that have been configured with message-level custom authentication (either custom token or username/password).

The message-level security mechanisms described in this section work alone or in concert with the message-level custom authentication mechanism, which is described in [Chapter 54, "Configuring Custom Authentication."](#) See [Section 54.9, "Combining WS-Security with Custom Username/Password and Tokens"](#) for information about using both types of security.

52.1 About Message-Level Security

Oracle Service Bus supports message-level security for SOAP messages that are sent over the HTTP (including HTTPS) or JMS protocols. Usually you use message-level security in addition to the transport-level security that these protocols offer. You can require Web services clients to provide credentials at the transport level, the message level, or both levels. If you require clients to provide credentials at both levels, Oracle Service Bus uses the message-level credentials for proxy service authentication and authorization.

To express the message-level security requirements for a proxy service or business service that is a Web service, you use the Web Services Policy (WS-Policy) framework. The Web Services Policy (WS-Policy) framework is described in [Chapter 52, "Configuring Message-Level Security for Web Services."](#)

With message-level security, a proxy service or business service specifies which of its operations are secured and which of the following security measures a Web services client must apply to its SOAP messages, which contain requests to invoke operations:

- **Authentication**
Requires a client to present an identity that can be compared with user accounts in the domain's authentication provider.
- **Message integrity through digital signatures**
Establishes the identity of the client that is requesting to invoke an operation and guarantees that no intermediary has altered the request. Also guarantees that the return values of the operation are returned to the client without being altered by an intermediary.
- **Message confidentiality through XML encryption**
Encrypts the request and the return value in the response and guarantees that no intermediary has viewed the request or the response.

All of these security measures require a client to encode security tokens in its SOAP messages, and the proxy service or business service specifies which types of security tokens it requires to be encoded in the SOAP messages.

52.1.1 Sample Sequence of Actions in Message-Level Security

To send a SOAP message to a proxy service that requires message-level security, the following actions occur:

1. A Web services client generates a SOAP header and adds the header to the SOAP message envelope. The header includes digital signatures, security tokens, and other constructs.
2. When the proxy service processes the secured envelope, it decrypts the message, which removes the security header.
3. The proxy service then verifies that the message conforms to its security requirements. For example, the proxy service confirms that the required message parts were signed and/or encrypted and that the required tokens are present with the required claims.
4. The entire process is repeated in reverse for the response from the proxy service to the client.

52.2 Message-Level Access Control Policies for Proxy Services

While message integrity and message confidentiality guarantee that intermediaries do not view or modify messages, and while message authentication requires clients to prove that they are known users, they do nothing to specify which known users are allowed (authorized) to invoke proxy service operations.

To limit access to authorized users, you use the Oracle Service Bus Console to create message-level access control policies. These policies allow a proxy service to process only those SOAP messages from authorized clients.

52.3 Configuring Proxy Service Message-Level Security

You can configure a proxy service to support one of the following techniques for inbound message-level security:

- **Active-Intermediary**

The proxy service processes the header in the client's SOAP messages and enforces the message-level access control policy on the messages.

For example, a client encrypts and signs its SOAP message and sends it to a proxy service. The proxy service decrypts the message and verifies the digital signature, then routes the message. Before the proxy service sends the response back to the client, the proxy service signs and encrypts the message. The client then decrypts the message and verifies the proxy service's digital signature.

- **Pass-Through**

Instead of processing the header in the client's SOAP messages, the proxy service passes the message untouched to a business service. Although the proxy service does not process the secured sections of the SOAP message, it can route the message based on values in the header. When the business service receives the message, it processes the security header and acts on the request. Note that the business service must use the Web Services Policy (WS-Policy) framework to describe which of its operations are secured with message-level security. The business service sends its response to the proxy service, and the proxy service passes the response untouched to the client.

For example, the client encrypts and signs the message and sends it to the proxy service. The proxy service does not decrypt the message or verify the digital signature; it simply routes the message to the business service. The business service decrypts the messages and verifies the digital signature, and then processes the request. The response path is similar.

52.3.1 Creating an Active Intermediary Proxy Service: Main Steps

To create a proxy service to act as an active intermediary:

1. In a text editor or IDE, create a WSDL document to define the proxy service:
 - If you plan to bind the policies directly from the console, the WSDL does not need to have policy statements.
 - If you want the policy to be WSDL-based, attach one or more Web Services Policy (WS-Policy) statements to the WSDL document, including one or more of the predefined policies.
2. In the Oracle Service Bus Console, import the WSDL document into the Oracle Service Bus WSDL repository and resolve any WSDL dependencies.

See "Adding WSDLs" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

3. If you have not already configured the WebLogic security framework to support Oracle Service Bus, do one or more of the following depending on whether the WS-Policy of any of the operations in the proxy service contains security policy assertions that secure **requests** from clients to the proxy service:
 - If you want operation request policies to require authentication with a WS-Security X.509 certificate token, configure the Web Service security configuration named `__SERVICE_BUS_INBOUND_WEB_SERVICE_SECURITY_MBEAN__`. See [Section 45.7, "Configuring the Oracle WebLogic Security Framework: Main Steps."](#)
 - If you want operation request policies to require authentication with a WS-Security Username/Password token with password digest, make sure to enable password digests. See [Section 45.7, "Configuring the Oracle WebLogic Security Framework: Main Steps."](#)
 - If you want operation request policies to require the use of SAML tokens, you must configure a SAML asserting party for this proxy service. See [Section 53.3, "Authenticating SAML Tokens in Proxy Service Requests."](#)
 - If you want operation request policies to require digital signatures, register the accepted client signature verification certificates in the Oracle WebLogic Server Certificate Registry. See [Section 45.7, "Configuring the Oracle WebLogic Security Framework: Main Steps."](#)
 - If you want operation request policies to require digital encryption, configure a service key provider that contains an encryption credential. The proxy service will use this credential to decrypt the encrypted SOAP message. See "Adding a service key provider" in "Service Key Providers" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.
4. In the Oracle Service Bus Console, do one or more of the following depending whether the WS-Policy of any of the operations in the proxy service contains security policy assertions that secure **responses** from the proxy service to clients:
 - If any operation response policy requires digital signatures, configure a service key provider that contains a digital signature credential. You can create one service key provider that contains credentials for both encryption and digital signatures. See "Adding Service Key Providers" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.
 - If any operation response policy specifies encryption, the client must send its certificate to the proxy service on the request. The proxy service will use the client's public key to encrypt its response. The client certificate must *not* be the same as the proxy service's encryption certificate.
5. In the Oracle Service Bus Console, create a proxy service from the WSDL that you imported. Activate your changes.
6. If the WSDL document does not have WS-Policy attachments and you want to add them, or if you want to specify a different WS-Policy from that of the WSDL, edit the proxy service you just created to do the following from the **Policies** tab:
 - a. Select **Custom Policy Bindings**.
 - b. To specify policies that apply to the entire service, expand the *service name* entry. Click Add to search for and select your policies.

- c. To specify policies that apply to an operation or the request/response of that operation, expand the *operation name* entry. Click Add to search for and select your policies.

Update the policy binding.

7. Edit the proxy service you just created to do the following from the **Security** tab:
 - a. Specify the service key provider that you created.
 - b. Select the **Process WS-Security Header** check box.
 - c. Optionally, modify the proxy service's default message-level access control policy, which specifies conditions under which users, groups, or roles can invoke the secured operations. See "Editing Message-Level Access Policies" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.
 - d. Optionally, modify the proxy service's message-level custom authentication settings.

52.3.2 Creating a Pass-Through Proxy Service: Main Steps

To create a pass-through proxy service:

1. Create a business service to which the proxy service will pass the unprocessed SOAP message. There are two configuration methods:
 - The business service is a Web service that contains WS-Policy statements.
 - The business service directly binds the WS-Policies. The WSDL on which the service is based should not have any WS-Policy statements.

See [Section 52.4, "Configuring Business Service Message-Level Security: Main Steps."](#)

2. If the WSDL document does not have WS-Policy attachments and you want to add them, or if you want to specify a different WS-Policy from that of the WSDL, edit the business service you just created to do the following from the **Policies** tab:
 - a. Select **Custom Policy Bindings**.
 - b. To specify policies that apply to the entire service, expand the *service name* entry. Click Add to search for and select your policies.
 - c. To specify policies that apply to an operation or the request/response of that operation, expand the *operation name* entry. Click Add to search for and select your policies.

Update the policy binding.

3. In the Oracle Service Bus Console, create a proxy service from a WSDL document. You can use the same WSDL document that you used for the business service that you created. Activate your changes.
4. If you should later edit the proxy service you just created, do **not** select the **Process WS-Security Header** check box on the **Security** tab.
5. Configure the proxy service to route to the business service that you created.

If you route to the business service based on the operation that the client's SOAP message is requesting to invoke, you must configure the routing so that it specifies an operation selection algorithm other than the SOAP body algorithm. Make sure the actions in the proxy service pipeline do not modify the WS-Security header or any parts of the SOAP envelope that are signed or encrypted. Changes to clear-text

message parts covered by digital signatures almost always break the digital signature because the signature cannot be verified later.

See [Section 2.3, "Working with Proxy Services."](#)

52.4 Configuring Business Service Message-Level Security: Main Steps

Outbound message-level security applies to messages between Oracle Service Bus proxy services and SOAP-HTTP or SOAP-JMS business services. It applies security to both the request and the response.

To configure outbound message-level security for a business service that represents a SOAP-HTTP or SOAP-JMS Web service:

1. In a text editor or IDE, create a WSDL document to define the policy.
2. In the Oracle Service Bus Console, import the Web service's WSDL document into the Oracle Service Bus WSDL repository and resolve any WSDL dependencies.

See "Adding WSDLs" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

3. In the Oracle Service Bus Console, do one or more of the following depending on whether the WSDL document contains WS-Policy statements that secure **requests** from a proxy service to the business service:
 - If any operation request policy includes an identity assertion with WS-Security Username Token as one of the supported token types, configure a service account for the business service. In the service account, provide the user name and password that you want the proxy service to send to the business service. Proxy services that route to this business service will get the username and password from this service account. See [Section 2.1.16, "Creating Service Account Resources"](#) and [Section 2.2, "Working with Business Services."](#)
 - If any operation request policy requires authentication with a WS-Security Username/Password token with password digest, make sure to enable password digests. See [Section 45.7, "Configuring the Oracle WebLogic Security Framework: Main Steps."](#)
 - If any operation request policy requires digital signatures, configure a service key provider that contains a digital signature credential. You can create one service key provider that contains credentials for both encryption and digital signatures. See "Adding Service Key Providers" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.
4. If any operation **response** policy in the business service requires encryption (that is, the business service encrypts the response with the proxy service's encryption public key), configure a service key provider and assign an encryption credential to the service key provider.

Caution: Encrypted back-end response messages: If the response policy of the business service specifies encryption, the proxy service will send its encryption certificate to the business service on the request. The business service will encrypt its response using the proxy service's public key. The proxy service encryption credential must not be the same as the business service encryption credential.

5. If any policy in the business service specifies using SAML assertions, configure a WebLogic SAML Credential Mapping Provider V2 asserting party. For more

information, see [Section 53.1, "Configuring SAML Credential Mapping: Main Steps."](#)

6. Create a business service from the WSDL that you imported. Activate your changes.
See [Section 2.2, "Working with Business Services."](#)
7. If you want to directly attach the policies to the service, edit the business service you just created to do the following from the **Policies** tab:
 - a. Select **Custom Policy Bindings**.
 - b. To specify policies that apply to the entire service, expand the *service name* entry. Click Add to search for and select your policies.
 - c. To specify policies that apply to an operation or the request/response of that operation, expand the *operation name* entry. Click Add to search for and select your policies.

Click Update to update the business service.

8. Create a proxy service that routes SOAP messages to the business service. You can use either an active-intermediary proxy service or a pass-through proxy service.
See [Section 52.3.1, "Creating an Active Intermediary Proxy Service: Main Steps."](#)

52.5 Examples of Custom WS-Policy Statements

The following sections provide examples of custom WS-Policy statements written under the WS-Policy specification using the proprietary Oracle schema for security policy:

- [Section 52.5.1, "Example: Encrypting Part of the SOAP Body and Header"](#)
- [Section 52.5.2, "Example: Encryption Policy for a Business Service"](#)
- [Section 52.5.3, "Example: Encrypting a Custom SOAP Header"](#)
- [Section 52.5.4, "Example: Signing the Message Body and Headers"](#)
- [Section 52.5.5, "Example: Signing a SOAP Body with SAML Holder-of-Key"](#)
- [Section 52.5.6, "Example: Authenticating, Signing, and Encrypting a SOAP Body and Headers with SAML Sender Vouches"](#)

52.5.1 Example: Encrypting Part of the SOAP Body and Header

If you need to specify that particular parts of the body of a SOAP message are encrypted or digitally signed, rather than the entire body, you must create a custom WS-Policy file.

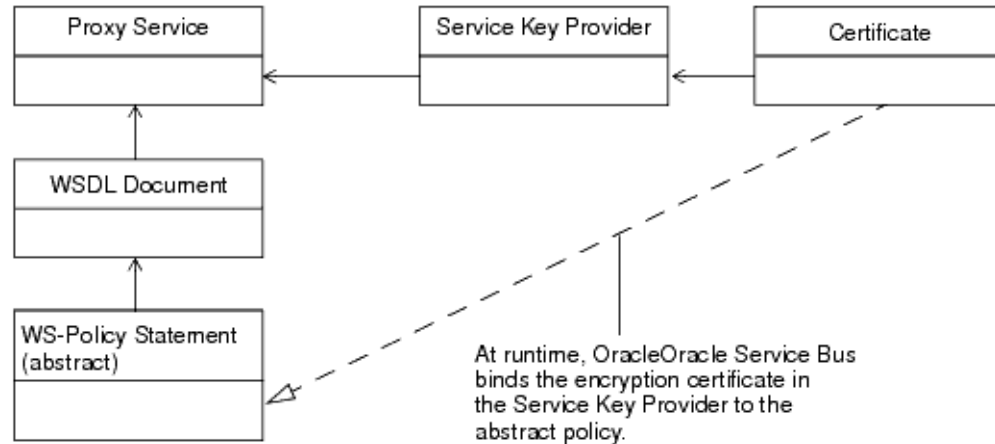
[Example 52-1](#) is an abstract WS-Policy statement that does the following:

- Requires the message from the client to include a user name and password token for authentication
- Requires the client to encrypt the user name and password token (which is in the security header)
- Requires the client to encrypt the `/definitions/message/CreditCardNumber` element

This policy cannot be used with a business service because it is abstract: its `KeyInfo` element does not contain the certificate used for encryption. Instead, when you

activate a proxy service that uses this WS-Policy statement, Oracle Service Bus binds to the WS-Policy statement the encryption certificate from the service key provider that you associate with the proxy service. See "Service Key Providers" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

Figure 52–1 Binding a Certificate to an Abstract Policy



Also in [Example 52–1](#):

- The `KeyWrappingAlgorithm` element specifies that the client must use the RSA 1.5 algorithm to wrap symmetric keys.
- The `EncryptionAlgorithm` specifies that the client must use the Triple DES (Data Encryption Standard) algorithm perform encrypt the security header and message body.

Example 52–1 Encrypting Part of the SOAP Body and Header

```

<wsp:Policy
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:wssp="http://www.bea.com/wls90/security/policy"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
    wssecurity-utility-1.0.xsd"
  xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
    wssecurity-secext-1.0.xsd"
  xmlns:m="http://example.org"
  wsu:Id="encrypt-custom-body-element-and-username-token">
  <!-- Require messages to provide a user name and password token
    for authentication -->
  <wssp:Identity>
    <wssp:SupportedTokens>
      <wssp:SecurityToken IncludeInMessage="true"
        TokenType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
          wss-username-token-profile-1.0#UsernameToken">
        <wssp:UsePassword Type="http://docs.oasis-open.org/wss/2004/01/
          oasis-200401-wss-username-token-profile-1.0#PasswordText"/>
      </wssp:SecurityToken>
    </wssp:SupportedTokens>
  </wssp:Identity>
  <wssp:Confidentiality>
    <wssp:KeyWrappingAlgorithm
      URI="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
    <!-- Require the user name and password in the security header
      to be encrypted -->
  
```

```

<wssp:Target>
  <wssp:EncryptionAlgorithm
    URI="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
  <wssp:MessageParts
    Dialect="http://www.bea.com/wls90/security/policy/wsee#part">
    wls:SecurityHeader(wsse:UsernameToken)
  </wssp:MessageParts>
</wssp:Target>
<!-- Require the /definitions/message/CreditCardNumber element to
  be encrypted -->
<wssp:Target>
  <wssp:EncryptionAlgorithm
    URI="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
  <wssp:MessageParts>
    wsp:GetBody(.) /m:CreditCardNumber
  </wssp:MessageParts>
</wssp:Target>
<!-- This is an abstract policy because the KeyInfo element is
  empty. The KeyInfo data is bound to the policy at runtime -->
  <wssp:KeyInfo/>
</wssp:Confidentiality>
</wsp:Policy>

```

52.5.2 Example: Encryption Policy for a Business Service

If you want messages to a business service to be encrypted, you must create a custom WS-Policy. The policy must be concrete (it must contain the encryption certificate instead of using a certificate from a service key provider) and it must be located directly in a WSDL document instead of being included by reference.

Typically, you would require messages to a business service to be encrypted if the proxy service that sends messages to the business service is a pass-through proxy service. That is, the proxy service that receives messages from a client does not process the SOAP message. Instead, the proxy service routes the message to the business service, and the business service takes on the responsibility of Web Services Security. See [Chapter 52, "Configuring Message-Level Security for Web Services."](#)

[Example 52–2](#) is a WSDL document that contains a concrete policy. Note the following about this example:

- The policy requires clients to encrypt the message body.
- The `KeyInfo` element specifies the type of token that a client must provide to is the parent element that is used to describe and embed the encryption certificate. The `BinarySecurityToken` element contains the base-64 encoded encryption certificate (the value is truncated in the example). If your certificate is in PEM format, the content of the PEM file (without the PEM prefix and suffix) is the base-64 encoded representation of the certificate. If your encryption certificate is stored in a JDK keystore, you can easily export it to a PEM file.
- The policy provides a unique ID and the WSDL uses a URI fragment to refer to the ID. See [Section 51.4, "Attaching WS-Policy Statements to WSDL Documents."](#)

Example 52–2 *Encrypting the Body with a Concrete Policy, Embedding the Policy in the WSDL Document*

```

<definitions name="WssServiceDefinitions"
  targetNamespace="http://com.bea.alsb/tests/wss"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wssu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
    wssecurity-utility-1.0.xsd"

```

```

xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
...>
<wsp:UsingPolicy xmlns:n1="http://schemas.xmlsoap.org/wsdl/"
  n1:Required="true"/>
<!-- The policy provides a unique ID -->
<wsp:Policy wsu:Id="myEncrypt.xml">
  <wssp:Confidentiality
    xmlns:wssp="http://www.bea.com/wls90/security/policy">
    <wssp:KeyWrappingAlgorithm
      URI="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
    <!-- Require the user name and password in the security header
      to be encrypted -->
    <wssp:Target>
      <wssp:EncryptionAlgorithm
        URI="http://www.w3.org/2001/04/xmlenc#tripleDES-cbc"/>
      <wssp:MessageParts Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part">
        wsp:Body()
      </wssp:MessageParts>
    </wssp:Target>
    <!-- Embed the token type and encryption certificate -->
    <wssp:KeyInfo>
      <wssp:SecurityToken
        TokenType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
          wss-x509-token-profile-1.0#X509v3"/>
      <wssp:SecurityTokenReference>
        <wssp:Embedded>
          <wsse:BinarySecurityToken
            EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
              200401-wss-soap-message-security-1.0#Base64Binary"
            ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-
              200401-wss-x509-token-profile-1.0#X509v3"
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
              200401-wss-wssecurity-secext-1.0.xsd">
              MIICfjCCAeegAwIBAgIQV/PDyj3...
            </wsse:BinarySecurityToken>
          </wssp:Embedded>
        </wssp:SecurityTokenReference>
      </wssp:KeyInfo>
    </wssp:Confidentiality>
  </wsp:Policy>
<binding name="WssServiceSoapBinding" type="tns:WssService">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getPurchaseOrder">
    <soap:operation soapAction="" style="document"/>
    <input>
      <soap:body parts="parameters" use="literal"/>
    <!-- Use a URI fragment to refer to the unique policy ID -->
    <wsp:Policy>
      <wsp:PolicyReference URI="#myEncrypt.xml"/>
    </wsp:Policy>
    </input>
    <output>
      <soap:body parts="parameters" use="literal"/>
    </output>
  </operation>
</binding>
...
</definitions>

```


52.5.3 Example: Encrypting a Custom SOAP Header

[Example 52-3](#) is an abstract WS-Policy statement that encrypts a custom header named `CreditCardNumber`.

If you need to specify that particular parts of the body of a SOAP message are encrypted or digitally signed, rather than the entire body, you must create a custom WS-Policy file.

This policy cannot be used with a business service because it is abstract: its `KeyInfo` element does not contain the certificate used for encryption. Instead, when you activate a proxy service that uses this WS-Policy statement, Oracle Service Bus binds to the WS-Policy statement the encryption certificate from the service key provider that you associate with the proxy service. See "Service Key Providers" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

Also of note in [Example 52-3](#):

- The `KeyWrappingAlgorithm` element specifies that the client must use the RSA 1.5 algorithm to wrap symmetric keys.
- The `EncryptionAlgorithm` specifies that the client must use the Triple DES (Data Encryption Standard) algorithm perform encrypt the security header.

Example 52-3 Encrypting a Custom SOAP Header

```
<wsp:Policy
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:wssp="http://www.bea.com/wls90/security/policy"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
    wssecurity-utility-1.0.xsd"
  wsu:Id="dig-sig-for-get-header">
  <wssp:Confidentiality>
    <wssp:KeyWrappingAlgorithm
      URI="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
    <!-- Require the custom CreditCardNumber header to be encrypted -->
    <wssp:Target>
      <wssp:EncryptionAlgorithm
        URI="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
      <wssp:MessageParts
        Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116">
        wsp:GetHeader(.) /n:CreditCardNumber
      </wssp:MessageParts>
    </wssp:Target>
    <wssp:KeyInfo/>
  </wssp:Confidentiality>
</wsp:Policy>
```

52.5.4 Example: Signing the Message Body and Headers

[Example 52-4](#) is a WS-Policy statement that requires a digital signature to access the following in the SOAP message:

- A custom header named `header1`
- All system headers
- The timestamp security header
- The message body

Example 52–4 Requiring a Signature for SOAP Headers and Body

```

<wsp:Policy
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:wssp="http://www.bea.com/wls90/security/policy"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wsssecurity-utility-1.0.xsd"
  wsu:Id="sign-custom-header-policy">
  <wssp:Integrity>
    <wssp:SignatureAlgorithm
      URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <wssp:CanonicalizationAlgorithm
      URI="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <!-- Require the custom header header1 to be signed -->
    <wssp:Target>
      <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <wssp:MessageParts
        Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116"
        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wsssecurity-secect-1.0.xsd"
        xmlns:n="http://example.org">
        wsp:GetHeader(/n:header1
      </wssp:MessageParts>
    </wssp:Target>
    <!-- Require the system headers to be signed -->
    <wssp:Target>
      <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <wssp:MessageParts
        Dialect="http://www.bea.com/wls90/security/policy/wsee#part">
        wls:SystemHeaders()
      </wssp:MessageParts>
    </wssp:Target>
    <!-- Require the Timestamp header to be signed -->
    <wssp:Target>
      <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <wssp:MessageParts
        Dialect="http://www.bea.com/wls90/security/policy/wsee#part">
        wls:SecurityHeader(wsu:Timestamp)
      </wssp:MessageParts>
    </wssp:Target>
    <!-- Require the message body to be signed -->
    <wssp:Target>
      <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <wssp:MessageParts
        Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part">
        wsp:Body()
      </wssp:MessageParts>
    </wssp:Target>
  </wssp:Integrity>
  <wssp:MessageAge/>
</wsp:Policy>

```

52.5.5 Example: Signing a SOAP Body with SAML Holder-of-Key

[Example 52–5](#) is a WS-Policy statement that requires the SAML assenter to use the holder-of-key method to sign the message body. The purpose of a SAML token with "holder-of-key" subject confirmation is to allow the subject to use an X.509 certificate that may not be trusted by the receiver to protect the integrity of the request messages.

For more information about the two SAML confirmation methods (sender-vouches or holder-of-key), see "SAML Token Profile Support in WebLogic Web Services" in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*.

The "Oracle WebLogic Server Security Policy Assertion Reference" in the *Oracle Fusion Middleware WebLogic Web Services Reference for Oracle WebLogic Server* describes the policy elements in detail.

Note the following about this example:

- `Integrity` specifies that part or all of the SOAP message must be digitally signed, as well as the algorithms and keys that are used to sign the SOAP message.
- `SignatureAlgorithm` specifies the cryptographic algorithm used to compute the digital signature.
- `CanonicalizationAlgorithm` specifies the algorithm used to canonicalize (use in simple or standard form) the SOAP message elements that are digitally signed. You can specify only `http://www.w3.org/2001/10/xml-exc-c14n#`.
- `DigestAlgorithm` specifies the digest algorithm that is used when digitally signing the specified parts of a SOAP message. You can specify only `http://www.w3.org/2000/09/xmlsig#sha1`.
- `MessageParts` specifies the parts of the SOAP message that should be signed, in this case the body.
- `Dialect` identifies the dialect used to identify the parts of the SOAP message that should be signed.
- `SupportedTokens` specifies the list of supported security tokens that can be used for digital signatures.
- `SecurityToken` specifies the security token that is supported for digital signatures.

`IncludeInMessage` specifies whether to include the token in the SOAP message. Valid values are true or false. The default value of this attribute is true when used in the `<Integrity>` assertion.

`TokenType` specifies the type of security token, in this case to specify a SAML token.

- `Claims` specifies additional metadata information that is associated with a particular type of security token. For SAML tokens, you must define a `<ConfirmationMethod>` child element to specify the type of SAML confirmation (sender-vouches or holder-of-key).
- `ConfirmationMethod` specifies the type of confirmation method, either sender-vouches or holder-of-key, that is used when using SAML tokens for identity.

Specify the `<ConfirmationMethod>` assertion within an `<Integrity>` assertion. The reason you put the SAML token in the `<Integrity>` assertion for this confirmation method is that the Web Service runtime must prove the integrity of the message, which is not required by sender-vouches.

Example 52–5 Signing a SOAP Body with SAML Holder-of-Key Method

```
<wsp:Policy
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:wssp="http://www.bea.com/wls90/security/policy"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
    wssecurity-utility-1.0.xsd"
```

```

xmlns:wls="http://www.bea.com/wls90/security/policy/wsee#part"
wsu:Id="saml-holder-of-key-signed">
<wssp: Integrity>
  <wssp: SignatureAlgorithm
    URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
  <wssp: CanonicalizationAlgorithm
    URI="http://www.w3.org/2001/10/xml-exc-c14n#"/>
  <wssp: Target>
    <wssp: DigestAlgorithm URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <wssp: MessageParts
      Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part">
      wsp:Body()
    </wssp: MessageParts>
  </wssp: Target>
  <wssp: SupportedTokens>
    <wssp: SecurityToken IncludeInMessage="true"
      TokenType="http://docs.oasis-open.org/wss/2004/01/oasis-2004-01-saml-
      token-profile-1.0#SAMLAssertionID">
      <wssp: Claims>
        <wssp: ConfirmationMethod>holder-of-key</wssp: ConfirmationMethod>
      </wssp: Claims>
    </wssp: SecurityToken>
  </wssp: SupportedTokens>
</wssp: Integrity>
</wsp: Policy>

```

52.5.6 Example: Authenticating, Signing, and Encrypting a SOAP Body and Headers with SAML Sender Vouches

[Example 52–6](#) is a WS-Policy statement that requires the SAML asserter to use the sender-vouches method to sign the message body and headers.

In sender-vouches the asserting party (different from the subject) vouches for the verification of the subject. The receiver must have a trust relationship with the asserting party.

For more information about the two SAML confirmation methods (sender-vouches or holder-of-key), see "SAML Token Profile Support in WebLogic Web Services" in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server*.

The "Oracle Web Services Security Policy Assertion Reference" in the *Oracle Fusion Middleware WebLogic Web Services Reference for Oracle WebLogic Server* describes the policy elements in detail.

Note the following about this example:

- `Identity` specifies the type of security tokens.
- `SupportedTokens` specifies the list of supported security tokens that can be used for digital signatures.
- `SecurityToken` specifies the security token that is supported for digital signatures.

`IncludeInMessage` is not specified because the value of this attribute is always true when used in the `<Identity>` assertion, even if you explicitly set it to false.

`TokenType` specifies the type of security token to specify a SAML token.

- `Claims` specifies additional metadata information that is associated with a particular type of security token. For SAML tokens, you must define a

<ConfirmationMethod> child element to specify the type of SAML confirmation (sender-vouches or holder-of-key).

- ConfirmationMethod specifies the type of confirmation method, either sender-vouches or holder-of-key, that is used when using SAML tokens for identity.
- Integrity specifies that part or all of the SOAP message must be digitally signed (in this example both the body and security headers), as well as the algorithms and keys that are used to sign the SOAP message.
- SignatureAlgorithm specifies the cryptographic algorithm used to compute the digital signature.
- CanonicalizationAlgorithm specifies the algorithm used to canonicalize (use in simple or standard form) the SOAP message elements that are digitally signed. You can specify only `http://www.w3.org/2001/10/xml-exc-c14n#`.
- Target encapsulates information about which targets of a SOAP message are to be encrypted or signed, depending on the parent element. The child elements also depend on the parent element:
 - When used in <Integrity>, you can specify the <DigestAlgorithm>, <Transform>, and <MessageParts> child elements.
 - When used in <Confidentiality>, you can specify the <EncryptionAlgorithm>, <Transform>, and <MessageParts> child elements.
- DigestAlgorithm specifies the digest algorithm that is used when digitally signing the specified parts of a SOAP message. You can specify only `http://www.w3.org/2000/09/xmldsig#sha1`.
- MessageParts specifies the parts of the SOAP message that should be signed, in this case the body and security header.
- Dialect identifies the dialect used to identify the parts of the SOAP message that should be signed.
- Confidentiality specifies that part or all of the SOAP message must be encrypted, as well as the algorithms and keys that are used to encrypt the SOAP message. The example requires that the body and security headers must be encrypted using triple-DES.

Example 52-6 Signing a SOAP Body and Headers with SAML Sender-Vouches Method

```
<wsp:Policy
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:wssp="http://www.bea.com/wls90/security/policy"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
    wssecurity-utility-1.0.xsd"
  xmlns:wls="http://www.bea.com/wls90/security/policy/wsee#part"
  wsu:Id="samlPolicy-sender-vouches-signed-encrypted">
  <wssp:Identity>
    <wssp:SupportedTokens>
      <wssp:SecurityToken
        TokenType="http://docs.oasis-open.org/wss/2004/01/oasis-2004-01-
          saml-token-profile-1.0#SAMLAssertionID">
        <wssp:Claims>
          <wssp:ConfirmationMethod>
            sender-vouches
          </wssp:ConfirmationMethod>
        </wssp:Claims>
      </wssp:SecurityToken>
    </wssp:SupportedTokens>
  </wssp:Identity>
</wsp:Policy>
```

```

        </wssp:SecurityToken>
    </wssp:SupportedTokens>
</wssp:Identity>
<b>wssp: Integrity</b>
    <b>wssp: SignatureAlgorithm</b>
        URI="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <b>wssp: CanonicalizationAlgorithm</b>
        URI="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    <b>wssp: Target</b>
        <b>wssp: DigestAlgorithm</b>
            URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <b>wssp: MessageParts</b>
            Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part">
                wsp:Body ()
            </wssp: MessageParts>
        </wssp: Target>
    </wssp: Target>
    <b>wssp: Target</b>
        <b>wssp: DigestAlgorithm</b>
            URI="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <b>wssp: MessageParts</b>
            Dialect="http://www.bea.com/wls90/security/policy/wsee#part">
                wls:SecurityHeader (Assertion)
            </wssp: MessageParts>
        </wssp: Target>
</wssp: Integrity>
<b>wssp: Confidentiality</b>
    <b>wssp: KeyWrappingAlgorithm</b>
        URI="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
    <b>wssp: Target</b>
        <b>wssp: EncryptionAlgorithm</b>
            URI="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
        <b>wssp: MessageParts</b>
            Dialect="http://www.bea.com/wls90/security/policy/wsee#part">
                wls:SecurityHeader (Assertion)
            </wssp: MessageParts>
        </wssp: Target>
    </wssp: Target>
    <b>wssp: Target</b>
        <b>wssp: EncryptionAlgorithm</b>
            URI="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
        <b>wssp: MessageParts</b>
            Dialect="http://schemas.xmlsoap.org/2002/12/wsse#part">
                wsp:Body ()
            </wssp: MessageParts>
        </wssp: Target>
    </wssp: KeyInfo/>
</wssp: Confidentiality>
</wsp: Policy>

```

52.6 Disabling Business Service Message-Level Security

Some infrequently used design patterns preempt a proxy service from automatically generating the outbound WS-Security SOAP envelope and instead use an XQuery expression to create the envelope. If you use this design pattern, to prevent a proxy service from automatically generating the outbound WS-Security SOAP envelope, you must create an action in the proxy service's message flow that sets the value of the `./ctx:security/ctx:doOutboundWss` element in the `$outbound` message context variable to `xs:boolean("false")`. You can create the action in either of the following places:

- In a request stage of a pipeline pair. See [Section 2.4.16, "Adding and Configuring Pipeline Pair Nodes in Message Flows."](#)
- In a request action of a route node. See [Section 2.4.25, "Adding and Configuring Route Nodes in Message Flows."](#)

For information about the `$outbound` message context variable, see "Message Context" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*.

Under some circumstances, when you attempt to activate a session in which you have created or modified a proxy service with outbound message-level security disabled, the Oracle Service Bus Console reports validation errors (you cannot commit a session that contains errors). If your session validation reports errors because you have disabled outbound message-level security, modify the Oracle Service Bus startup command so that it sets the following system property to `true`:

```
com.bea.wli.sb.security.wss.LaxOutboundWssValidation
```

Then restart Oracle Service Bus. With this property set to `true`, the Oracle Service Bus Console reports warnings instead of errors (you can commit a session that reports warning messages).

Future releases of Oracle Service Bus will provide an easier way to disable outbound message-level security.

Using SAML for Authentication

Security Assertion Markup Language (SAML) defines a framework for exchanging authentication and authorization information between online business partners. Oracle Service Bus enables the following techniques for using SAML:

- If your clients do not provide SAML tokens but your business services require them, you can configure a proxy service to map the client's identity to a SAML token. See [Section 53.1, "Configuring SAML Credential Mapping: Main Steps."](#)
- If your clients provide SAML tokens to a pass-through proxy service, you can propagate the client's SAML token to the business service. See [Section 53.2, "Configuring SAML Pass-Through Identity Propagation."](#)
- If your clients provide SAML tokens to an active intermediary proxy service, you can configure the proxy service to assert the client's identity. See [Section 53.3, "Authenticating SAML Tokens in Proxy Service Requests."](#)

For an overview of SAML, see the OASIS technical overview at the following URL:

<http://www.oasis-open.org/committees/download.php/6837/sstc-saml-tech-overview-1.1-cd.pdf>

The complete SAML specification set of documents are available at the following URL:

<http://www.oasis-open.org/committees/download.php/3400/oasis-sstc-saml-1.1-pdf-xsd.zip>

53.1 Configuring SAML Credential Mapping: Main Steps

If your clients do not provide SAML tokens but your business services require them, you can configure a proxy service to map the client's identity to a SAML token.

This technique requires the business service to be a Web service with WS-Policy statements that require authentication using SAML tokens.

To configure SAML credential mapping:

1. Configure a trust relationship between Oracle Service Bus and the system (message consumer) that the business service represents.

The message consumer acts as a relying party and must have a trust relationship with Oracle Service Bus.

See "Important Information Regarding Cross-Domain Security Support" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

2. Configure the SAML providers:

For Oracle Web Services Manager policies: See "Configuring SAML" (WSSEC2376) in the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

For WLS 9.2 policies: Configure the WebLogic SAML Identity Assertion Provider V2 and the WebLogic SAML Credential Mapping Provider V2 in your security domain. See "Configuring a SAML Identity Assertion Provider" and "Configuring a SAML Credential Mapping Provider" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

3. Configure a proxy service to authenticate clients using any of the following techniques:
 - HTTP or HTTPS BASIC (client provides user name and password in the request)
 - HTTPS Client certificate
 - Message-level authentication (using any of the supported token profiles)

If a client request includes a WS-Security security header, you must configure the proxy service to process this header on the proxy service side of the message. In Oracle Service Bus, you cannot add a SAML header (or any other WS-Security header) to a SOAP envelope that already contains a WS-Security header, neither can you add SAML (or other) security tokens to an existing security header.
 - Third-party authentication
4. Configure the proxy service to include a SAML token in the WS-Security header of its outbound request.

Note: If you configured the proxy service for dynamic routing, the message context determines the target URL for the request. If the assertion is signed, you must configure the certificate.

When the proxy service sends its outbound request, it generates a SAML assertion on behalf of the client. When the business service processes the WS-Security header, it validates the SAML assertion, creates a security context for the identity in the SAML assertion, and invokes the Web service with this security context.

53.2 Configuring SAML Pass-Through Identity Propagation

If your clients provide SAML tokens to a pass-through proxy service, you can propagate the client's SAML token to the business service.

This technique requires the business service to be a Web service with WS-Policy statements that require authentication using SAML tokens.

To configure SAML pass-through identity propagation:

1. Configure a trust relationship between Oracle Service Bus and the back-end service.

See "Important Information Regarding Cross-Domain Security Support" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

2. Configure the back-end service to act as a SAML relying party.

For Oracle Web Services Manager policies: See "How to Configure Oracle Platform Security Services (OPSS) for SAML Policies" in the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

For WLS 9.2 policies: See "Create a SAML Relying Party" in the *Oracle Fusion Middleware Oracle WebLogic Server Administration Console Online Help*.

3. Configure a pass-through proxy service.
See [Section 52.3.2, "Creating a Pass-Through Proxy Service: Main Steps."](#)
4. Configure a SOAP-HTTP or SOAP-JMS business service with WS-Policy statements that require authentication using SAML tokens.
See [Section 52.4, "Configuring Business Service Message-Level Security: Main Steps."](#)

53.3 Authenticating SAML Tokens in Proxy Service Requests

If your clients provide SAML tokens to an active intermediary proxy service, you can configure the proxy service to assert the client's identity.

To configure a proxy service to use SAML tokens to authenticate clients:

1. Configure a trust relationship between the client software and Oracle Service Bus.
Oracle Service Bus relies on SAML assertions issued by the client, or on behalf of the client.

See "Important Information Regarding Cross-Domain Security Support" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

2. Configure SAML.

For Oracle Web Services Manager policies: See "Configuring SAML" (WSSEC2376) in the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

For WLS 9.2 policies: Configure the WebLogic SAML Identity Assertion Provider V2 to validate tokens issued by the client's SAML authority. See "Configuring a SAML Identity Assertion Provider" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

When configuring the identity assertion provider, note the following requirements:

- The confirmation method from the policy must match the SAML profile in the SAML asserting party.
 - Specify the asserting party target URL to be the relative URL of the proxy (not including the protocol and host information).
 - For signed assertions, add the certificate to the Identity Asserter registry.
3. Configure the SAML credential mapping provider.

For Oracle Web Services Manager policies: See "Configuring SAML" (WSSEC2376) in the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

For WLS 9.2 policies: Configure the WebLogic SAML Credential Mapping Provider V2 in your security domain. See "Configuring a SAML Credential Mapping Provider" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

4. Create an active intermediary proxy service that communicates over the HTTP, HTTPS, or JMS protocol. The proxy service must be a Web service with a WS-Policy statement that requires authentication and accepts SAML tokens.

A proxy service that communicates over the "local" transport type cannot use a SAML token profile to authenticate.

53.4 Configuring SAML Authentication with Service Bus (SB) Transport

If you are using SAML-based authentication with the SB transport, be sure to follow these configuration requirements:

- On the asserting party, configure the SAML Credential mapper with URI `http://openuri.org/<OSBProxyServiceURI>`, where `<OSBProxyServiceURI>` is the SB transport service URI.
- When configuring the Identity Assertion provider on the Oracle Service Bus side (the relying party), use the asserting party target URL as the proxy endpoint URI. Do not include the protocol and host information. For example, `/<OSBProxyServiceURL>`.

53.5 Troubleshooting SAML Web Services Security

Question: I am trying to propagate my proxy service transport identity to a destination business service and keep receiving error, `Unable to add security token for identity`. What does this mean?

Answer: There are various causes for this error. Generally this means one of the following problems:

- The SAML Credential Mapper is not configured correctly. Double check that the configuration is in accordance with the following instructions:

For Oracle Web Services Manager policies: "Configuring SAML" (WSSEC2376) in the *Oracle Fusion Middleware Security and Administrator's Guide for Web Services*.

For WLS 9.2 policies: "Configuring a SAML Credential Mapping Provider" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

- Another common source of this error is that there is no subject information to propagate. To generate a SAML token, you must have a transport-level or message-level subject. Make sure that the client has a subject. This can be done by inspecting `$security` message context variable.

Question: I am trying to propagate my proxy service transport identity to a destination business service using SAML holder-of-key and keep receiving error, `Failure to add signature`. What does this mean?

Answer: There are various causes for this error, but most likely is that the credentials are not configured for the business service's service key provider. When Oracle Service Bus generates an outbound holder-of-key assertion, it generally also generates a digital signature over the message contents, so that the recipient can verify not only that a message is received from a particular user, but that the message has not been tampered with. To generate the signature, the business service must have a service key provider with a digital signature credential associated with it.

Question: I am trying to configure an active intermediary proxy service that receives SAML identity tokens and keep receiving errors that look like: `The SAML token is not valid`. How do I fix this?

Answer: This is generally caused by a lack of a SAML Identity Asserter or SAML Identity Asserter asserting party configuration for the proxy. For a proxy service to receive SAML assertions in active intermediary mode, it must have a SAML Identity Asserter configured. For more details, see "Configuring a SAML Identity Assertion Provider" in *Oracle Fusion Middleware Securing Oracle WebLogic Server*.

Configuring Custom Authentication

Oracle Service Bus supports client-specified custom authentication credentials for both transport- and message-level proxy service requests. The custom authentication credentials can be in the form of tokens, or a username and password token combination.

Oracle Service Bus accepts and attempts to authenticate a custom token passed to a proxy service in an HTTP header, SOAP header (for SOAP-based proxy services) or in the payload (for non-SOAP proxy services). You use the proxy service configuration wizard to configure the proxy service with the mechanism by which the token is passed, and the token type.

Oracle Service Bus also accepts and attempts to authenticate a username and password token passed in a SOAP header (for SOAP based proxy services), or in the payload for non-SOAP proxy services. You use the proxy service configuration wizard to configure the proxy service with the mechanism by which the username and password are passed.

Note: The custom authentication mechanisms work alone or in concert with the message-level security for Web services described in [Chapter 52, "Configuring Message-Level Security for Web Services."](#) See [Section 54.9, "Combining WS-Security with Custom Username/Password and Tokens"](#) for information about using both types of security.

The following custom authentication mechanisms are supported:

- Transport-Level Security
 - Custom token in an HTTP header
- Message-Level Security
 - For SOAP-based proxy services
 - * Custom token in a SOAP header
 - * Username/password in a SOAP header
 - For non-SOAP-based proxy services
 - * Custom token in the payload of any XML-based proxy services
 - * Username/password in the payload of any XML-based proxy services

This section describes the following custom authentication topics:

- [Section 54.1, "What Are Custom Authentication Tokens?"](#)

- [Section 54.1.1, "Custom Authentication Token Use and Deployment"](#)
- [Section 54.2, "Understanding Transport-Level Custom Authentication"](#)
- [Section 54.3, "Understanding Message-Level Custom Authentication"](#)
- [Section 54.5, "Configuring Identity Assertion Providers for Custom Tokens"](#)
- [Section 54.6, "Configuring Custom Authentication Transport-Level Security"](#)
- [Section 54.7, "Configuring Custom Authentication Message-Level Security"](#)
- [Section 54.8, "Propagating the Identity Obtained From Custom Authentication Tokens"](#)
- [Section 54.9, "Combining WS-Security with Custom Username/Password and Tokens"](#)

54.1 What Are Custom Authentication Tokens?

An authentication token is some data, represented as a string or XML, that identifies an entity (user or process), such as an X509 client certificate. Typically, authentication tokens are designed to be used within specific security protocols. Some authentication tokens are cryptographically protected and some are not. Some authentication tokens carry key material.

In the context of Oracle Service Bus, a custom authentication token can be a username/password or an opaque identity assertion token in a user-defined location in the request. A username/password token is allowed in a SOAP header (for SOAP-based services) or in the payload of some non-SOAP proxy service. An identity assertion token is allowed in an HTTP header, in a SOAP header (for SOAP-based services), or in the payload of some non-SOAP proxy service. The Oracle Service Bus domain must include an Identity Assertion provider that supports the token type.

Oracle Service Bus uses the authenticated user to establish a security context for the client. The security context established by authenticating a custom token or username and password can be used as the basis for outbound credential mapping and access control.

To authenticate and authorize clients who supply tokens for authentication, you must configure an Identity Assertion provider that maps the client's credential to an Oracle Service Bus user. Oracle Service Bus uses this resulting username to establish a security context for the client.

54.1.1 Custom Authentication Token Use and Deployment

The addition of custom authentication token support in Oracle Service Bus addresses two customer needs. In the first scenario, a proxy service request has a username/password somewhere in the message payload, for example in a SOAP header. Oracle Service Bus must get this username/password and authenticate the user.

In the second scenario, the message contains some kind of authentication token (other than username/password), such as a secure-token-xyz token. The token may be in an HTTP header or in the message payload. Oracle Service Bus must get the token and authenticate it. In either case, a security context is established if authentication succeeds.

Most security-related configuration is typically done at deployment time, and custom authentication fits that model: it can be configured directly on the production

environment at deployment time. Alternatively, you can configure authentication during staging and import it into the production environment.

Custom authentication, which includes both username/password tokens and custom tokens, is an integral part of the proxy service definition. When a proxy service is exported, any configuration of custom tokens is included in the jar file. When a new version of the proxy service is imported, the previous configuration is overwritten with whatever configuration is contained in the jar file.

Only users in the **IntegrationDeployer** or **IntegrationAdministrator** roles can configure custom token authentication. Users in the **IntegrationOperator** or **IntegrationMonitor** roles have read-only access to this configuration.

54.2 Understanding Transport-Level Custom Authentication

You can authenticate client requests at the transport-level via custom authentication tokens. You specify a custom token in an HTTP header. The HTTP (and HTTPS) configuration page of the service definition wizard allows you to configure client authentication. The options for HTTP and HTTPS proxy services are:

- None
- Basic
- Custom Authentication
- Client Certificate (HTTPS Only)

These are mutually exclusive options.

If you choose custom authentication, you must also specify the name of the HTTP header that is to carry the token, and the token type.

The steps for configuring transport-level custom credentials are described in [Section 2.3, "Working with Proxy Services."](#)

The custom authentication token can be any active token type, previously configured for an Identity Assertion provider, that is carried in an HTTP header.

You need to configure, or create and configure, an Identity Assertion provider that handles the token type you plan to use. See [Section 54.5, "Configuring Identity Assertion Providers for Custom Tokens."](#)

After you have configured the transport-level custom credentials, you can then additionally configure the message level security configuration, as described in [Chapter 52, "Configuring Message-Level Security for Web Services."](#)

54.2.1 Importing and Exporting and Transport-Level Custom Token Authentication

Transport-level custom authentication tokens are published to the UDDI. The `client-auth` property is present in the `instanceParams` of the HTTP or HTTPS transport attributes whenever authentication is configured. As described in "Transport Attributes" in the *Oracle Fusion Middleware Administrator's Guide for Oracle Service Bus*, the possible values of `client-auth` are `BASIC`, `CLIENT-CERT` and `CUSTOM-TOKEN`. Whenever the value is `CUSTOM-TOKEN`, two additional properties are present: `token-header` and `token-type`.

Note: Oracle Service Bus business service definitions do not support custom token authentication. If you import a service from UDDI that has client-auth equal to CUSTOM-TOKEN, the service is imported as if it does not have any authentication configuration.

54.3 Understanding Message-Level Custom Authentication

Oracle Service Bus supports client-specified custom authentication credentials for proxy service message-level requests. The custom authentication credentials can be in the form of a custom token, or a username and password.

Oracle Service Bus accepts and attempts to authenticate a custom token passed to a proxy service in a SOAP header (for SOAP-based proxy services), or in the payload (for non-SOAP proxy services). You use the proxy service configuration wizard to configure the proxy service with the mechanism by which the token is passed, and the token type.

Oracle Service Bus also accepts and attempts to authenticate a username and password token passed in a SOAP header (for SOAP based proxy services), or in the payload for non-SOAP proxy services. You use the proxy service configuration wizard to configure the proxy service with the mechanism by which the username and password are passed.

The following proxy service message-level authentication mechanisms are now supported:

- For SOAP-based proxy services
 - Custom token in a SOAP header
 - Username/password in a SOAP header
- For non-SOAP-based proxy services
 - Custom token in the payload of any XML-based proxy services
 - Username/password in the payload of any XML-based proxy services

Message-level custom tokens and message-level username and password are supported on proxy services of the following binding types:

- WSDL-SOAP
- WSDL-XML
- Abstract SOAP
- Abstract XML
- Mixed – XML (in the request)
- Mixed – MFL (in the request)

54.4 Format of XPath Expressions

The configuration for both custom username/password and custom token is similar. In both cases, you specify XPath expressions that enable Oracle Service Bus to locate the necessary information. The root of these XPath expressions is as follows:

- Use `soap-env:Envelope/soap-env:Header` if the service binding is anySOAP or WSDL-SOAP.

- Use `soap-env:Body` (specifically, the contents of the `$body` variable) if the service binding is not SOAP based.

Note: All XPath expressions must be in a valid XPath 2.0 format. The XPath expressions must use the XPath "declare namespace" syntax to declare any namespaces used, as follows:

```
declare namespace
ns='http://webservices.mycompany.com/MyExampleService';
```

For example,

```
declare namespace y="http://foo";./y:my-custom-token/text()
```

54.5 Configuring Identity Assertion Providers for Custom Tokens

An Identity Assertion provider is a specific form of Authentication provider that allows users or system processes to assert their identity using tokens. A client's identity is established through the use of client-supplied tokens. The Identity Assertion provider validates the token. If the token is successfully validated, the Identity Assertion provider maps the token to an Oracle Service Bus username, and returns the username. Identity is said to be "asserted" when the token is mapped to the username. Oracle Service Bus then uses this user name to establish a security context for the client.

If you want the proxy service to consume a custom token, check the provided Oracle WebLogic Server Identity Assertion providers to see if one meets your needs. Oracle WebLogic Server includes a broad array of Identity Assertion providers, including the following:

- The *WebLogic Identity Assertion provider* validates X.509 and IIOP-CSIV2 tokens and optionally can use a user name mapper to map that token to a user.
- The *SAML Identity Assertion provider*, which acts as a consumer of SAML security assertions.

If you want the Oracle Service Bus proxy service to consume a custom token that is not handled by one of the bundled Identity Assertion providers, for example a `secure-token-xyz` token, you (or a third-party) must first write an Oracle WebLogic Server Identity Assertion provider that supports the token type and use the Oracle WebLogic Server Administration Console to add that provider to the security realm.

You develop Identity Assertion providers to support the specific types of custom tokens that you will be using to assert the identities of users. You can develop an Identity Assertion provider to support multiple token types. While you can have multiple Identity Assertion providers in a security realm with the ability to validate the same token type, only one Identity Assertion provider can actually perform this validation.

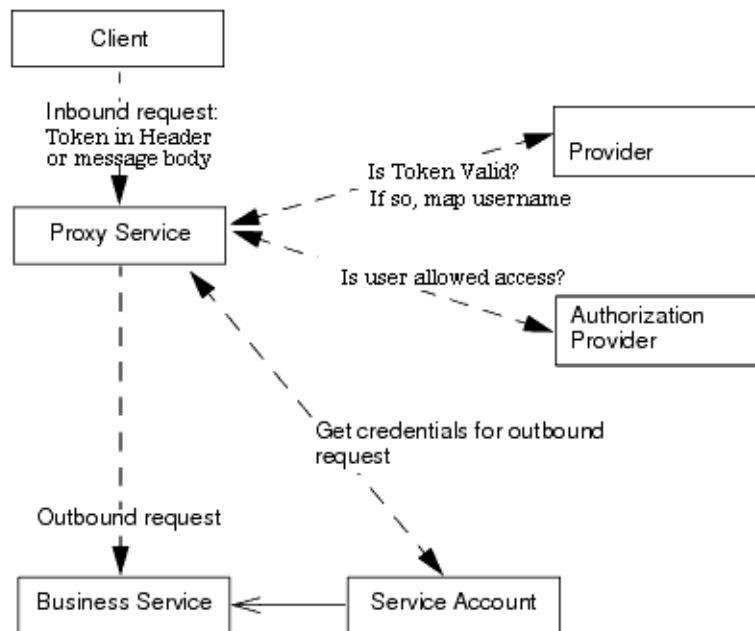
The Identity Assertion process is shown in [Figure 54-1](#), and works as follows:

1. The proxy service gets the authentication token from the inbound request.
2. The token is passed to an Identity Assertion provider that is responsible for validating tokens of that type and that is configured as "active."
3. The Identity Assertion provider validates the token.
4. If the token is successfully validated, the Identity Assertion provider maps the token to a username, and returns the username.

5. Oracle Service Bus then continues the authentication process with this username and, if successful, obtains the authenticated subject.
6. Oracle Service Bus creates the security context. The security context established by authenticating a custom token or username and password can be used as the basis for outbound credential mapping and access control.

See "Identity Assertion and Tokens" in *Oracle Fusion Middleware Understanding Security for Oracle WebLogic Server* for additional information.

Figure 54–1 Identity Assertion and Custom Tokens



54.5.1 Object Type of Custom Tokens

For transport-level identity assertion, the header value is passed as a `java.lang.String` to the identity assertion providers. For message-level identity assertion, the XPath expression is evaluated as follows:

- If the XPath expression returns multiple nodes, an error is raised and identity assertion is not called.
- If the XPath expression returns an empty result, identity assertion is called with a null argument.
- If the XPath expression returns a single token of type TEXT or ATTR (See `XmlCursor.TokenType` at <http://xmlbeans.apache.org/docs/2.0.0/reference/org/apache/xmlbeans/XmlCursor.TokenType.html>), the string value of the text node or attribute is passed (as returned by `XmlCursor.getStringValue()`). Otherwise, a single `XmlObject` is passed.

54.5.2 Configuring a Custom Token Type in an Identity Assertion Provider

The steps required to complete these tasks are described in detail in the following Oracle WebLogic Server documents:

- "How to Create New Token Types" in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server* describes how to create custom token types for an Identity Assertion provider.
- *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server* describes how to configure identity assertion providers in the Oracle WebLogic Server Administration Console.

For your convenience, the steps for creating custom token types for an Identity Assertion provider and configuring that provider in the Oracle WebLogic Server Administration Console are briefly listed here. However, you will need to consult the Oracle WebLogic Server documentation to actually complete the tasks.

54.5.2.1 Steps for Configuring a Custom Token Type in an Identity Assertion Provider

You can develop a custom Identity Assertion provider by following these steps:

1. Create the new token types. See "How to Create New Token Types" in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server*.
2. "Create Runtime Classes Using the Appropriate SSPIs," described in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server*. That section shows the `SampleIdentityAsserterProviderImpl.java` class, which is the runtime class for the sample Identity Assertion provider.
3. "Generate an MBean Type Using the WebLogic MBeanMaker," described in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server*.
4. "Configure the Custom Identity Assertion Provider Using the Administration Console," described in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server*.
5. Define the active token type. For this task, see "Configuring Identity Assertion Providers" in *Oracle Fusion Middleware Securing Oracle WebLogic Server* and "How to Make New Token Types Available for Identity Assertion Provider Configurations" in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server*.

54.5.2.2 Setting the Supported and Active Types in the MBean

When you configure a custom Identity Assertion provider, the **Supported Types** field displays a list of the token types that the Identity Assertion provider supports. You enter zero or more of the supported types in the **Active Types** field.

The content for the **Supported Types** field is obtained from the **SupportedTypes** attribute of the MBean Definition File (MDF), which you use to generate your custom Identity Assertion provider's MBean type. An example from the sample Identity Assertion provider is shown in [Example 54–1](#). For more information about MDFs and MBean types, see "Generate an MBean Type Using the WebLogic MBeanMaker" in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server*.

Example 54–1 SampleIdentityAsserter MDF: SupportedTypes Attribute

```
<MBeanType>
...
<MBeanAttribute
Name = "SupportedTypes"
Type = "java.lang.String[]"
Writeable = "false"
Default = "new String[] {&quot;SamplePerimeterAtnToken&quot;}"/>
/>
```

```
...
</MBeanType>
```

Similarly, the content for the **Active Types** field is obtained from the **ActiveTypes** attribute of the MBean Definition File (MDF). You can default the **ActiveTypes** attribute in the MDF so that it does not have to be set manually with the Oracle WebLogic Server Administration Console. An example from the sample Identity Assertion provider is shown in [Example 54–2](#).

Example 54–2 SampleIdentityAsserter MDF: ActiveTypes Attribute with Default

```
<MBeanAttribute
Name= "ActiveTypes"
Type= "java.lang.String[]"
Default = "new String[] { &quot;SamplePerimeterAtnToken&quot; }"
/>
```

While defaulting the **ActiveTypes** attribute is convenient, you should only do this if no other Identity Assertion provider will ever validate that token type. Otherwise, it would be easy to configure an invalid security realm (where more than one Identity Assertion provider attempts to validate the same token type). Best practice dictates that all MDFs for Identity Assertion providers turn off the token type by default; then an administrator can manually make the token type active by configuring the Identity Assertion provider that validates it.

54.6 Configuring Custom Authentication Transport-Level Security

You ultimately configure custom authentication for transport-level security. However, before you get to this step of the process, you must first configure, or potentially create and configure, an Identity Assertion provider that understands the token type you plan to use.

The steps required to complete these tasks are described in detail in the following Oracle WebLogic Server documents:

- If one of the bundled Identity Assertion providers meets your needs, see "Configure Identity Assertion Providers" in *Oracle Fusion Middleware Securing Oracle WebLogic Server* for instructions on how to configure this Identity Assertion provider in the Oracle WebLogic Server Administration Console.
- *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server* describes how to create custom token types for an Identity Assertion provider.
- "Configuring Identity Assertion Providers" in *Oracle Fusion Middleware Securing Oracle WebLogic Server* describes how to configure identity assertion providers in the Oracle WebLogic Server Administration Console.

54.6.1 Steps for Configuring Custom Authentication Transport-Level Security

The steps for configuring custom authentication transport-level security are as follows:

1. Determine which custom token format you will be using.
2. Determine if an existing provider meets your needs. "Choosing an Authentication Provider" in *Oracle Fusion Middleware Securing Oracle WebLogic Server* offers guidance on this task.
3. Configure, or create and configure, an Identity Assertion provider that supports the token format.

4. The Identity Assertion provider maps the token to a username. Add the client's username to the Oracle Service Bus Security Configuration module.
5. On the protocol-dependent transport configuration page, specify the **Authentication Header** where Oracle Service Bus is to find the token and the **Authentication Token Type**. Only those token types that are currently active for a configured Identity Assertion provider are displayed.

54.7 Configuring Custom Authentication Message-Level Security

You ultimately configure custom authentication message-level security. However, before you get to this step of the process, you must first configure, or potentially create and configure, an Authentication provider or Identity Assertion provider that understands the token type you plan to use.

The steps required to complete these tasks are described in detail in the following Oracle WebLogic Server documents:

- If one of the bundled Authentication or Identity Assertion providers meets your needs, see "Configuring Authentication Providers" in *Oracle Fusion Middleware Securing Oracle WebLogic Server* for instructions on how to configure this Authentication provider in the Oracle WebLogic Server Administration Console.
- "How to Create New Token Types" in *Oracle Fusion Middleware Developing Security Providers for Oracle WebLogic Server* describes how to create custom token types for an identity assertion provider.
- "Configuring Identity Assertion Providers" in *Oracle Fusion Middleware Securing Oracle WebLogic Server* describes how to configure identity assertion providers in the Oracle WebLogic Server Administration Console.

54.7.1 Steps for Configuring Custom Authentication Message-Level Security

The steps for configuring custom authentication message-level security are as follows:

1. Determine which custom username/password or token format you will be using.
2. Determine if an existing provider meets your needs. "Choosing an Authentication Provider" in *Oracle Fusion Middleware Securing Oracle WebLogic Server* offers guidance on this task.

If you specify any **Context Properties** you will probably need to create your own provider because the provider must know which property names to expect.

3. Configure, or create and configure, an authentication provider or identity assertion provider that supports the username/password or token format, respectively. This provider must also understand any **Context Properties** that you want to provide.
4. Add the client's user name to the Oracle Service Bus Security Configuration module.
5. On the Security page, configure a new or existing proxy service for the **User Name XPath**, **User Password XPath**, or **Token Type** and **Token Path**, as appropriate.
6. Specify the **Property Name** and **Value Selector** of any **Context Properties** that you want to provide.

54.8 Propagating the Identity Obtained From Custom Authentication Tokens

The security context established via a custom token or custom username/password is in no way unique, and you can use it for credential mapping. If you implement both transport-level authentication and message-level authentication, the message-level security context is always used for credential mapping and identity propagation.

For example, if the proxy service authenticates the client via a secure-token-xyz token in a SOAP header, the authenticated subject is used during any mapped service account lookup. The subject is also used when generating SAML tokens on outbound messages. Java callouts can also run under the authentication context associated with a custom token or custom username/password.

If a custom username/password is used, the username/password in the custom token can be used for outbound HTTP BASIC or outbound WS-Security Username Token authentication if a pass-through service account is used.

54.9 Combining WS-Security with Custom Username/Password and Tokens

You can secure Oracle Service Bus proxy services with either transport-level security (for example, HTTPS) and message-level security (for example, WS-Security and custom tokens), or a combination of both. That is, you can configure an Oracle Service Bus proxy service with both transport-level authentication and message-level authentication.

For example, client requests can be authenticated at the transport level with custom tokens in HTTP headers, and at the message level with WSS security tokens, custom tokens, or username/passwords, except in the Web Services Security header.

However, note the following restriction: Although it is possible to combine WS-Security and message-level custom tokens, the WS-Security policy must **not** require proxy service authentication based on WS-Security tokens. Message-level custom tokens and WS-Security proxy service authentication are mutually exclusive.

Consider the following distinction:

- It is allowable to configure a proxy service that expects a custom token of type `MyToken` in SOAP header `<foo:MyToken>` and that has a WS-Security policy that requires signing or encryption of some message parts (for example, the `<foo:MyToken>` header and SOAP body).
- It is not allowable to configure a proxy service that requires a custom token in header `<foo:MyToken>` and that also has a WS-Security policy that requires a SAML token or any other form of authentication.

Message-Level Security with .Net 2.0

This chapter explains how to configure message-level security between .NET 2.0 and Oracle Service Bus. The chapter contains the following topics:

- [Section 55.1, "Message-Level Security Between .NET 2.0 and Oracle Service Bus"](#)
- [Section 55.2, "What is .NET?"](#)
- [Section 55.3, "Message-Level Security Configuration in .NET For Oracle Service Bus Interoperability"](#)
- [Section 55.4, "Oracle Service Bus Configuration for Message-Level Security with .NET"](#)

55.1 Message-Level Security Between .NET 2.0 and Oracle Service Bus

You can set up Message-level security between the Microsoft .NET 2.0 framework and Oracle Service Bus. Message-level security applies security checks to a SOAP message after a Web services client establishes a connection with an Oracle Service Bus proxy service or business service and before the proxy service or business service processes the message.

55.2 What is .NET?

The .NET framework is a software component that you can add to the Microsoft Windows operating system. It provides pre-coded solutions to common program requirements, and manages the execution of programs written specifically for the framework.

55.3 Message-Level Security Configuration in .NET For Oracle Service Bus Interoperability

This section provides the steps that you need to perform for .NET 2.0 and for Oracle Service Bus to configure message-level security.

Caution: Before you perform these steps, you must follow the steps in [Chapter 52, "Configuring Message-Level Security for Web Services"](#) to configure inbound and outbound messaging for Oracle Service Bus.

To configure message-level security between .NET and Oracle Service Bus:

1. Verify that you completed the steps to configure inbound and outbound messaging for Oracle Service Bus. See the Warning above for instructions.
2. Download Web Service Enhancements (WSE) 3.0 from <http://msdn2.microsoft.com/en-us/webservices> and install it. WSE 3.0 is a SOAP extension managed API (Microsoft.Web.Services3.dll) that is compatible with the .NET 2.0 framework.
3. After you install WSE 3.0, you must enable the WSE features for your web application and enable WSE Soap Protocol Factory support. You can enable both these features using wizards in Visual Studio. See <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wse3.0/html/5a8f03b1-16ac-4c5c-9d9e-132a9c0b628a.asp> for details.

After you enable WSE 3.0, you will notice the following restrictions:

- WSE 3.0 no longer supports WS-Policy and therefore, WS-SecurityPolicy for configuration purposes, as it did in .NET 1.1 and WSE 2.0. WSE 3.0 supports only a proprietary policy configuration using the `wse3policyCache.config` file (or equivalent .NET code) that provides similar features to those in WSE 2.0. One consequence of this is that the WSDLs for the .NET web service no longer contain WS-Policy statements. On the other hand, Oracle Service Bus supports an Oracle WebLogic Server-proprietary format that is based on the assertions described in the December 18, 2002 version of the Web Services Security Policy Language (WS-SecurityPolicy) specification. In order to consume .NET WSDLs in Oracle Service Bus, you must incorporate the equivalent Oracle Service Bus proprietary version of WS-Policy in the WSDL.

The WSDL code sample in [Example 55-1](#) shows how to configure WS-Policy for message-level identity propagation, confidentiality, and integrity in Oracle Service Bus.

- WSE 3.0 provides policy configuration for a few Turnkey Security Assertions in the `wse3policyCache.config` file, which can be selected with a wizard in Visual Studio. The certificate that maps to providing message-level security (encryption and signing, for example) is `MutualCertificate10`. For details on configuring the `MutualCertificate10` Security Assertion, see <http://msdn2.microsoft.com/en-us/library/aa480581.aspx>.
- The WSE Soap Protocol Factory does not support security with SOAP 1.2. The generated client stubs using the Web Reference option in Visual Studio contain the security-enabled operations only if you select SOAP 1.1. Message-level security interoperability works only with SOAP 1.1.
- As with .NET 1.1 and WSE 2.0, you must disable automatic signing of WS-Addressing headers and timestamps that are configured by default. You must change some of the properties in the `wse3policyCache.config` file, as shown in the following example:

Default Config

```
<protection>
  <request signatureOptions="IncludeAddressing, IncludeTimestamp,
    IncludeSoapBody" encryptBody="true" />
  <response signatureOptions="IncludeAddressing, IncludeTimestamp,
    IncludeSoapBody" encryptBody="true" />
  <fault signatureOptions="IncludeAddressing, IncludeTimestamp,
    IncludeSoapBody" encryptBody="false" />
</protection>
```

Required Config

```
<protection>
  <request signatureOptions="IncludeSoapBody" encryptBody="true" />
  <response signatureOptions="IncludeAddressing, IncludeTimestamp,
IncludeSoapBody" encryptBody="true"
  />
  <fault signatureOptions="IncludeSoapBody" encryptBody="false" />
</protection>
```

- By default, WSE 3.0 expects the key wrapping algorithm to be OAEP. However, Oracle Service Bus uses the RSA15 algorithm. If the configuration remains as OAEP, the following exception appears:
Microsoft.Web.Services3.Security.SecurityFault: An unsupported signature or encryption algorithm was used
System.Exception: WSE3002: The receiver is expecting the key wrapping algorithm to be http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p, but the incoming message used http://www.w3.org/2001/04/xmlenc#rsa-1_5. You can change the key wrapping algorithm by configuring the security token manager.

To avoid this error, add the following configuration to the `web.config` file (on the .NET web service) and the `app.config` file (on the .NET client side) under the `<microsoft.web.services3>` `<security>` elements:

```
<binarySecurityTokenManager>
  <add valueType="http://docs.oasis-open.org/wss/2004/01/
oasis-200401-wss-x509-token-profile-1.0#X509v3";>
  <keyAlgorithm name="RSA15" />
  </add>
</binarySecurityTokenManager>
```

This configuration forces WSE to use RSA15 instead of OAEP.

- For Username Token Authentication, .NET provides a `usernameForCertificateSecurity` turnkey assertion that secures the communication channel between the client and the service at the message layer using the service's X.509 certificate. However, this certificate depends on the ability to reference `<EncryptedKey>` elements as security tokens, and enables the option for signature confirmation to correlate a response message with the request that prompted it.

An alternative for Username Token Authentication is the `.NET usernameOverTransportSecurity` turnkey assertion, which assumes that communication between the client and service will be secured at the transport layer. This approach is WS-Security compatible and supports message-level authentication over SSL. If you want to combine the `usernameOverTransportSecurity` turnkey assertion with other message-level security mechanisms, such as encryption and signing, you must write custom code in .NET.

55.4 Oracle Service Bus Configuration for Message-Level Security with .NET

Before you configure Oracle Service Bus, the following conditions must exist:

- A .NET client invokes an Oracle Service Bus proxy with a plain text message (for example, message-level security does not exist between the .NET client and the Oracle Service Bus proxy).
- Oracle Service Bus enforces outbound message-level security on the SOAP request.

Note: For cases where the .NET client has message-level security enabled, you can use Oracle Service Bus as a pass-through proxy.

To configure Oracle Service Bus for message-level security with .NET:

1. Change the encryption algorithm from `tripledes-cbc` to `aes256-cbc`:

```
<wssp:EncryptionAlgorithm URI="http://www.w3.org/2001/04/xmlenc#aes256-cbc" />
```

2. Change the `sign.xml` policy on the WSDL. This attribute is on the integrity assertion element.

```
<wssp:Integrity SignToken='false' ... >
...
</wssp:Integrity>
```

By default this value is true.

3. The .NET web service expects the WS-Addressing `<wsa:To>` element to contain its own URL. As the .NET client first invokes the Oracle Service Bus proxy, the `<wsa:To>` element is originally set to the Oracle Service Bus proxy URL. Change this URL to the URL of the .NET web service in the Oracle Service Bus proxy message flow, using a `Replace` action as shown in the following example:

Original URL

```
<wsa:To wsu:Id="To_1mbmRK4w0bo2Dz1z"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd";>http://localhost:7001/SecurityALSBProxy</
wsa:To>
```

URL after Replace Action

```
<wsa:To wsu:Id="To_1mbmRK4w0bo2Dz1z"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd";>http://localhost/SimpleSecurity/
SecurityService.asmx</wsa:To>
```

If you do not change this URL, the following error appears:

```
Microsoft.Web.Services3.Addressing.AddressingFault: Destination Unreachable
System.Exception: WSE846: The header must match the actor URI value of the web
service. The actor URI value can be explicitly specified using
SoapActorAttribute on the ASMX class. In the absence of the attribute, the
actor URI is assumed to be equal to the HTTP Request Url of the incoming
message. The header received contained
"http://localhost:7001/SecurityALSBProxy"; while the receiver is expecting
"http://localhost/SimpleSecurity/SecurityService.asmx";
```

4. The .NET client includes its own Timestamp elements to the SOAP header. Oracle Service Bus adds an additional Timestamp header that results in the following error:

Microsoft.Web.Services3.Security.SecurityFault: An error was discovered processing the header Microsoft.Web.Services3.Security.SecurityFormatException: WSE001: The input was not a valid Security element because it contains more than one Timestamp child element.

To solve this issue, use a Delete action to remove the original Timestamp elements that the .NET client adds in the message flow.

5. Add the CertificateRegistry certification path provider. You add this from the WLS Administration Console from **realm > Providers > Certification Path > New** and then select CertificateRegistry from the drop down.

Activate the change and restart the server.

After you restart the server, edit the CertificateRegistry provider you just created. From the Management tab add the following three certificates:

- The public certificate of Oracle Service Bus
- The public certificate of .NET
- The root agency (issuer of these certificates)

Note: One way to add the certificates is to import them from a jks store via the Migration tab. Provide the actual path of the identity store.

6. On the Configuration (Common) tab for the CertificateRegistry provider, select Current Builder to make it the current builder.

Save these changes. Then, activate and restart the server.

7. The WLS keystore requires these same certificates:

- The public certificate of Oracle Service Bus
- The public certificate of .NET
- The root agency (issuer of these certificates)

You configure the identity and trust keystores for an Oracle WebLogic Server instance on the server Configuration: Keystores page. To do this, see *Configure Identity and Trust* in the Oracle WebLogic Server online help.

55.4.1 Sample WSDL File

The sample WSDL in [Example 55–1](#) shows how to configure WS-Policy for message-level identity propagation, confidentiality, and integrity in Oracle Service Bus.

Example 55–1 Configuring WS-Policy for Message-Level Security

```
<?xml version='1.0' encoding='UTF-8'?>
<definitions name="SecureHello WorldServiceDefinitions" targetNamespace=
    "http://www.bea.com"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:s0="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
        wssecurity-utility-1.0.xsd"
    xmlns:s1="http://www.bea.com"
    xmlns:s2="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
    <wsp:UsingPolicy xmlns:n1="http://schemas.xmlsoap.org/wsdl/"
```

```

    n1:Required="true"/>
<wsp:Policy s0:Id="Encrypt.xml">
  <wssp:Confidentiality xmlns:wssp="http://www.bea.com/wls90/
    security/policy">
    <wssp:KeyWrappingAlgorithm URI="http://www.w3.org/2001/04/
      xmlenc#rsa-1_5"/>
    <wssp:Target>
      <wssp:EncryptionAlgorithm URI="http://www.w3.org/2001/
        04/xmlenc#aes256-cbc"/>
      <wssp:MessageParts Dialect="http://schemas.xmlsoap.org
        /2002/12/wsse#part">wsp:Body()
      </wssp:MessageParts>
    </wssp:Target>
    <wssp:KeyInfo>
      <wssp:SecurityToken TokenType="http://docs.oasis-open.
        org/wss/2004/01/oasis-200401-wss-x509-token-
        profile-1.0#X509v3"/>
      <wssp:SecurityTokenReference>
        <wssp:Embedded>
          <wsse:BinarySecurityToken EncodingType="http:
            //docs.oasis-open.org/wss/2004/
            01/oasis-200401-wss-soap-message
            -security-1.0#Base64Binary"
            ValueType="http://docs.oasis-open.org/
            wss/2004/01/oasis-200401-wss-x509
            -token-profile-1.0#X509v3"
            xmlns:wsse="http://docs.oasis-open.org/
            wss/2004/01/oasis-200401-wss-wssecurity-
            secext-1.0.xsd">MIIB7DCCAZYCEN+FH0mYRZU
            YPLiIutc01IIwDQYJKoZIhvcNAQEEBQAwTELMAK
            GA1UEBhMCVVMxEDA0BgNVBAGTB015U3RhdGUxDzA
            NBgNVBACTBk15VG93bjEXMBUGA1UEChMOTXlPcmd
            hbml6YXRpb24xGTAXBgNVBAsTEEZPUiBURVNUSU5
            HIE9OTFkxEzARBgNVBAMTCkNlcnRHZW5DQUiWWhc
            NMDYwNjA3MDQ0MDM2WhcNMjEwNjA4MDQ0MDM2WjB
            6MQswCQYDVQQGEwJVUzEQMA4GA1UECBYHTXlTdGF
            0ZTEPMA0GA1UEBxYGTXlUb3duMRcwFQYDVQQKF5
            NeU9yZ2FuaXphdGlvbjEZMBCGA1UECXYQRk9SIFR
            FU1RJTkcgT05MWTEUMBIGAlUEAxYLYmFuZ3BsdHc
            zazIwXDNANBgkqhkiG9w0BAQEFAANLADBIaKEAxv2
            nWByAF2Xr9wrb06ydrccqPt2VQa0xcwfdZZ6oG1j
            1TXq+G5/Q82v7CdxjyWUQBuaZduQx9wFCrAe/aWV
            pgQIDAQABMA0GCSqGSIb3DQEBBAUAA0EARbwf18w
            X915jL5reY+isriNF0EfUs5ck53WRNowiapJx2ea
            ZE03quksJgeJ0z0Hekkr/aTQnkMV1xIt1HxMKRw=
            =</wsse:BinarySecurityToken>
          </wssp:Embedded>
        </wssp:SecurityTokenReference>
      </wssp:KeyInfo>
    </wssp:Confidentiality>
  </wsp:Policy>
<wsp:Policy s0:Id="Auth.xml">
  <wssp:Identity xmlns:wssp="http://www.bea.com/wls90/security/
    policy">
    <wssp:SupportedTokens>
      <wssp:SecurityToken TokenType="http://docs.oasis-open.
        org/wss/2004/01/oasis-200401-wss-username-token
        -profile-1.0#UsernameToken">
      <wssp:UsePassword Type="http://docs.oasis-open.
        org/wss/2004/01/oasis-200401-wss-username

```

```

        -token-profile-1.0#PasswordText"/>
    </wssp:SecurityToken>
</wssp:SupportedTokens>
</wssp:Identity>
</wsp:Policy>
<wsp:Policy s0:Id="Sign.xml">
    <wssp:Integrity SignToken='false'
xmlns:wls="http://www.bea.com/wls90/security/
policy/wsee#part"xmlns:wssp="http://www.bea.com/wls90/
security/policy" xmlns:wsu="http://docs.oasis-open.org/wss
/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
<wssp:SignatureAlgorithm URI="http://www.w3.org/2000/09/
xmldsig#rsa-sha1"/>
<wssp:CanonicalizationAlgorithm URI="http://www.w3.org/
2001/10/xml-exc-c14n#"/>
<wssp:Target>
    <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09
/xmldsig#sha1"/>
    <wssp:MessageParts Dialect="http://www.bea.com/wls90/
security/policy/wsee#part">
        wls:SystemHeaders()
    </wssp:MessageParts>
</wssp:Target>
<wssp:Target>
    <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09
/xmldsig#sha1"/>
    <wssp:MessageParts Dialect="http://www.bea.com/wls90/
security/policy/wsee#part">
        wls:SecurityHeader(wsu:Timestamp)
    </wssp:MessageParts>
</wssp:Target>
<wssp:Target>
    <wssp:DigestAlgorithm URI="http://www.w3.org/2000/09/
xmldsig#sha1"/>
    <wssp:MessageParts Dialect="http://schemas.xmlsoap.
org/2002/12/wsse#part">
wsp:Body()
</wssp:MessageParts>
</wssp:Target>
<wssp:SupportedTokens>
    <wssp:SecurityToken IncludeInMessage="true" TokenType=
"http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-x509-token-profile-1.0#X509v3">
    <wssp:TokenIssuer>CN=CACERT,OU=FOR TESTING ONLY,
O=MyOrganization,L=MyTown,ST=MyState,C=US,1.2.
840.113549.1.9.1=#160f737570706f7274406265612e636
f6d,CN=Demo Certificate Authority Constraints,OU=
Security,O=BEA WebLogic,L=San Francisco,ST=
California,C=US,1.2.840.113549.1.9.1=#16107365637
572697479406265612e636f6d,CN=Demo Certificate
Authority Constraints,OU=Security,O=BEA WebLogic,
L=San Francisco,ST=California,C=US,CN=CertGenCAB,
OU=FOR TESTING ONLY,O=MyOrganization,L=MyTown,ST=
MyState,C=US,CN=Equifax Secure eBusiness CA-1,O=
Equifax Secure Inc.,C=US,CN=VeriSign Class 1
Public Primary Certification Authority - G3,OU=
(c)1999 VeriSign\, Inc. - For authorized use only,
OU=VeriSign Trust Network,O=VeriSign\, Inc.,C=US,
OU=VeriSign Trust Network,OU=(c) 1998 VeriSign\,
Inc. - For authorized use only,OU=Class 2 Public

```

```
Primary Certification Authority - G2,O=VeriSign\,
Inc.,C=US,CN=VeriSign Class 3 Public Primary
Certification Authority - G3,OU=(c) 1999
VeriSign\,Inc. - For authorized use only,OU=
VeriSign Trust Network,O=VeriSign\,Inc.,C=US,CN=
Entrust.net Client Certification Authority,OU=(c)
2000 Entrust.net Limited,OU=www.entrust.net/
GCCA_CPS incorp. by ref. (limits liab.),O=Entrust
.net,OU=Go Daddy Class 2 Certification Authority,
O=The Go Daddy Group\, Inc.,C=US,CN=GTE Cyber
Trust Global Root,OU=GTE CyberTrust Solutions\,
Inc., O=GTE Corporation,C=US,CN=Entrust.net
Secure Server Certification Authority,OU=(c) 2000
Entrust.net Limited,OU=www.entrust.net/SSL_CPS
incorp. by ref. (limits liab.),O=Entrust.net,OU=
Class 1 Public Primary Certification Authority,
O=VeriSign\, Inc.,C=US,1.2.840.113549.1.9.1=#161
9706572736f6e616c2d6261736963407468617774652e636
f6d,CN=Thawte Personal Basic CA,OU=Certification
Services Division,O=Thawte Consulting,L=Cape
Town, ST=Western Cape,C=ZA,OU=VeriSign Trust
Network, OU=(c) 1998 VeriSign\, Inc. - For
authorized use only,OU=Class 1 Public Primary
Certification Authority - G2,O=VeriSign\, Inc.,
C=US,CN=Entrust.net Secure Server Certification
Authority,OU=(c) 1999 Entrust.net Limited,OU=
www.entrust.net/CPS incorp. by ref.(limits iab.),
O=Entrust.net,C=US, 1.2.840.113549.1.9.1=#161c706
572736f6e616c2d667265656d61696c407468617774652e63
6f6d,CN=Thawte Personal Freemail CA,OU=
Certification Services Div,O=Thawte Consulting, L
=Cape Town,ST=Western Cape,C=ZA,OU=Class 3 Public
Primary Certification Authority,O=VeriSign\, Inc.
C=US,CN=GTE CyberTrust Root,O=GTE Corporation,C=
US,CN=VeriSign Class 2 Public Primary Certificate
Authority - G3,OU=(c) 1999 VeriSign\, Inc. - For
authorized use only,OU=VeriSign Trust Network,O=
VeriSign\,Inc.,C=US,1.2.840.113549.1.9.1=#1617736
5727665722d6365727473407468617774652e636f6d,CN=
Thawte Server CA,OU=Certification Services
Division,O=Thawte Consulting cc,L=Cape Town,ST=
Western Cape,C=ZA,OU=Equifax Secure Certificate
Authority,O=Equifax,C=US,1.2.840.113549.1.9.1=#16
1b706572736f6e616c2d7072656d69756d407468617774652
e636f6d,CN=Thawte Personal Premium CA,OU=
Certification Services Division,O=Thawte
Consulting,L=Cape Town,ST=Western Cape,C=ZA,1.2.
840.113549.1.9.1=#16197072656d69756d2d73657276657
2407468617774652e636f6d,CN=Thawte Premium Server
CA,OU=Certification Services Division,O=Thawte
Consulting cc,L=Cape Town,ST=Western Cape,C=ZA,
OU=VeriSign Trust Network,OU=(c) 1998 VeriSign\,
Inc. - For authorized use only,OU=Class 3 Public
Primary Certification Authority - G2,O=VeriSign\,
Inc.,C=US,CN=Entrust.net Certification Authority
(2048),OU=(c) 1999 Entrust.net Limited,OU=www
.entrust.net/CPS_2048 incorp. by ref. (limits
liab.),O=Entrust.net,1.2.840.113549.1.9.1=#1611
696e666f4076616c69636572742e636f6d,CN=http://www.
valicert.com/,OU=ValiCert Class 2 Policy
```



```

Validation Authority,O=ValiCert\, Inc.,L=Vali
cert Validation Network,CN=Baltimore CyberTrust
Root, OU=CyberTrust,O=Baltimore,C=IE,OU=Secure
Server Certification Authority,O=RSA Data
Security\, Inc.,C=US,CN=Entrust.net Client
Cert Authority,OU=(c) 1999 Entrust.net Limited,
OU=www.entrust.net/Client_CA_Info/CPS incorp. by
ref. limits liab.,O=Entrust.net,C=US,CN=GeoTrust
Global CA,O=GeoTrust Inc.,C=US,CN=GTE CyberTrust
Root 5,OU=GTE CyberTrust Solutions\, Inc.,O=GTE
Corporation,C=US,OU=Starfield Class 2
Certification Authority,O=Starfield
Technologies\, Inc.,C=US,CN=Equifax Secure
Global eBusiness CA-1,O=Equifax Secure Inc.,C=US,
CN=Baltimore CyberTrust Code Signing Root,OU=
CyberTrust,O=Baltimore,C=IE,OU=Class 2 Public
Primary Certification Authority,O=VeriSign\,
Inc.,C=US,OU=Equifax Secure eBusiness CA-2,O=
Equifax Secure,C=US,</wssp:TokenIssuer>
</wssp:SecurityToken>
</wssp:SupportedTokens>
</wssp:Integrity>
<wssp:MessageAge Age="60" xmlns:wssp="http://www.bea.com/wls90/
security/policy"/>
</wsp:Policy>
<types>
<xs:schema attributeFormDefault="unqualified" elementFormDefault=
"qualified" targetNamespace="http://www.bea.com" xmlns:s0="
http://www.bea.com" xmlns:s1="http://schemas.xmlsoap.org
/wsd/soap/" xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/
09/policy" xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="sayHello">
<xs:complexType>
<xs:sequence>
<xs:element name="s" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="sayHelloResponse">
<xs:complexType>
<xs:sequence>
<xs:element name="return" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
</types>
<message name="sayHello">
<part element="s1:sayHello" name="parameters"/>
</message>
<message name="sayHelloResponse">
<part element="s1:sayHelloResponse" name="parameters"/>
</message>
<portType name="SecureHelloWorldPortType" wsp:PolicyURIs="#Sign.xml
#Auth.xml #Encrypt.xml">
<operation name="sayHello" parameterOrder="parameters">
<input message="s1:sayHello"/>
<output message="s1:sayHelloResponse"/>
</operation>
</portType>

```

```
<binding name="SecureHelloWorldServiceSoapBinding" type="s1:
SecureHelloWorldPortType">
  <s2:binding style="document" transport="http://schemas.
xmlsoap.org/ soap/http"/>
  <operation name="sayHello">
    <s2:operation soapAction="" style="document"/>
    <input>
      <s2:body parts="parameters" use="literal"/>
    </input>
    <output>
      <s2:body parts="parameters" use="literal"/>
    </output>
  </operation>
</binding>
<service name="SecureHelloWorldService">
  <port binding="s1:SecureHelloWorldServiceSoapBinding"
name="SecureHelloWorldServicePort">
    <s2:address location="http://localhost:9111/
SecureHelloWorldService/SecureHelloWorld
Service"/>
  </port>
</service>
</definitions>
```

Part VIII

Appendix

This part contains miscellaneous development information and reference material.
Chapters include:

- [Appendix A, "Transport SDK UML Sequence Diagrams"](#)

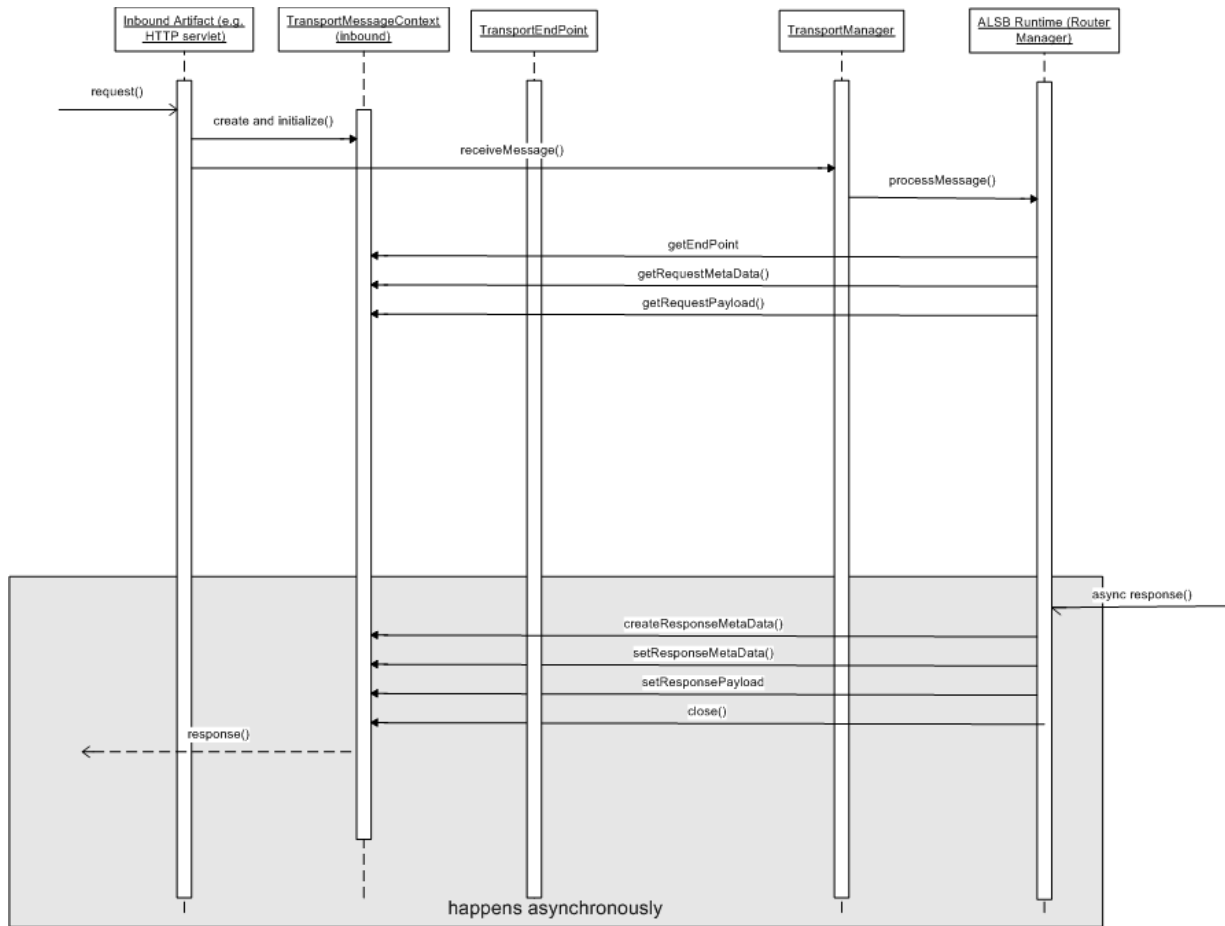
Transport SDK UML Sequence Diagrams

This chapter contains UML sequence diagrams that describe the flow of method calls through Oracle Service Bus runtime.

A.1 Oracle Service Bus Runtime Inbound Messages

The sequence diagram in [Figure A-1](#) describes the flow of inbound messages through Oracle Service Bus runtime.

First, an inbound artifact, such as an HTTP Servlet, intercepts a client request. The transport provider creates a data structure called `InboundTransportMessageContext`. The message context packages headers from the request into a metadata object, converting the payload from an HTTP stream into a specific Oracle Service Bus source object. The transport provider calls the transport manager to receive the message. The transport manager preprocesses the message and passes the message to the Oracle Service Bus runtime for processing. The Oracle Service Bus runtime asks for the message context's service, service version, and other information. It also asks about the metadata and payload, which are required for processing. The runtime asks the `MessageContext` to create the response metadata and the response payload, and then calls `close()`. The response is sent back to the client.

Figure A-1 Inbound Messages at Runtime

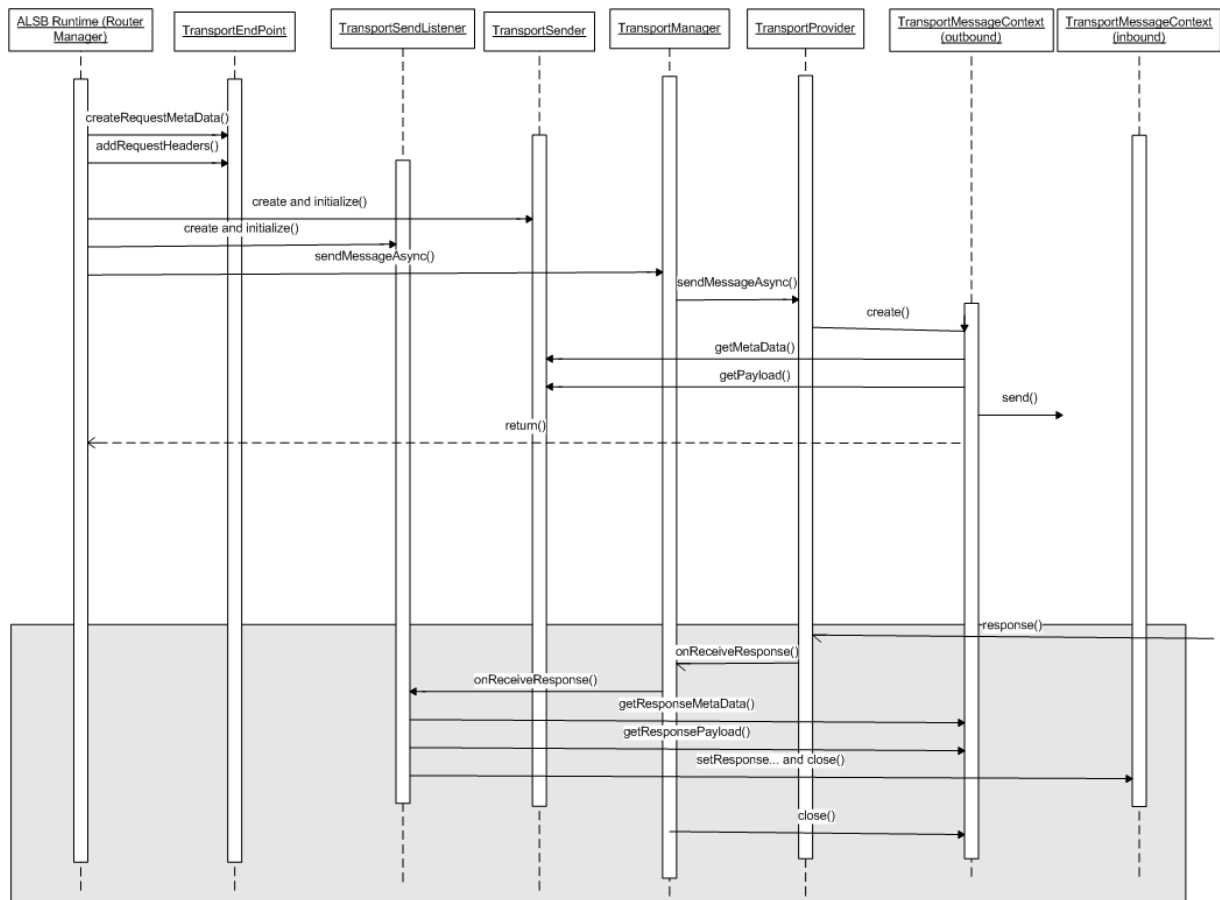
A.2 Oracle Service Bus Runtime Outbound Messages

The sequence diagram shown in [Figure A-2](#) describes the flow of outbound messages through Oracle Service Bus runtime.

The Oracle Service Bus runtime routes the message to an external service. The transport provider creates metadata for the request and creates a `TransportSender` object, which includes information about the payload and quality of service and retry information. Next, the provider calls `TransportManager` (the central hub for the transport subsystem) to send the message asynchronously. `TransportManager` calls the transport provider to send the message. The transport provider creates an `OutboundTransportMessageContext`. The transport provider then asks about the metadata and payload and other information and takes appropriate action. For example, for a JMS message, the transport provider uses the JMS API to populate the headers and the payload and calls the protocol-specific send operation.

When a response comes in, the transport provider calls the `TransportSendListener` object. Eventually the transport manager invokes the response pipeline. After pipeline actions are executed, the outbound endpoint is closed.

Figure A–2 Outbound Messages at Runtime



A.3 Design Time Service Registration

During service registration, a wizard guides you through a number of Oracle Service Bus Console pages. [Figure A–3](#) describes the service registration process. The basic steps include:

- Specifying the name of the service, the service binding type, and other information.
- Selecting from a list of transport providers (protocols). The Oracle Service Bus Console calls the transport manager to retrieve an object for each one of these entries in the list and gets a UI binding from each transport provider. This binding answers questions that the console requests, such as what is or is not supported. This step allows the console page to be populated with appropriate information.
- Entering transport-specific information. A transport provider specific form is generated automatically. The transport provider controls the contents of the page.
- Reviewing a summary page.

Finally, the transport provider is contacted and asked to validate the endpoint configuration and register the new endpoint. The endpoint is only created after activation occurs.

Figure A-3 Service Registration

