

**Oracle® Application Server Adapters for Files,
FTP, Databases, and Enterprise Messaging**

User's Guide

10g Release 3 (10.1.3.1.0)

B28994-02

February 2007

Oracle Application Server Adapters for Files, FTP, Databases, and Enterprise Messaging User's Guide, 10g Release 3 (10.1.3.1.0)

B28994-02

Copyright © 2006, 2007, Oracle. All rights reserved.

Primary Authors: Sheela Vasudevan, Rima Dave

Contributing Authors: Mark Kennedy, Deanna Bradshaw

Contributors: Navneet Singh, Shashi Suravarapu, Amandeep Mahajan, Bo Stern, Rahul Srivastava, Srimant Misra, Govind Pandey, Deepak Agarwal, Raghavendra Chandrashekar, Stephen Mcritchie, Michael Chiocca, Rod Fernandez, Sunil Gopal, Meera Srinivasan, Oracle BPEL Process Manager and Oracle Enterprise Service Bus development, product management, and quality assurance teams

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States 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, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software—Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Contents

Preface	xv
Audience	xv
Documentation Accessibility	xv
Related Documents	xvi
Conventions	xvi
 1 Introduction to Oracle Adapters for Files, FTP, Databases, and Enterprise Messaging	
1.1 Technology Adapters Integration with Oracle BPEL Process Manager	1-1
1.2 Technology Adapters Integration with Oracle Enterprise Service Bus	1-3
 2 Oracle Application Server Adapter for Files/FTP	
2.1 Introduction to File and FTP Adapters	2-1
2.1.1 File and FTP Adapters Features	2-2
2.1.1.1 File Formats	2-2
2.1.1.2 FTP Servers	2-2
2.1.1.3 Inbound and Outbound Interactions	2-2
2.1.1.4 File Debatching	2-3
2.1.1.5 Dynamic Outbound Directory and File Name Specification	2-3
2.1.1.6 Security	2-4
2.1.1.7 Proxy Support	2-4
2.1.1.8 No Payload Support	2-4
2.1.1.9 File-Based Triggers	2-5
2.1.1.10 High Availability	2-5
2.1.2 File and FTP Adapters Architecture	2-5
2.1.3 File and FTP Adapters Integration with Oracle BPEL Process Manager	2-5
2.1.4 File and FTP Adapters Integration with Oracle Enterprise Service Bus	2-6
2.2 File and FTP Adapters Concepts	2-7
2.2.1 File Adapter Read File Concepts	2-7
2.2.1.1 Inbound Operation	2-7
2.2.1.2 Inbound File Directory Specifications	2-8
2.2.1.3 File Matching and Batch Processing	2-11
2.2.1.4 File Polling	2-15
2.2.1.5 File Processing	2-17
2.2.1.6 Postprocessing	2-17

2.2.1.7	Native Data Translation	2-17
2.2.1.8	Error Handling.....	2-18
2.2.1.9	Guaranteed Delivery and Recovery from Server Failures	2-21
2.2.1.10	Inbound Service Name WSDL File	2-22
2.2.1.11	Inbound Header WSDL File.....	2-23
2.2.1.12	Synchronous File Reading Capabilities.....	2-24
2.2.2	File Adapter Write File Concepts	2-25
2.2.2.1	Outbound Operation.....	2-26
2.2.2.2	Outbound File Directory Creation	2-26
2.2.2.3	Native Data Translation	2-36
2.2.2.4	Outbound Error Handling	2-37
2.2.2.5	Outbound Service Name WSDL File	2-37
2.2.2.6	Outbound Header WSDL File	2-38
2.2.3	FTP Adapter for Get File Concepts	2-39
2.2.4	FTP Adapter for Put File Concepts	2-45
2.3	Using Secure FTP with the FTP Adapter.....	2-46
2.3.1	Secure FTP Overview	2-47
2.3.2	Installing and Configuring OpenSSL.....	2-48
2.3.3	Installing and Configuring vsftpd.....	2-49
2.3.4	Creating an Oracle Wallet	2-50
2.3.5	Setting Up the FTP Adapter.....	2-51
2.4	Using SFTP with the FTP Adapter	2-51
2.4.1	SFTP Overview.....	2-52
2.4.1.1	Encryption	2-52
2.4.1.2	Authentication	2-52
2.4.1.3	Integrity.....	2-52
2.4.1.4	Data Compression	2-53
2.4.2	Install and Configure OpenSSH for Windows.....	2-53
2.4.3	Set up the FTP Adapter for SFTP	2-54
2.4.3.1	Configuring the FTP Adapter for Password Authentication.....	2-56
2.4.3.2	Configuring the FTP Adapter for Public Key Authentication	2-57
2.5	Configuring the FTP Adapter for HTTP Proxy	2-59
2.5.1	Configuring for Plain FTP Mode.....	2-59
2.5.1.1	Proxy Definition File	2-60
2.5.2	Configuring for SFTP Mode.....	2-62
2.6	File and FTP Adapters Use Cases for Oracle BPEL Process Manager	2-63
2.6.1	File Adapter Use Cases for Oracle BPEL Process Manager.....	2-63
2.6.1.1	File Reading	2-64
2.6.1.2	Message Debatching	2-64
2.6.1.3	Reading Delimited Content Files	2-64
2.6.1.4	Reading Positional (Fixed Length) Content Files	2-64
2.6.1.5	File Writing.....	2-64
2.6.2	FTP Adapter Use Case for Oracle BPEL Process Manager.....	2-65
2.7	File and FTP Adapters Use Case for Oracle Enterprise Service Bus	2-65
2.7.1	Prerequisites	2-65
2.7.2	Creating an Oracle Enterprise Service Bus Application and Project.....	2-65
2.7.3	Importing the Schema Definition (.XSD) Files	2-66

2.7.4	Creating the Inbound File Adapter Service	2-66
2.7.5	Creating the Outbound FTP Adapter Service	2-67
2.7.6	Creating the Routing Rule	2-67
2.7.7	Registering Services with Oracle Enterprise Service Bus	2-68
2.7.8	Run-Time Task	2-68

3 Oracle Application Server Adapter for Advanced Queuing

3.1	Introduction to the AQ Adapter	3-1
3.1.1	AQ Adapter Features	3-1
3.1.1.1	Enqueue-Specific Features (Message Production)	3-2
3.1.1.2	Dequeue and Enqueue Features	3-3
3.1.1.3	Supported ADT Payload Types	3-4
3.1.1.4	Native Format Builder Wizard	3-5
3.1.2	AQ Adapter Integration with Oracle BPEL Process Manager	3-6
3.1.3	AQ Adapter Integration with Oracle Enterprise Service Bus	3-6
3.2	Use Cases for the AQ Adapter	3-6
3.2.1	Adapter Configuration Wizard Walkthrough	3-6
3.2.1.1	Meeting Prerequisites	3-7
3.2.1.2	Creating a New BPEL Project	3-7
3.2.1.3	Adding a Partner Link	3-9
3.2.1.4	Generated WSDL File	3-19
3.2.2	Dequeuing and Enqueuing Object and ADT Payloads	3-20
3.2.3	Dequeuing One Column of the Object/ADT Payload	3-21
3.2.4	Processing Large Numbers of Messages	3-21
3.2.5	Using Correlation ID for Filtering Messages During Dequeue	3-22
3.2.6	Enqueuing and Dequeuing from Multisubscriber Queues	3-22
3.2.7	Rule-Based Subscription for Multiconsumer Queues	3-23
3.2.8	General Tips when working with AQ Adapter	3-24
3.2.8.1	Tips when working with AQ Adapter	3-24
3.2.8.2	Tips when working with JMS Adapter For AQ	3-24
3.3	AQ Adapter Use Cases for Oracle BPEL Process Manager	3-24
3.3.1	Using AQ Headers in a BPEL Process	3-25
3.3.2	Header Variables in JDeveloper BPEL Designer	3-25
3.3.3	Configuring a Message Rejection Handler for Data Errors	3-26
3.3.4	Example connectionString for RAC	3-26
3.4	AQ Adapter Use Cases for Oracle Enterprise Service Bus	3-27
3.4.1	Meeting Prerequisites	3-27
3.4.2	Creating a New ESB Project	3-27
3.4.3	Creating Inbound AQ Adapter	3-28
3.4.4	Creating Outbound AQ Adapter	3-36
3.4.5	Configuring Routing Service	3-40
3.4.6	Checking the ESB Console	3-47
3.4.7	Checking Execution in the ESB Control	3-48

4 Oracle Application Server Adapter for Databases

4.1	Introduction to the Database Adapter	4-1
-----	--	-----

4.1.1	Database Adapter Features	4-1
4.1.1.1	Querying over Multiple Tables	4-3
4.1.2	Design Overview	4-7
4.1.3	Database Adapter Integration with Oracle BPEL Process Manager	4-8
4.1.4	Database Adapter Integration with Oracle Enterprise Service Bus	4-8
4.2	Database Adapter Concepts	4-8
4.2.1	Relational-to-XML Mapping	4-9
4.2.1.1	Relational Types to XML Schema Types.....	4-12
4.2.1.2	Mapping Any Relational Schema to Any XML Schema	4-13
4.2.2	SQL Operations as Web Services.....	4-13
4.2.2.1	DML Operations	4-13
4.2.2.2	Polling Strategies	4-15
4.3	Database Adapter Use Case for Oracle BPEL Process Manager	4-23
4.3.1	Use Case: One.....	4-25
4.3.2	Starting the Adapter Configuration Wizard.....	4-25
4.3.3	Connecting to a Database	4-26
4.3.4	Selecting the Operation Type.....	4-26
4.3.5	Selecting and Importing Tables	4-28
4.3.6	Defining Primary Keys	4-28
4.3.7	Creating Relationships.....	4-29
4.3.7.1	What Happens When Relationships Are Created or Removed	4-31
4.3.7.2	Different Types of One-to-One Mappings.....	4-31
4.3.8	Creating the Object Filter.....	4-32
4.3.9	Defining a WHERE Clause.....	4-32
4.3.10	Choosing an After-Read Strategy.....	4-34
4.3.10.1	Delete the Rows that Were Read	4-35
4.3.10.2	Update a Field in the Table (Logical Delete)	4-35
4.3.10.3	Update a Sequencing Table.....	4-36
4.3.11	Internal Processes at Design Time.....	4-37
4.3.11.1	Importing Tables.....	4-37
4.3.11.2	Creating Relationships.....	4-37
4.3.11.3	Generating Design-Time Artifacts	4-38
4.3.12	Use Case: Two	4-38
4.3.12.1	Prerequisites	4-38
4.3.12.2	Creating a New BPEL Project	4-38
4.3.12.3	Starting the Adapter Configuration Wizard.....	4-39
4.4	Database Adapter Use Cases for Oracle Enterprise Service Bus	4-45
4.4.1	Prerequisites	4-46
4.4.2	Creating a New ESB Project	4-46
4.4.3	Creating Inbound Database Adapter.....	4-47
4.4.4	Creating an Outbound Database Adapter Service	4-56
4.4.5	Configuring Routing Service	4-60
4.4.6	Checking the ESB Console.....	4-65
4.4.7	Checking Execution in the ESB Control	4-66
4.5	Advanced Configuration	4-67
4.5.1	The TopLink Workbench Project.....	4-67
4.5.1.1	Deleting a Descriptor	4-68

4.5.1.2	Returning Partial Objects When Querying	4-68
4.5.1.3	Renaming a Mapping.....	4-70
4.5.1.4	Configuring Offline Database Tables	4-70
4.5.2	Relational-to-XML Mappings (toplink_mappings.xml)	4-71
4.5.3	The Service Definition (WSDL).....	4-73
4.5.3.1	DBWriteInteractionSpec	4-74
4.5.3.2	DBReadInteractionSpec	4-75
4.5.3.3	DBActivationSpec.....	4-76
4.5.4	XML Schema Definition (XSD)	4-78
4.5.5	Deployment	4-79
4.5.5.1	How the Database Adapter Gets Connection Information	4-79
4.5.5.2	Out-of-the-box Deployment.....	4-80
4.5.5.3	Production Deployment	4-81
4.5.5.4	Advanced Properties of the oc4j-ra.xml File.....	4-83
4.5.5.5	Comparison: Pre-10.1.3 and Post-10.1.3	4-87
4.5.6	Performance.....	4-88
4.5.6.1	Use Cases for Performance.....	4-88
4.5.6.2	Performance Hit List	4-88
4.5.6.3	Outbound Write: Should You Use Merge, Write, or Insert?	4-89
4.5.6.4	The TopLink Cache: When Should You Use It?	4-90
4.5.6.5	Existence Checking.....	4-90
4.5.6.6	Inbound Polling: maxRaiseSize	4-90
4.5.6.7	Inbound Polling: Choosing a Polling Strategy	4-90
4.5.6.8	Relationship Reading (Batch Attribute and Joined Attribute Reading)	4-90
4.5.6.9	Connection Pooling	4-90
4.5.6.10	Inbound Distributed Polling.....	4-91
4.5.7	detectOmissions Feature	4-92
4.5.8	OutputCompletedXml Feature.....	4-94
4.6	Third-Party JDBC Driver and Database Connection Configuration.....	4-95
4.6.1	Using a Microsoft SQL Server.....	4-96
4.6.1.1	MS JDBC Driver	4-96
4.6.1.2	DataDirect Driver	4-96
4.6.2	Using an IBM DB2 Database	4-98
4.6.2.1	DataDirect Driver	4-98
4.6.2.2	JT400 Driver (AS400 DB2)	4-98
4.6.3	Using a Sybase Database	4-98
4.6.3.1	jconn Driver	4-98
4.6.3.2	DataDirect Driver	4-98
4.6.4	Using an InterSystems Caché Database	4-98
4.6.5	Using a MySQL 4 Database.....	4-98
4.6.6	Summary of Third-Party and Oracle Lite Database Connection Information.....	4-99
4.6.7	Location of JDBC Driver JAR Files and Setting the Class Path.....	4-100
4.7	Stored Procedure and Function Support.....	4-100
4.7.1	Design Time: Using the Adapter Configuration Wizard.....	4-100
4.7.1.1	Using Top-Level Standalone APIs	4-101
4.7.1.2	Using Packaged APIs and Overloading.....	4-105
4.7.2	Design Time: Using the Command-Line Utility	4-107

4.7.2.1	Common Command-Line Functionality	4-107
4.7.2.2	Generated Output.....	4-108
4.7.2.3	Supported Third-Party Database	4-108
4.7.2.3.1	Microsoft SQL Server 2000 and 2005	4-108
4.7.2.3.2	IBM DB2 v8.2	4-110
4.7.3	Design Time: WSDL and XSD Generation.....	4-111
4.7.3.1	The WSDL–XSD Relationship.....	4-112
4.7.3.2	Supported Primitive Data Types.....	4-113
4.7.3.3	Generated XSD Attributes.....	4-113
4.7.3.4	User-Defined Types.....	4-114
4.7.3.5	Complex User-Defined Types	4-116
4.7.3.6	Object Type Inheritance.....	4-116
4.7.3.7	Object References.....	4-117
4.7.3.8	Referencing Types in Other Schemas	4-118
4.7.4	Run Time: Before Stored Procedure Invocation	4-118
4.7.4.1	Value Binding.....	4-118
4.7.4.2	Data Type Conversions	4-120
4.7.5	Run Time: After Stored Procedure Invocation.....	4-121
4.7.5.1	Data Type Conversions	4-121
4.7.5.2	Null Values	4-121
4.7.5.3	Function Return Values	4-122
4.7.6	Runtime: Common Third-Party Database Functionality.....	4-122
4.7.7	Advanced Topics	4-122
4.7.7.1	Support for REF CURSOR.....	4-122
4.7.7.1.1	Design Time	4-123
4.7.7.1.2	Runtime.....	4-124
4.7.7.2	Support for PL/SQL Boolean, PL/SQL Record, and PL/SQL Table Types...	4-124
4.7.7.2.1	Default Clauses in Wrapper Procedures.....	4-127
4.8	Use Case for Creating and Configuring a Stored Procedure in JDeveloper BPEL Designer... 4-127	
4.8.1	Creating a Stored Procedure	4-128
4.8.2	Creating a New BPEL Project	4-128
4.8.3	Creating a Partner Link.....	4-130
4.8.4	Creating an Invoke Activity	4-134
4.8.5	Creating an Initial Assign Activity.....	4-137
4.8.6	Creating a Second Assign Activity.....	4-139
4.8.7	Validating, Compiling, and Deploying the Greeting Process.....	4-140
4.8.8	Running the Greeting Process.....	4-140

5 Oracle Application Server Adapter for Java Message Service

5.1	Introduction to the JMS Adapter	5-1
5.1.1	JMS Adapter Features	5-1
5.1.2	JMS Adapter Integration with Oracle BPEL Process Manager	5-2
5.1.3	JMS Adapter Integration with Oracle Enterprise Service Bus	5-3
5.2	JMS Adapter Use Cases for Oracle BPEL Process Manager	5-3
5.2.1	Case One: Configuring a JMS Adapter.....	5-3
5.2.1.1	Concepts.....	5-3

5.2.1.2	Using the Adapter Configuration Wizard to Configure a JMS Adapter.....	5-5
5.2.1.3	Generated WSDL File	5-15
5.2.1.4	oc4j-ra.xml file	5-16
5.2.1.5	Produce Message Procedure	5-17
5.2.1.6	Configuring for OJMS.....	5-18
5.2.1.7	Configuring for OC4J JMS.....	5-20
5.2.1.8	Configuring for TIBCO JMS.....	5-20
5.2.1.8.1	Direct Connection.....	5-21
5.2.1.9	Configuring for IBM Websphere JMS	5-22
5.2.2	Case Two: MQSeries Queue Integration Through the OracleAS Adapter for JMS	5-23
5.2.2.1	Configuring the MQSeries Consumer Service	5-23
5.2.2.2	Configuring the MQSeries Producer Service	5-26
5.2.2.3	Configuring an End-to-End BPEL Process	5-29
5.3	JMS Adapter Use Cases for Oracle Enterprise Service Bus	5-33
5.3.1	Meeting Prerequisites.....	5-34
5.3.2	Preparing Database accounts	5-34
5.3.3	Creating Stored Procedure	5-35
5.3.4	Creating Destinations in AQ.....	5-35
5.3.5	Creating a New ESB Project	5-35
5.3.6	Creating Inbound JMS Adapter.....	5-36
5.3.7	Creating Outbound Database Adapter Service.....	5-47
5.3.8	Configuring Routing Service	5-53
5.3.9	Promoting the Projects from Development to Production	5-57
5.3.10	Configuring a JMS Adapter in Managed Mode.....	5-57
5.3.11	JMS Adapter: Configuring Database Resource Provider.....	5-58
5.3.12	JMS Adapter: Configuring JMS Destinations in oc4j-ra.xml	5-58
5.3.13	Database Adapter: Configuring Database Destinations in oc4j-ra.xml	5-58
5.3.14	Database Adapter: Configuring data-sources.xml	5-59
5.3.15	Restart Server	5-59
5.3.16	Registering With ESB	5-59
5.3.17	Checking the ESB Console.....	5-59
5.3.18	Sending a JMS Message to Trigger the New Service.....	5-60
5.3.19	Checking Execution in the ESB Control	5-61

6 Oracle Application Server Adapter for Native MQSeries

6.1	MQSeries Message Queuing Concepts	6-1
6.1.1	MQSeries Concepts.....	6-3
6.2	Introduction to Native MQSeries Adapter.....	6-4
6.2.1	The Need for MQSeries Adapter	6-4
6.2.2	MQSeries Adapter Integration with Oracle BPEL Process Manager	6-5
6.2.3	MQSeries Adapter Integration with Oracle Enterprise Service Bus	6-5
6.3	MQSeries Adapter Concepts	6-6
6.3.1	Messaging Scenarios	6-6
6.3.1.1	Enqueue Message	6-6
6.3.1.2	Dequeue Message.....	6-9
6.3.1.3	Request-Response (Oracle BPEL Process Manager as a Client)	6-11
6.3.1.4	Synchronous Request-Response (Oracle BPEL Process Manager as Server) ...	6-16

6.3.1.5	Asynchronous Request-Response (Oracle BPEL Process Manager as Server).	6-19
6.3.1.6	Request-Response Synchronous (Oracle Enterprise Service Bus as Server).....	6-21
6.3.2	Message Properties	6-22
6.3.2.1	Messages Types	6-22
6.3.2.2	Message Format	6-23
6.3.2.3	Message Expiry	6-23
6.3.2.4	Message Priority	6-23
6.3.2.5	Message Persistence	6-24
6.3.3	Correlation Schemas.....	6-24
6.3.4	Distribution List Support.....	6-25
6.3.5	Report Messages	6-25
6.3.6	Filter-by Criteria.....	6-26
6.3.7	Message Delivery Failure Options	6-26
6.3.8	Message Segmentation.....	6-26
6.3.9	Message Grouping.....	6-27
6.3.10	Integration with CICS	6-29
6.4	Configuring the MQSeries Adapter	6-34
6.4.1	Add the com.ibm.mq.jar to the MQSeries Adapter Classpath.....	6-34
6.4.2	Modify the oc4j-ra.xml File.....	6-35
6.5	MQSeries Adapter Use Cases for Oracle BPEL Process Manager	6-36
6.5.1	Message Enqueue/Dequeue	6-36
6.5.1.1	Prerequisites	6-37
6.5.1.2	Creating the Inbound Adapter Service	6-37
6.5.1.3	Creating the Outbound Adapter Service	6-40
6.5.1.4	Creating the Receive Activity	6-42
6.5.1.5	Creating the Invoke Activity.....	6-43
6.5.1.6	Creating the Assign Activity.....	6-43
6.5.1.7	Run-Time	6-45
6.5.2	Synchronous Request-Response	6-46
6.5.2.1	Prerequisites	6-46
6.5.2.2	Creating the Synchronous Request-Response Service	6-47
6.5.2.3	Creating the Receive Activity	6-48
6.5.2.4	Creating the Reply Activity	6-49
6.5.2.5	Creating the Transform Activity	6-50
6.5.2.6	Run-Time	6-51
6.5.3	Demonstrations and Samples	6-51

7 Native Format Builder Wizard

7.1	Creating Native Schema Files with the Native Format Builder Wizard.....	7-1
7.1.1	Supported Formats	7-2
7.1.1.1	Delimited	7-3
7.1.1.2	Fixed Length (Positional)	7-3
7.1.1.3	DTD	7-3
7.1.1.4	COBOL Copybook.....	7-3
7.1.2	Native Format Builder Wizard Windows	7-7
7.2	Understanding Native Schema	7-8
7.2.1	Use Cases for the Native Format Builder	7-8

7.2.1.1	Defining a Comma-Separated Value File Structure	7-8
7.2.1.2	Defining a * Separated Value File Structure	7-10
7.2.1.3	Defining a Fixed-Length Structure	7-10
7.2.1.4	Defining a More Complex Structure - Invoice	7-11
7.2.1.5	Using the Native Format Translator for Removing or Adding Namespaces to XML with No Namespace 7-14	
7.2.1.6	COBOL Copybook.....	7-14
7.2.2	Native Schema Constructs.....	7-22
7.2.2.1	Defining Fixed-Length Data	7-22
7.2.2.2	Defining Terminated Data	7-25
7.2.2.3	Defining Surrounded Data.....	7-27
7.2.2.4	Defining Lists	7-28
7.2.2.5	Defining Arrays	7-30
7.2.2.6	Conditional Processing.....	7-36
7.2.2.7	Defining Dates	7-43
7.2.2.8	Using Variables.....	7-45
7.2.2.9	Defining Prefixes and Suffixes.....	7-46
7.2.2.10	Defining Skipping Data	7-47
7.2.2.11	Defining fixed and default values	7-48
7.2.2.12	Defining write	7-49
7.2.2.13	Defining LookAhead.....	7-50
7.2.2.14	Defining outboundHeader	7-52
7.3	Native Schema Constructs.....	7-53

A Troubleshooting and Workarounds

A.1	Troubleshooting the Oracle Application Server Adapter for Databases.....	A-1
A.1.1	Could Not Create TopLink Session Exception.....	A-2
A.1.2	Could Not Find Adapter for eis/DB/ <i>my_connection</i>	A-3
A.1.3	Changes Through TopLink Workbench.....	A-3
A.1.4	Redeploying from the Command Line.....	A-3
A.1.5	Cannot Change Customers_table.xsd.....	A-3
A.1.6	No Target Foreign Keys Error.....	A-3
A.1.7	No Primary Key Exception.....	A-4
A.1.8	dateTime Conversion Exceptions.....	A-6
A.1.9	Issues with Oracle DATE.....	A-6
A.1.10	TIMESTAMP Datatype Is Not Supported for a Microsoft SQL Server Database	A-7
A.1.11	Handling a Database Adapter Fault	A-8
A.1.12	Table Not Found: SQL Exception.....	A-8
A.1.13	Switching from a Development Database to a Production Database	A-8
A.1.14	Only One Employee Per Department Appears.....	A-9
A.1.15	Outbound SELECT on a CHAR(X) or NCHAR Column Returns No Rows.....	A-9
A.1.16	ORA-00932: Inconsistent Datatypes Exception Querying CLOBs.....	A-9
A.1.17	ORA-17157: 4K/32K Driver Limit with CLOBs and BLOBs	A-10
A.1.18	MERGE Sometimes Does UPDATE Instead of INSERT, or Vice Versa.....	A-10
A.1.19	Message Loss with the MERGE Invoke Operation.....	A-11
A.1.20	Integrity Violation Occurs with Delete or DeletePollingStrategy	A-12
A.1.21	Some Queried Rows Appear Twice or Not at All in the Query Result	A-13

A.1.22	Importing a Same-Named Table, with Same Schema Name, but Different Databases	A-13
A.1.23	Problems Creating a Relationship Manually for a Composite Primary Key	A-13
A.1.24	Must Fully Specify Relationships Involving Composite Primary Keys	A-14
A.1.25	Database Adapter Throws an Exception When Using a BFILE.....	A-14
A.1.26	During Design-Time, Wizard Does Not Allow Deletion of a Table	A-14
A.1.27	Changes to JDeveloper Project Are Made Even If Wizard Is Cancelled	A-14
A.1.28	Problems Removing a Relationship, Then Adding a New Relationship with the Same Name	A-14
A.1.29	Problems Importing Third-Party Database Tables.....	A-15
A.1.30	Problems Importing Object Tables.....	A-16
A.1.31	Relationships Not Autogenerated When Tables Are Imported Separately.....	A-16
A.1.32	Primary Key Is Not Saved	A-16
A.1.33	Table Column Name Is a Java Keyword	A-17
A.1.34	Catching a Database Exception.....	A-18
A.1.35	Connection Settings Error: Too Many Transactions.....	A-18
A.1.36	ORA-01747: invalid user.table.column, table.column, or column specification, When Using SELECT FOR UPDATE	A-19
A.1.37	Update Only Sometimes Performs Inserts/Deletes for Child Records.....	A-19
A.2	Troubleshooting the Oracle Application Server Adapter for Databases When Using Stored Procedures	A-19
A.2.1	Design Time: Unsupported or Undefined Parameter Types	A-20
A.2.2	Design Time: Referencing User-Defined Types in Other Schemas	A-20
A.2.3	Run Time: Parameter Mismatches	A-21
A.2.4	Run Time: Stored Procedure Not Defined in the Database.....	A-22
A.2.5	Configuring Multiple Adapters in the Inbound Direction Using Correlation Sets	A-23
A.3	Troubleshooting the Oracle Application Server Adapter for Files/FTP	A-23
A.3.1	Changing Logical Names with the Adapter Configuration Wizard.....	A-23
A.3.2	Creating File Names with Spaces with the Native Format Builder Wizard	A-23
A.3.3	Common User Errors	A-23
A.4	Troubleshooting the Oracle Application Server Adapter for Advanced Queuing.....	A-25
A.4.1	Inbound Errors.....	A-25
A.4.1.1	JNDI Lookup Failed	A-25
A.4.1.2	During Initialization, I/O Exception: Network Adapter Did Not Establish the Connection	A-26
A.4.1.3	Incorrect Username/Password	A-27
A.4.1.4	Queue Not Found.....	A-27
A.4.1.5	User Does Not Have DBMS_AQIN Privileges, Which Are Required by the AQ Java API	A-27
A.4.1.6	Translation Error	A-28
A.4.1.7	Subscriber Already Exists When Using MessageRuleSelector	A-28
A.4.2	Outbound Errors.....	A-29
A.4.2.1	JNDI Lookup Failed	A-29
A.4.2.2	I/O Exception: Network Adapter Could Not Establish the Connection	A-29
A.4.2.3	Queue Not Found.....	A-30
A.4.2.4	Incorrect Username/Password	A-30
A.4.2.5	User Does Not Have DBMS_AQIN Privileges, Which Are Required by the AQ Java API	A-31

A.4.2.6	Translation Error	A-31
A.4.3	JDeveloper BPEL Designer Errors.....	A-31
A.4.4	Translation Error.....	A-33
A.4.5	Other Problems	A-34
A.5	Troubleshooting the Oracle Application Server Adapter for Java Message Service (JMS)	A-36

Index

Preface

This guide describes how to use the technology adapters that are provided with Oracle BPEL Process Manager and Oracle Enterprise Service Bus.

This preface contains the following topics:

- [Audience](#)
- [Documentation Accessibility](#)
- [Related Documents](#)
- [Conventions](#)

Audience

Oracle Application Server Adapters for Files, FTP, Databases, and Enterprise Messaging User's Guide is intended for anyone who is interested in using these adapters.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. 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.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Related Documents

For more information, see the following documents in the Oracle Other Product One Release 7.0 documentation set or in the Oracle Other Product Two Release 6.1 documentation set:

- *Oracle Application Server Adapter Concepts Guide*
- *Oracle Application Server Administrator's Guide*
- *Oracle Enterprise Service Bus Developer's Guide*
- *Oracle BPEL Process Manager Developer's Guide*

Conventions

The following text conventions are used in this document:

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

Introduction to Oracle Adapters for Files, FTP, Databases, and Enterprise Messaging

This chapter describes the file, FTP, database, and enterprise messaging adapters that are provided with Oracle BPEL Process Manager and Oracle Enterprise Service Bus. The adapters enable you to integrate BPEL processes or ESB services to file systems, FTP servers, database tables, database queues (advanced queues, or AQ), message queues, Java Message Services (JMS), and Oracle Applications. See *Oracle BPEL Process Manager Developer's Guide* or *Oracle Enterprise Service Bus Developer's Guide* for information about BPEL processes or ESB services .

This chapter contains the following topics:

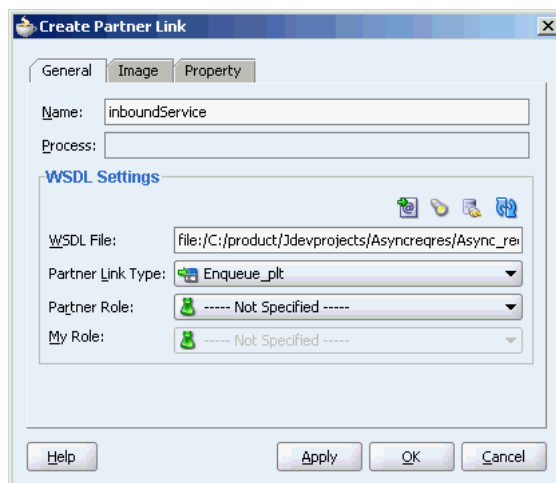
- [Section 1.1, "Technology Adapters Integration with Oracle BPEL Process Manager"](#)
- [Section 1.2, "Technology Adapters Integration with Oracle Enterprise Service Bus"](#)

See *Oracle Application Server Adapter Concepts* for information about application and mainframe adapters.

1.1 Technology Adapters Integration with Oracle BPEL Process Manager

From the Partner Link dialog box in Oracle BPEL Process Manager, shown in [Figure 1-1](#), you can access the adapters that are provided with Oracle BPEL Process Manager.

Figure 1-1 Partner Link dialog box



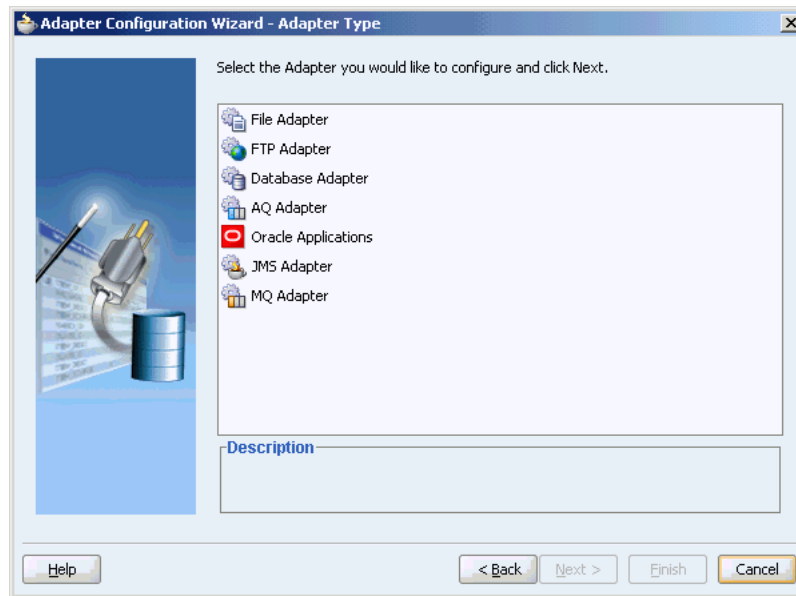
Click the **Define Adapter Service** icon, shown in [Figure 1-2](#), to access the Adapter Configuration Wizard.

Figure 1-2 Defining an Adapter



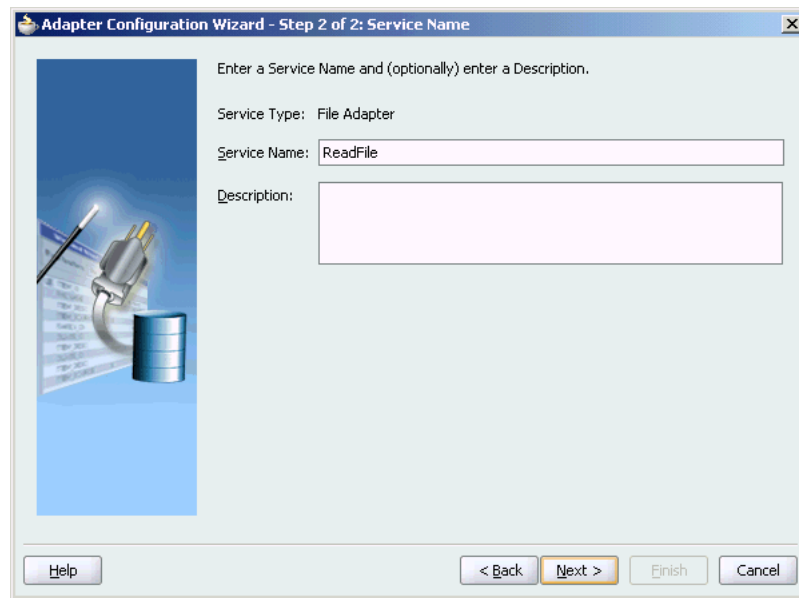
This wizard enables you to configure the types of adapters shown in [Figure 1-3](#) for use with BPEL processes.

Figure 1-3 Adapter Types



When you select an adapter type, the Service Name window shown in [Figure 1-4](#) prompts you to enter a name. For this example, **File Adapter** was selected in [Figure 1-3](#). When the wizard completes, a WSDL file by this service name appears in the **Applications Navigator** for the BPEL process (for this example, named **ReadFile.wsdl**). This file includes the adapter configuration settings you specify with this wizard. Other configuration files (such as header files and files specific to the adapter) are also created and display in the **Applications Navigator**.

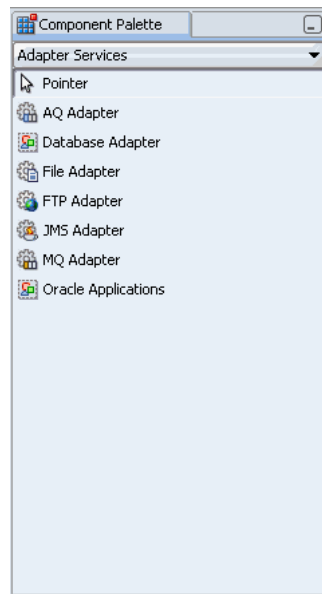
See *Oracle Application Server Adapter for Oracle Applications User's Guide* for information on using the Oracle Applications adapter listed in [Figure 1-3](#).

Figure 1–4 Adapter Service Name

The Adapter Configuration Wizard windows that appear after the Service Name window are based on the adapter type you selected. These configuration windows and the information you must provide are described in later chapters of this guide.

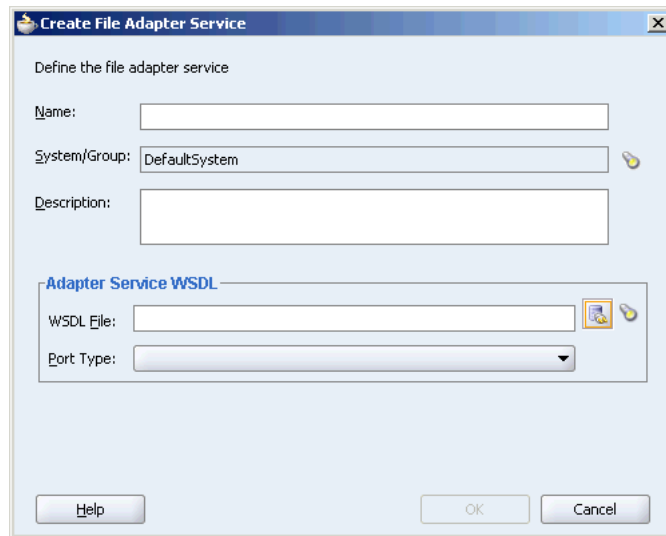
1.2 Technology Adapters Integration with Oracle Enterprise Service Bus

You can access the adapters that are provided with Oracle Enterprise Service Bus through the Component Palette shown in [Figure 1–5](#). When you select **Adapter Services** in Component Palette, all adapters supported in Oracle Enterprise Service Bus are displayed as shown in [Figure 1–5](#).

Figure 1–5 Component Palette

You can drag and drop any adapter service in the design area. [Figure 1–6](#) displays the Create File Adapter Service dialog box which starts when you drag a File adapter service to the design area. You can specify the adapter service name here.

Figure 1–6 Create File Adapter Service dialog box

The image shows a Windows-style dialog box titled "Create File Adapter Service". It contains several input fields: "Name:" with a text box, "System/Group:" with a text box containing "DefaultSystem" and a key icon, "Description:" with a text box, and a section titled "Adapter Service WSDL" containing "WSDL File:" with a text box and a file icon, and "Port Type:" with a dropdown menu. At the bottom are "Help", "OK", and "Cancel" buttons.

To configure the adapter service, click the **Configure adapter service wsdl** icon shown highlighted in [Figure 1–6](#). This starts the Adapter Configuration wizard. When the wizard completes, a WSDL file by the service name appears in the **Applications Navigator** for the ESB service. This file includes the adapter configuration settings you specify with this wizard. Other configuration files (such as header files and files specific to the adapter) are also created and display in the **Applications Navigator**.

See *Oracle Application Server Adapter for Oracle Applications User's Guide* for information on using the Oracle Applications adapter listed in [Figure 1–5](#).

Oracle Application Server Adapter for Files/FTP

This chapter describes how to use the Oracle Application Server Adapter for Files/FTP (file and FTP adapters), which work in conjunction with Oracle BPEL Process Manager and Oracle Enterprise Service Bus as an external service. References to use cases for the file and FTP adapters are also provided.

This chapter contains the following sections:

- [Section 2.1, "Introduction to File and FTP Adapters"](#)
- [Section 2.2, "File and FTP Adapters Concepts"](#)
- [Section 2.3, "Using Secure FTP with the FTP Adapter"](#)
- [Section 2.4, "Using SFTP with the FTP Adapter"](#)
- [Section 2.6, "File and FTP Adapters Use Cases for Oracle BPEL Process Manager"](#)
- [Section 2.7, "File and FTP Adapters Use Case for Oracle Enterprise Service Bus"](#)

Note: The term *Oracle Application Server Adapter for Files/FTP* is used for the file and FTP adapters, which are separate adapters with very similar functionality.

2.1 Introduction to File and FTP Adapters

Oracle BPEL Process Manager and Oracle Enterprise Service Bus include the file and FTP adapters. The file and FTP adapters enable a BPEL process or an ESB service to exchange (read and write) files on local file systems and remote file systems (through use of the file transfer protocol (FTP)). The file contents can be both XML and non-XML data formats.

This section contains the following topics:

- [File and FTP Adapters Features](#)
- [File and FTP Adapters Architecture](#)
- [File and FTP Adapters Integration with Oracle BPEL Process Manager](#)
- [File and FTP Adapters Integration with Oracle Enterprise Service Bus](#)

2.1.1 File and FTP Adapters Features

The file and FTP adapters enable you to configure a BPEL process or an ESB service to interact with local and remote file system directories. This section explains the following features of the file and FTP adapters:

- [File Formats](#)
- [FTP Servers](#)
- [Inbound and Outbound Interactions](#)
- [File Debatching](#)
- [Dynamic Outbound Directory and File Name Specification](#)
- [Security](#)
- [Proxy Support](#)
- [No Payload Support](#)
- [File-Based Triggers](#)
- [High Availability](#)

2.1.1.1 File Formats

The file and FTP adapters can read and write the following file formats and use the adapter translator component at both design time and run time:

- XML (both XSD- and DTD-based)
- Delimited
- Fixed positional
- Binary data
- COBOL Copybook data

The file and FTP adapters can also treat file contents as an opaque object and pass the contents in their original format (without performing translation). The opaque option handles binary data, such as JPGs and GIFs, whose structure cannot be captured in an XSD or data you do not want to have translated.

The translator enables the file and FTP adapters to convert native data in various formats to XML. The native data can be simple (just a flat structure) or complex (with parent-child relationships). The translator can handle both XML as well as non-XML (native) format of data.

2.1.1.2 FTP Servers

The FTP adapter supports Solaris and Windows FTP servers. The FTP adapter also supports the use of FTP over SSL (FTPS) on Solaris.

2.1.1.3 Inbound and Outbound Interactions

The file and FTP adapters exchange files in the inbound and outbound directions. Based on the direction, the file and FTP adapters perform a different set of tasks.

For inbound files sent to Oracle BPEL Process Manager or Oracle Enterprise Service Bus, the file and FTP adapters perform the following operations:

1. Poll the file system looking for matches

2. Read and translate the file content based on the native schema (NXSD) defined at the design time
3. Publish the translated content as an XML message

This functionality of the file and FTP adapters is referred to as the file read operation and the component that provides this function as the file reader. This operation is known as a Java Connector Architecture (JCA) inbound interaction.

For outbound files sent from Oracle BPEL Process Manager or Oracle Enterprise Service Bus, the file and FTP adapters perform the following operations:

1. Receive messages from BPEL
2. Format the XML contents as specified at design time
3. Produce output files. The output files can be created based on following criterias: time elapsed, file size, and number of messages. You can also specify combination of these criterias for output files.

This functionality of the file and FTP adapters is referred to as the file write operation and the component that provides this functionality as the file writer. This operation is known as a JCA outbound interaction.

For the inbound and outbound directions, the file and FTP adapters use a set of configuration parameters. For example:

- The inbound file and FTP adapters have parameters for the inbound directory where the input file appears and the frequency with which to poll the directory.
- The outbound file and FTP adapters have parameters for the outbound directory in which to write the file and the file naming convention to use.

The file writer offers several conditions for output file creation. The output files can be created based on time elapsed, file size, and number of messages received.

The file reader supports polling conventions and offers several postprocessing options. You can specify to delete, move, or leave the file as it is after processing the file. The file reader can split the contents of a file and publish it in batches, instead of as a single message. This feature can be used for performance tuning of the file and FTP adapters. The file reader guarantees once and once-only delivery.

See the following sections for details about the read and write functionality of the file and FTP adapters:

- [Section 2.2.1, "File Adapter Read File Concepts"](#)
- [Section 2.2.2, "File Adapter Write File Concepts"](#)
- [Section 2.2.3, "FTP Adapter for Get File Concepts"](#)
- [Section 2.2.4, "FTP Adapter for Put File Concepts"](#)

2.1.1.4 File Debatching

When a file contains multiple messages, you can select to publish messages in a specific number of batches. This is referred to as debatching. During debatching, the file reader, upon restart, proceeds from where it left off in the previous run, thereby avoiding duplicate messages.

2.1.1.5 Dynamic Outbound Directory and File Name Specification

The file and FTP adapters enable you to dynamically specify the logical or physical name of the outbound file or outbound directory. For information about how to

specify dynamic outbound directory, refer to [Section 2.2.2.2, "Outbound File Directory Creation"](#).

2.1.1.6 Security

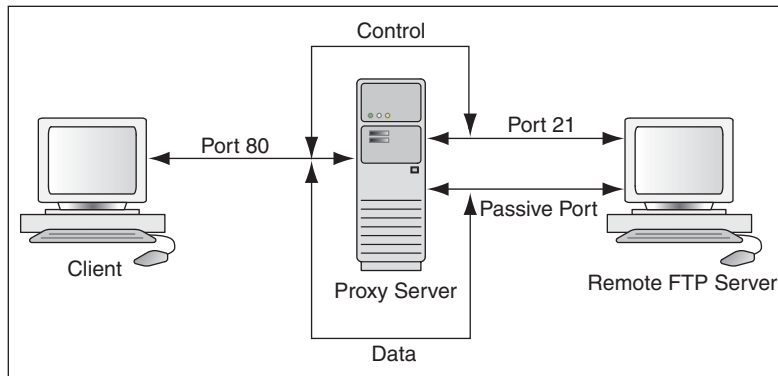
The FTP adapter supports secure FTP and SFTP to enable secure file transfer over the network.

Refer to [Section 2.3, "Using Secure FTP with the FTP Adapter"](#) and [Section 2.4, "Using SFTP with the FTP Adapter"](#) for more information.

2.1.1.7 Proxy Support

The proxy support feature of the FTP adapter can be used to transfer and retrieve data to and from the FTP servers that are located outside a firewall or can only be accessed through a proxy server. A proxy server enables the hosts in an intranet to indirectly connect to hosts on the Internet. [Figure 2–1](#) shows how a proxy server creates connections to simulate a direct connection between the client and the remote FTP server.

Figure 2–1 Remote FTP Server Communication Through a Proxy Server



To use the HTTP proxy feature, your proxy server must support FTP traffic via HTTP Connect. In addition, only passive data connections are supported with this feature. For information about how to configure the FTP adapter, refer to [Section 2.5, "Configuring the FTP Adapter for HTTP Proxy"](#).

2.1.1.8 No Payload Support

For Oracle BPEL Process Manager, the file and FTP adapters provide support for publishing only file metadata such as filename and directory to a BPEL process and exclude the payload. The BPEL process can use this metadata for subsequent processing. For example, the BPEL process can call another partner link and pass the file and directory name for further processing. So, the file and FTP adapters can be used as a notification service to notify a BPEL process whenever a new file appears in the inbound directory. To use this feature, you need to specify the value of the `UseHeaders` parameter, which is an activation spec parameter. Set the `UseHeaders` parameter as `true` in the file adapter WSDL file as shown in the following example:

```
UseHeaders="true"
```

A sample wsdl file is shown in the following example:

```
<jca:operation
    PhysicalDirectory="E:\Customer\In"
    ActivationSpec="oracle.tip.adapter.file.inbound.FileActivationSpec"
```



```

PhysicalArchiveDirectory="E:\Customer\Archive"
DeleteFile="true"
IncludeFiles=".*\.xml"
ExcludeFiles=".*\txt"
PollingFrequency="3"
MinimumAge="0"
OpaqueSchema="false" >
UseHeaders="true"
</jca:operation>

```

2.1.1.9 File-Based Triggers

The file and FTP adapters provide support for file-based triggers which can be used to control inbound adapter endpoint activation. For information about how to use file-based triggers, refer to [Section 2.2.1.4, "File Polling"](#).

2.1.1.10 High Availability

The file and FTP adapters support the high availability feature for the active-passive topology with Oracle BPEL Process Manager. Perform the following steps to configure the file and FTP adapters for this feature:

1. Create a share folder on highly available file system. This folder should have the write permission and should be accessible from all the systems that are running the file and FTP adapters.
2. Open the `pc.properties` file available in the `ORAHOME\bpel\system\service\config` directory.
3. Set `oracle.tip.adapter.file.control.dirpath` to the shared folder name or a UNC path.

For example:

```

oracle.tip.adapter.file.control.dirpath := Z:\myshare
or
oracle.tip.adapter.file.control.dirpath := \\Myserver\Mydirectory

```

4. Restart the Oracle BPEL Process Manager server.

2.1.2 File and FTP Adapters Architecture

The file and FTP adapters are based on JCA 1.5 architecture. JCA provides a standard architecture for integrating heterogeneous enterprise information systems (EIS). The adapter framework of the file and FTP adapters exposes the underlying JCA interactions as services (WSDL with JCA binding) for Oracle BPEL Process Manager integration. See *Oracle Application Server Adapter Concepts Guide* for details about OracleAS Adapter architecture.

2.1.3 File and FTP Adapters Integration with Oracle BPEL Process Manager

The file and FTP adapters are automatically integrated with Oracle BPEL Process Manager. When you create a partner link in JDeveloper BPEL Designer, the Adapter Configuration Wizard starts, as shown in [Figure 1-2](#).

This wizard enables you to select and configure the file and FTP adapters or other OracleAS Adapters, as shown in [Figure 1-3](#). The Adapter Configuration Wizard then prompts you to enter a service name, as shown in [Figure 1-4](#). When configuration is complete, a WSDL file of the same name is created in the Applications Navigator

section of Oracle JDeveloper. This WSDL file contains the configuration information you specify with the Adapter Configuration Wizard.

The Operations window of the Adapter Configuration Wizard prompts you to select an operation to perform. Based on your selection, different Adapter Configuration Wizard windows appear and prompt you for configuration information. [Table 2–1](#) lists the available operations and provides references to sections that describe the configuration information you must provide.

Table 2–1 Supported Operations for Oracle BPEL Process Manager

Operation	Section
File Adapter	-
■ Read File (inbound operation)	Section 2.2.1, "File Adapter Read File Concepts"
■ Write File (outbound operation)	Section 2.2.2, "File Adapter Write File Concepts"
FTP Adapter	-
■ Get File (inbound operation)	Section 2.2.3, "FTP Adapter for Get File Concepts"
■ Put File (outbound operation)	Section 2.2.4, "FTP Adapter for Put File Concepts"

See *Oracle Application Server Adapter Concepts Guide* for more information about OracleAS Adapter integration with Oracle BPEL Process Manager.

2.1.4 File and FTP Adapters Integration with Oracle Enterprise Service Bus

The file and FTP adapters are automatically integrated with Oracle Enterprise Service Bus. When you create a File or FTP adapter service in JDeveloper ESB Designer, the Adapter Configuration Wizard is started as shown in [Figure 1–2](#).

This wizard enables you to select and configure the file and FTP adapters. When configuration is complete, a WSDL file of the same name is created in the Applications Navigator section of Oracle JDeveloper. This WSDL file contains the configuration information you specify with the Adapter Configuration Wizard.

The Operations window of the Adapter Configuration Wizard prompts you to select an operation to perform. Based on your selection, different Adapter Configuration Wizard windows appear and prompt you for configuration information. [Table 2–2](#) lists the available operations and provides references to sections that describe the configuration information you must provide.

Table 2–2 Supported Operations for Oracle Enterprise Service Bus

Operation	Section
File Adapter	-
■ Read File (inbound operation)	Section 2.2.1, "File Adapter Read File Concepts"
■ Write File (outbound operation)	Section 2.2.2, "File Adapter Write File Concepts"
FTP Adapter	-
■ Get File (inbound operation)	Section 2.2.3, "FTP Adapter for Get File Concepts"
■ Put File (outbound operation)	Section 2.2.4, "FTP Adapter for Put File Concepts"

See *Oracle Application Server Adapter Concepts Guide* for more information about OracleAS Adapter integration with Oracle Enterprise Service Bus.

2.2 File and FTP Adapters Concepts

This section contains the following topics:

- [File Adapter Read File Concepts](#)
- [File Adapter Write File Concepts](#)
- [FTP Adapter for Get File Concepts](#)
- [FTP Adapter for Put File Concepts](#)

2.2.1 File Adapter Read File Concepts

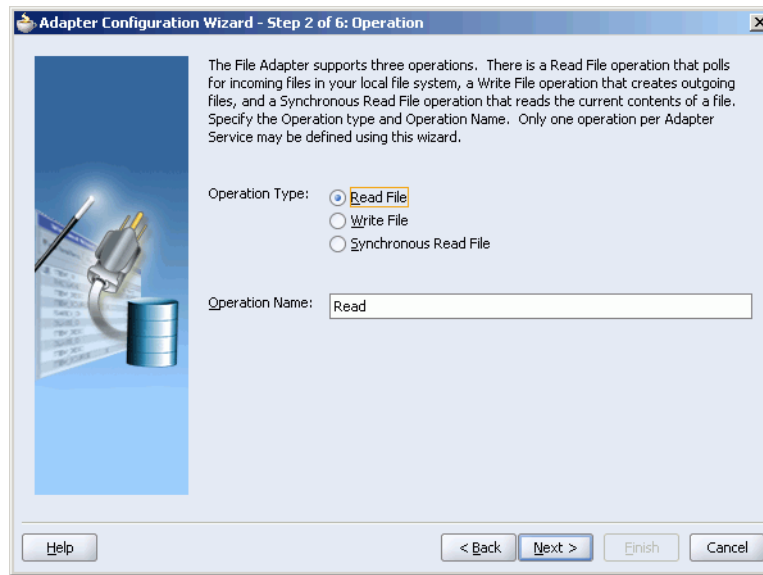
In the inbound direction, the file adapter polls and reads files from a file system for processing. This section provides an overview of the inbound file reading capabilities of the file adapter. You use the Adapter Configuration Wizard to configure the file adapter for use with a BPEL process or an ESB service. This creates an inbound WSDL file named after the service name you specify with the Adapter Configuration Wizard. An inbound header file named `fileAdapterInboundheader.wsdl` is also created.

This section contains the following topics:

- [Inbound Operation](#)
- [Inbound File Directory Specifications](#)
- [File Matching and Batch Processing](#)
- [File Polling](#)
- [File Processing](#)
- [Postprocessing](#)
- [Native Data Translation](#)
- [Error Handling](#)
- [Guaranteed Delivery and Recovery from Server Failures](#)
- [Inbound Service Name WSDL File](#)
- [Inbound Header WSDL File](#)
- [Synchronous File Reading Capabilities](#)

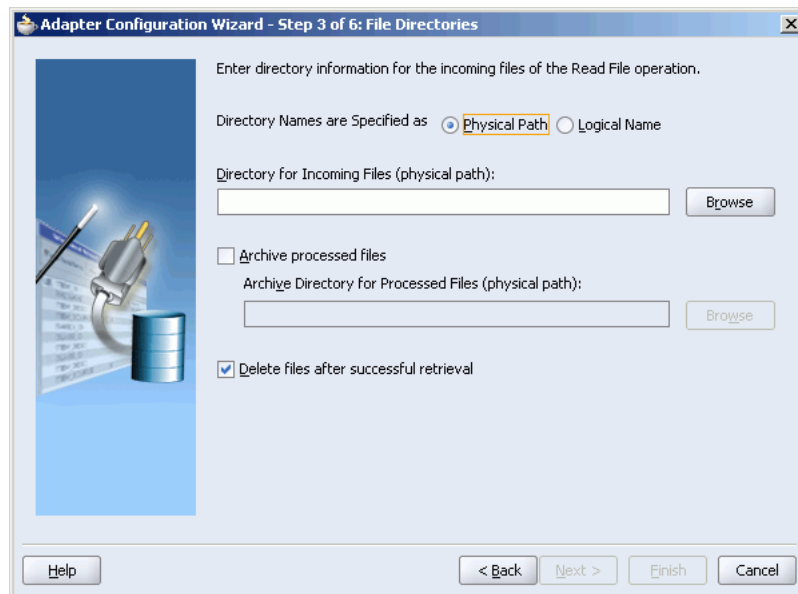
2.2.1.1 Inbound Operation

For inbound operations with the file adapter, you select **Read File** operation as shown in [Figure 2-2](#).

Figure 2–2 Selecting the Read File Operation

2.2.1.2 Inbound File Directory Specifications

The File Directories window of the Adapter Configuration Wizard shown in [Figure 2–3](#) enables you to specify information about the directory to use for reading inbound files and the directories in which to place successfully processed files.

Figure 2–3 Adapter Configuration Wizard-Specifying Incoming Files

The following sections describe the file directory information to specify:

- [Specifying Inbound Physical or Logical Directory Paths in Oracle BPEL Process Manager](#)
- [Specifying Inbound Physical or Logical Directory Paths in Oracle Enterprise Service Bus](#)
- [Archiving Successfully Processed Files](#)

■ Deleting Files After Retrieval

Specifying Inbound Physical or Logical Directory Paths in Oracle BPEL Process Manager

You can specify inbound directory names as physical or logical paths in Oracle BPEL Process Manager and Oracle Enterprise Service Bus. Physical paths are values such as `c:\inputDir`.

In Oracle BPEL Process Manager, logical properties are specified in the inbound WSDL file and their logical-physical mapping is resolved using partner link properties. You specify the logical parameters once at design time, and you can later modify the physical directory name as needed.

For example, the generated inbound WSDL file looks as follows for the logical input directory name `InputFileDir`.

```
<operation name="Read">
  <jca:operation
    LogicalDirectory="InputFileDir"
    ActivationSpec="oracle.tip.adapter.file.inbound.FileActivationSpec"
    IncludeFiles=".*"
    PollingFrequency="5"
    MinimumAge="0"
    DeleteFile="true"
    OpaqueSchema="true" >
  </jca:operation>
```

In the BPEL partner link of the `bpel.xml` file, you then provide the physical parameter values (in this case, the directory path) of the corresponding logical `ActivationSpec` or `InteractionSpec`. This resolves the mapping between the logical directory name and actual physical directory name.

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<BPELSuitcase>
  <BPELProcess id="ComplexStructure" src="ComplexStructure.bpel">
    <partnerLinkBindings>
      <partnerLinkBinding name="InboundPL">
        <property name="wsdlLocation">ComplexStructureInbound.wsdl</property>
      </partnerLinkBinding>
      <partnerLinkBinding name="OutboundPL">
        <property name="wsdlLocation">ComplexStructureOutbound.wsdl</property>
      </partnerLinkBinding>
    </partnerLinkBindings>
    <activationAgents>
      <activationAgentclassName=
"oracle.tip.adapter.fw.agent.jca.JCAActivationAgent"partnerLink="InboundPL">
        <property name="InputFileDir">C:/ora_home/integration/bpm/samples/tutorials/
121.FileAdapter/ComplexStructure/InputDir/</property>
        <property name="portType">Read_ptt</property>
      </activationAgent>
    </activationAgents>
  </BPELProcess>
</BPELSuitcase>
```

Note: Multiple file adapters polling one inbound directory are not supported. Ensure that each is polling its own unique directory.

Editing the Existing Logical Name Removes the Mapping in the bpel.xml File The File Directories window of the Adapter Configuration Wizard enables you to specify logical names instead of physical directory paths for both the incoming files directory and archive directory when using the FTP or File adapter. You must also provide corresponding property settings in the `bpel.xml` file (either manually or through the Property tab of the partner link activity).

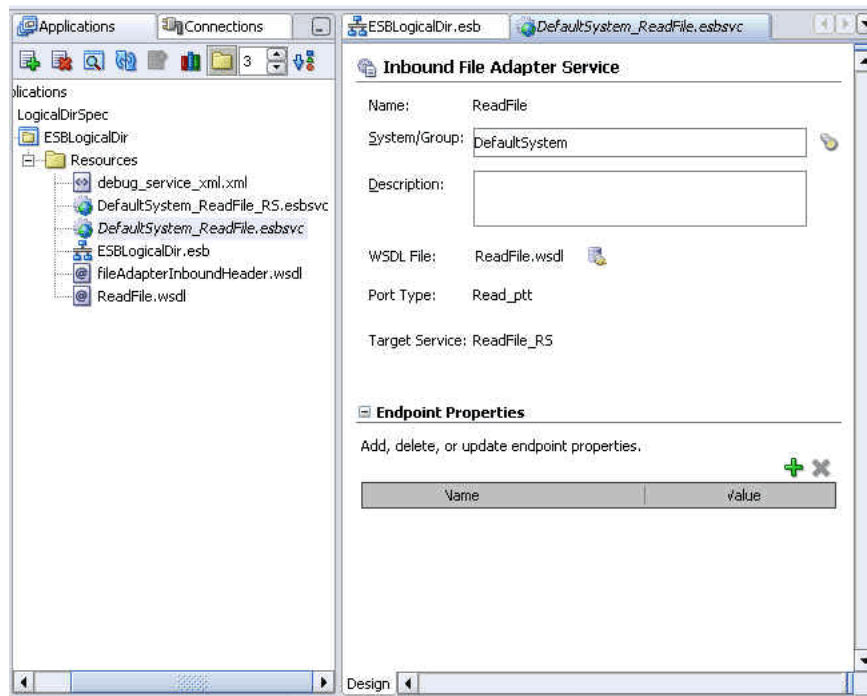
If you later rerun the Adapter Configuration Wizard and change the logical name of either the physical directory path or the archive directory, both properties are removed from the `bpel.xml` file. You must update both these properties in the `bpel.xml` file.

Specifying Inbound Physical or Logical Directory Paths in Oracle Enterprise Service Bus

You can specify inbound directory names as physical or logical paths in Oracle Enterprise Service Bus. Physical paths are values such as `c:\inputDir`.

For logical names, you can specify the logical names at design time in the File Directories window shown in [Figure 2-3](#) and then provide logical-physical mapping by using the Endpoint properties. For example, `ReadFile` is an inbound adapter service. You have specified `InputFileDir` as the logical directory name at design time. After completing the Adapter Configuration wizard, when you click the `ReadFile_esbsvc` file, the inbound file adapter window similar to the one shown in [Figure 2-4](#) is displayed.

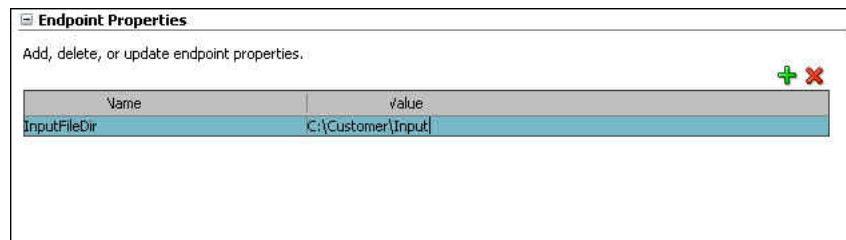
Figure 2-4 Sample Inbound File Adapter Window



When you click the plus sign (+) in Endpoint Properties section, the Endpoint Property Chooser dialog box shown in [Figure 2-5](#) is displayed.

Figure 2–5 Endpoint Property Chooser Dialog Box

Select the **InputFileDir** property and click **OK**. Now you can specify the physical directory name in the **Value** field as shown in [Figure 2–6](#).

Figure 2–6 Endpoint Properties Section

Archiving Successfully Processed Files

This option enables you to specify a directory in which to place successfully processed files. You can also specify the archive directory as a logical name. In this case, you must follow the logical-to-physical mappings described in ["Specifying Inbound Physical or Logical Directory Paths in Oracle BPEL Process Manager"](#) and ["Specifying Inbound Physical or Logical Directory Paths in Oracle Enterprise Service Bus"](#).

Note: Files that are 7MB or larger in size cannot be delivered. As an alternative, you can debatch large files (if they have multiple messages), and publish these files in messages of size less than 7 MB. This alternative is applicable only to structured files (comma-delimited or fixed position), which contain more than one message. It is not applicable to binary or XML files.

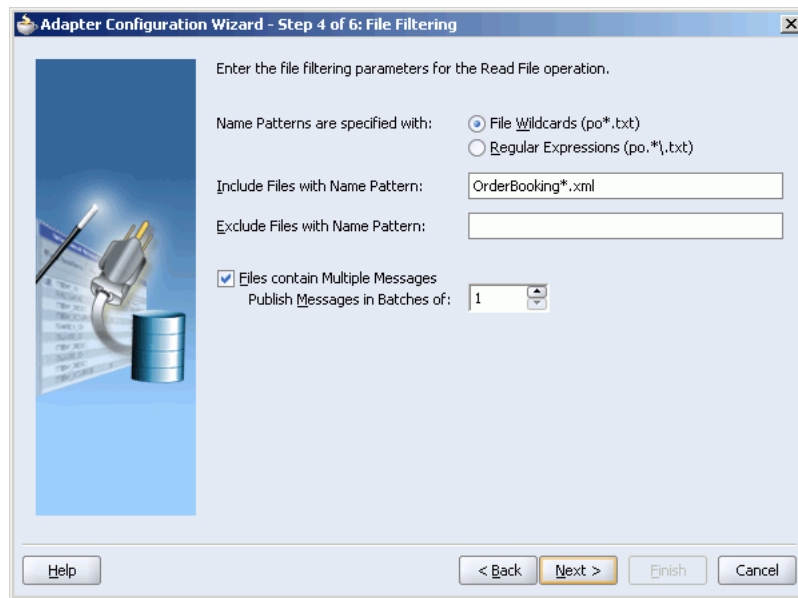
Deleting Files After Retrieval

This option enables you to specify whether or not to delete files after a successful retrieval. If this check box is not selected, processed files remain in the inbound directory, but are ignored. Only files with modification dates more recent than the last processed file are retrieved. If you place another file in the inbound directory with the same name as a file that has already been processed, but the modification date remains the same, then that file is not retrieved.

2.2.1.3 File Matching and Batch Processing

The File Filtering window of the Adapter Configuration Wizard shown in [Figure 2–7](#) enables you to specify details about the files to retrieve or ignore.

The file adapter acts as a file listener in the inbound direction. The file adapter polls the specified directory on a local or remote file system and looks for files that match a specified naming criteria.

Figure 2–7 Adapter Configuration Wizard-File Filtering

The following sections describe the file filtering information to specify:

- [Specifying a Naming Pattern](#)
- [Including and Excluding Files](#)
- [Debatching Multiple Inbound Messages](#)

Specifying a Naming Pattern

Specify the naming convention that the file adapter uses to poll for inbound files. You can also specify the naming convention for files you do not want to process. Two naming conventions are available for selection. The file adapter matches the files that appear in the inbound directory.

- File wildcards (po* .txt)

Retrieves all files that start with po and end with .txt. This convention conforms to Windows operating system standards.
- Regular expressions (po . * \ .txt)

Retrieves all files that start with po and end with .txt. This convention conforms to Java Development Kit (JDK) regular expression (regex) constructs.

Notes:

- If you later select a different naming pattern, ensure that you also change the naming conventions you specify in the Include Files and Exclude Files fields. The Adapter Configuration Wizard does not automatically make this change for you.
- Do *not* specify *.* as the convention for retrieving files.
- Be aware of any file length restrictions imposed by your operating system. For example, Windows operating system file names cannot be more than 256 characters in length (the filename, plus the complete directory path). Some operating systems also have restrictions on the use of specific characters in file names. For example, Windows operating systems do not allow characters such as \, /, :, *, <, >, or |.

Including and Excluding Files

If you use regular expressions, the values you specify in the Include Files and Exclude Files fields must conform to JDK regular expression (regex) constructs. For both fields, different regex patterns must be provided separately. The Include Files and Exclude Files fields correspond to the `IncludeFiles` and `ExcludeFiles` parameters, respectively, of the inbound WSDL file.

Note: The regex pattern complies with the JDK regex pattern. According to the JDK regex pattern, the correct connotation for a pattern of any characters with any number of occurrences is a period followed by a plus sign (`.+`). An asterisk (`*`) in a JDK regex is not a placeholder for a string of any characters with any number of occurrences.

If you want the inbound file adapter to pick up all file names that start with `po` and which have the extension `.txt`, you must specify the Include Files field as `po.*\.``txt` when the name pattern is a regular expression. In this regex pattern example:

- A period (`.`) indicates any character
- An asterisk (`*`) indicates any number of occurrences
- A backslash followed by a period (`\.`) indicates the character period (`.`), as indicated with the backslash escape character

The Exclude Files field is constructed similarly.

If you have Include Files field and Exclude Files field expressions that have an overlap, then the exclude files expression takes precedence. For example, if Include Files is set to `abc*.``txt` and Exclude Files is set to `abcd*.``txt`, then you receive any files prefixed with `abcd*.`

[Table 2–3](#) lists details of Java regex constructs.

Note: Do not begin JDK regex pattern names with the following characters: `+`, `?`, or `*`.

Table 2–3 Java Regular Expression Constructs

Matches	Construct
Characters	-
The character <i>x</i>	<i>x</i>
The backslash character	\\
The character with octal value 0 <i>n</i> (0 <= <i>n</i> <= 7)	\\0 <i>n</i>
The character with octal value 0 <i>nn</i> (0 <= <i>n</i> <= 7)	\\0 <i>nn</i>
The character with octal value 0 <i>mnn</i> (0 <= <i>m</i> <= 3, 0 <= <i>n</i> <= 7)	\\0 <i>mnn</i>
The character with hexadecimal value 0 <i>xhh</i>	\\x <i>hh</i>
The character with hexadecimal value 0 <i>xhhhh</i>	\\u <i>hhhh</i>
The tab character ('\\u0009')	\\t
The newline (line feed) character ('\\u000A')	\\n
The carriage-return character ('\\u000D')	\\r
The form-feed character ('\\u000C')	\\f
The alert (bell) character ('\\u0007')	\\a
The escape character ('\\u001B')	\\e
The control character corresponding to <i>x</i>	\\c <i>x</i>
-	-
Character classes	-
<i>a</i> , <i>b</i> , or <i>c</i> (simple class)	[<i>abc</i>]
Any character except <i>a</i> , <i>b</i> , or <i>c</i> (negation)	[^ <i>abc</i>]
<i>a</i> through <i>z</i> or <i>A</i> through <i>Z</i> , inclusive (range)	[<i>a-zA-Z</i>]
<i>a</i> through <i>d</i> , or <i>m</i> through <i>p</i> : [<i>a-dm-p</i>] (union)	[<i>a-d[m-p]</i>]
<i>d</i> , <i>e</i> , or <i>f</i> (intersection)	[<i>a-z&&[def]</i>]
<i>a</i> through <i>z</i> , except for <i>b</i> and <i>c</i> : [<i>ad-z</i>] (subtraction)	[<i>a-z&&[^bc]</i>]
<i>a</i> through <i>z</i> , and not <i>m</i> through <i>p</i> : [<i>a-lq-z</i>](subtraction)	[<i>a-z&&[^m-p]</i>]
-	-
Predefined character classes	-
Any character (may or may not match line terminators)	-
A digit: [0–9]	\\d
A nondigit: [^0–9]	\\D
A whitespace character: [\\t\\n\\x0B\\f\\r]	\\s
A nonwhitespace character: [^\\s]	\\S
A word character: [<i>a-zA-Z_0–9</i>]	\\w
A nonword character: [^\\w]	\\W
Greedy quantifiers	-
<i>X</i> , once or not at all	<i>X</i> ?
<i>X</i> , zero or more times	<i>X</i> *

Table 2–3 (Cont.) Java Regular Expression Constructs

Matches	Construct
X, one or more times	X+
X, exactly <i>n</i> times	X{n}
X, at least <i>n</i> times	X{n, }
X, at least <i>n</i> , but not more than <i>m</i> times	X{n, m}

For details about Java regex constructs, go to

<http://java.sun.com/j2se/1.4.2/docs/api>

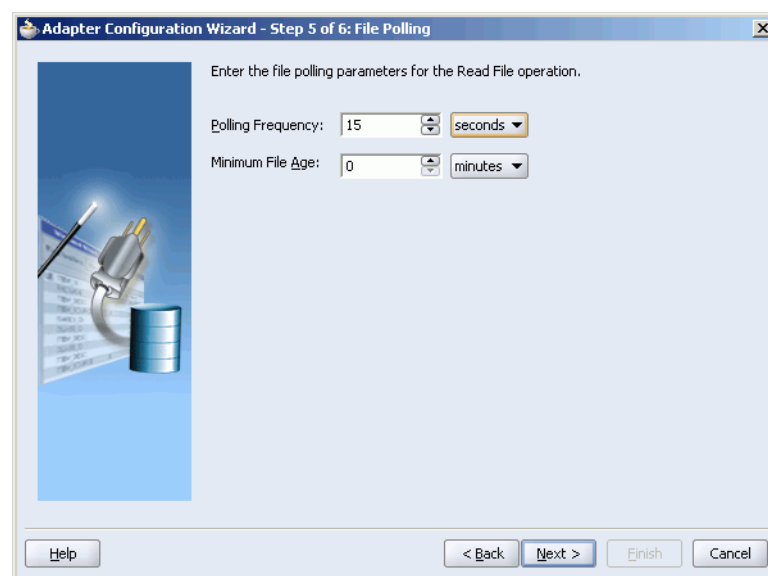
Debatching Multiple Inbound Messages

You can select if incoming files have more than one message, and specify the number of messages in one batch file to publish. When a file contains multiple messages and this check box is selected, this is referred to as debatching. Nondebatching is when the file contains only a single message and the check box is not selected. Debatching is not supported if the input file is in XML format.

2.2.1.4 File Polling

The File Polling window of the Adapter Configuration Wizard shown in [Figure 2–8](#) enables you to specify the following inbound polling parameters:

- The frequency with which to poll the inbound directory for new files to retrieve.
- The minimum file age of files to retrieve. For example, this enables a large file to be completely copied into the directory before it is retrieved for processing. The age is determined by the last modified time stamp. For example, if you know that it takes three to four minutes for a file to be written, then set the minimum age to five minutes. If a file is detected in the input directory and its modification time is less than five minutes older than the current time, then the file is not retrieved because it is still potentially being written to.

Figure 2–8 Adapter Configuration Wizard-File Polling

Using Trigger Files

By default, polling by inbound file and FTP adapters start as soon as the endpoint is activated. However, if you want more control over polling, then you can use a file-based trigger. Once the file and the FTP adapter finds the specified trigger file in local or remote directory, it starts polling for the files in the inbound directory.

For example, a BPEL process is writing files to a `InOut` directory. A second BPEL process is polling the same directory for files. If you want the second process to start polling the directory only after the first process has written all the files, then you can use a trigger file. You can configure the first process to create a trigger file at the end. The second process will start polling the inbound directory once it finds the trigger file.

The trigger file directory can be the same as the inbound polling directory or different from the inbound polling directory. However, if your trigger file directory and the inbound polling directory are the same, then you should ensure that the name of the trigger file is not similar to the file filter specified in the Adapter Configuration window shown in [Figure 2-7](#).

The content of a trigger file is never read and therefore should not be used as payload for an inbound receive activity.

[Table 2-4](#) lists the parameters that you need to specify in the inbound service WSDL file:

Table 2-4 Trigger File Parameters

Parameter	Description	Example
TriggerFilePhysicalDirectory or TriggerFileLogicalDirectory	The physical or logical name of the directory which the file and FTP adapters will look for the trigger file. The TriggerFilePhysicalDirectory and TriggerFileLogicalDirectory parameters are optional. These parameters should be used only if the trigger file directory is different from the inbound polling directory. By default, the file and FTP adapters looks for the trigger file in the inbound polling directory. The TriggerFileLogicalDirectory parameter is not supported in Oracle Enterprise Service Bus.	TriggerFilePhysicalDirectory="C:\foo" TriggerFileLogicalDirectory="TriggerFileDir"
TriggerFile	The name of the trigger file.	TriggerFile="Purchase order.trg"

A sample inbound service WSDL is shown in the following example:

```
<jca:operation
    PhysicalDirectory="C:\ORAHOME\integration\jdev\jdev\mywork\triggers_
    proj\inputDir"
```

```

    ActivationSpec="oracle.tip.adapter.file.inbound.FileActivationSpec"
    IncludeFiles="Order.*\*.xml"
    PollingFrequency="30"
    MinimumAge="0"
    DeleteFile="false"
    OpaqueSchema="true"
    TriggerFilePhysicalDirectory="C:\foo"
    TriggerFile="Purchaseorder.trg"
</jca:operation>

```

Note: The file and FTP adapters look for the trigger file in the trigger file directory each time the endpoint gets activated.

2.2.1.5 File Processing

The file adapter prepares the files for processing and delivers them to the adapter translator for translation and debatching (if necessary).

If you have many inbound files to process or very large files of more than 1 MB, you may need to increase the `config timeout` value in the `transaction-manager.xml` file as shown in the following example:

```
<transaction-timeout="30000"/>
```

The location of the `transaction-manager.xml` file depends on your installation type. [Table 2–5](#) lists the installation type and the corresponding `transaction-manager.xml` file location.

Table 2–5 Transaction-manager.xml File Location

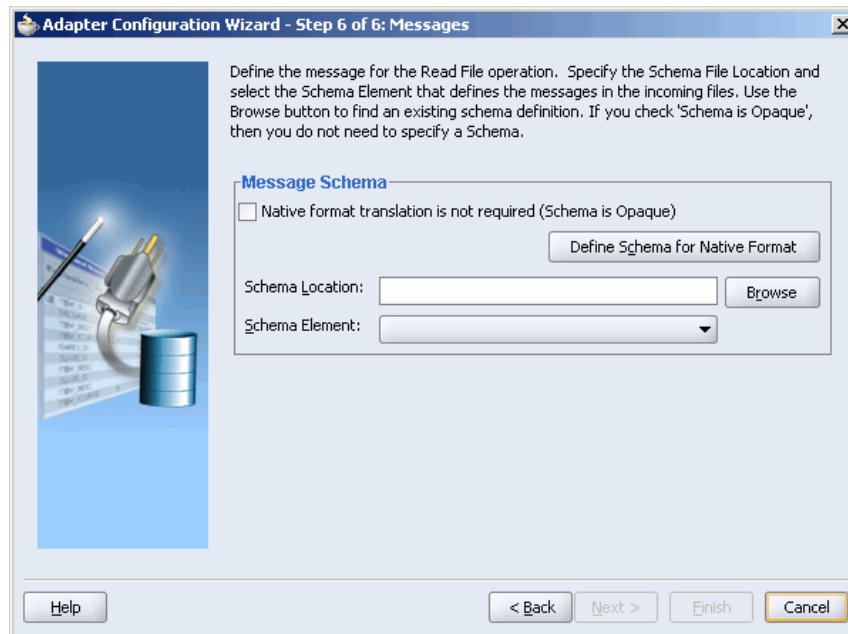
Installation Type	File Location
SOA Basic Installation	<i>Oracle_Home\j2ee\home\config</i>
Oracle Enterprise Service Bus on Oracle Application Server Middle tier	<i>Oracle_Home\j2ee\homemid\config</i>
Oracle BPEL Process Manager on Oracle Application Server Middle tier	<i>Oracle_Home\j2ee\homemid\config</i>

2.2.1.6 Postprocessing

The file adapter supports several postprocessing options. After processing the file, files are deleted if specified in the File Polling window shown in [Figure 2–8](#). Files can also be moved to a completion (archive) directory if specified in the File Directories window shown in [Figure 2–3](#).

2.2.1.7 Native Data Translation

The next Adapter Configuration Wizard window that appears is the Messages window shown in [Figure 2–9](#). This window enables you to select the XSD schema file for translation.

Figure 2–9 Specifying the Schema

If native format translation is not required (for example, a JPG or GIF image is being processed), then select the **Native format translation is not required** check box. The file is passed through in base-64 encoding.

XSD files are required for translation. If you want to define a new schema or convert an existing data type definition (DTD) or COBOL Copybook, then select **Define Schema for Native Format**. This starts the Native Format Builder Wizard. This wizard guides you through the creation of a native schema file from file formats such as comma-separated value (CSV), fixed-length, DTD, and COBOL Copybook. After the native schema file is created, the Messages window is displayed with the Schema File URL and Schema Element fields filled in. See [Section 7.1, "Creating Native Schema Files with the Native Format Builder Wizard"](#) for more information.

Note: Ensure that the schema you specify includes a namespace. If your schema does not have a namespace, then an error message is displayed.

2.2.1.8 Error Handling

The file adapter provides several inbound error handling capabilities:

- [rejectedMessageHandlers Property](#)
- [fatalErrorFailoverProcess Property](#)
- [uniqueMessageSeparator Property](#)
- [Default Error Directory](#)

rejectedMessageHandlers Property

This property is specific to Oracle BPEL Process Manager. You can configure your BPEL process to process the correct records of a file and write only the rejected records to an archive directory by setting the `rejectedMessageHandlers` property. For example, assume that you have a file with four records. If three records are processed

successfully and one record is not, then the file is processed with the three correct records. The errored record is written to a rejected messages directory. If a file has only message, the entire message is rejected.

You first define the `rejectedMessageHandlers` property as an `activationAgent` property in the `bpel.xml` file so that it applies to inbound WSDL operations only:

```
<BPELSuitcase>
  <BPELProcess src="ErrorTest.bpel" id="ErrorTest">
    <activationAgents>
      <activationAgent
        className="oracle.tip.adapter.fw.agent.jca.JCAActivationAgent"
        partnerLink="inboundPL">
      <property name="rejectedMessageHandlers">
        file://C:/orabpel/samples/test/errorTest/rejectedMessages
      </property>
    </activationAgents>
  </BPELProcess>
</BPELSuitcase>
```

This causes errored messages to be written to the configured directory using the following naming pattern:

`INVALID_MSG_ + process-name + operation-name + current-time`

The rejection handlers can be other BPEL processes, WSIF handlers, or Oracle Advanced Queue handlers.

fatalErrorFailoverProcess Property

This property is specific to Oracle BPEL Process Manager and not for Oracle Enterprise Service Bus. If the file adapter (or any OracleAS Adapter) encounters an unrecoverable system error (such as no more memory or a full disk), it can instruct the adapter framework to shut down the BPEL process.

You can optionally configure a standby (or failover) BPEL process to be invoked when the adapter starts the shutdown request of the (main) BPEL process.

You configure this failover BPEL process by setting the `fatalErrorFailoverProcess` property as an `activationAgent` property in the `bpel.xml` file.

```
<property name="fatalErrorFailoverProcess">
  bpel://bpel-domain:password|process-name|operation-name|
  input-message-part-name
</property>
```

where *password* (which can be omitted if it is `bpel`) can be encrypted.

For example:

```
<property name="fatalErrorFailoverProcess">
  bpel://default|JCA-FatalErrorHandler|handleError|message
</property>
```

or

```
<property name="fatalErrorFailoverProcess">
  bpel://default:C23487CFA591952D3ED0B81F0961F65A|JCA-FatalErrorHandler|handleError|
  message</property>
```

where the `bpel` password was specified in encrypted form (using the `encrypt.bat` (for Windows) or `encrypt.sh` (for UNIX) command line utility).

The fatal error BPEL process must use a specific input (WSDL) message type. This message type is defined in the system-provided WSDL file `FatalErrorMessageWSDL.wsdl`. This WSDL can be referenced (imported) using the following:

```
<import namespace="http://xmlns.oracle.com/pcbpel/fatalErrorHandler"
location="http://localhost:9700/orabpel/xmllib/jca/FatalErrorMessage.wsdl"/>
```

The XML schema type for this message is as follows:

```
<complexType name="FatalErrorMessageType">
  <sequence>
    <element name="Reason" type="string"/>
    <element name="Exception" type="string"/>
    <element name="StackTrace" type="string"/>
  </sequence>
</complexType>
```

The purpose of the failover BPEL process can be to undertake error compensating actions, alert someone through e-mail or short message service (SMS), or restart some other process.

uniqueMessageSeparator Property

In the case of debatching (multiple messages in a single file), messages from the first bad message to the end of the file are rejected. If each message has a unique separator and that separator is not part of any data, then rejection can be more fine-grained. In these cases, you can define a `uniqueMessageSeparator` in the schema element of the native schema to have the value of this unique message separator. This property controls how the adapter translator works when parsing through multiple records in one file (debatching). This property enables recovery even when detecting bad messages inside a large batch file; when a bad record is detected, the adapter translator skips to the next unique message separator boundary and continues from there. If you do not set this property, then all records that follow the errored record are also rejected.

The following schema file provides an example of using `uniqueMessageSeparator`.

```
<?xml version="1.0" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://TargetNamespace.com/Reader"
  xmlns:tns="http://TargetNamespace.com/Reader"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  nxsd:encoding="US-ASCII" nxsd:stream="chars"
  nxsd:version="NXSD" nxsd:uniqueMessageSeparator="${eol}">
  <xsd:element name="emp-listing">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="emp" minOccurs="1" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="GUID" type="xsd:string" nxsd:style="terminated"
                nxsd:terminatedBy=", " nxsd:quotedBy="&quot;">
            </xsd:element>
              <xsd:element name="Designation" type="xsd:string"
                nxsd:style="terminated" nxsd:terminatedBy=", "
                nxsd:quotedBy="&quot;">
            </xsd:element>
              <xsd:element name="Car" type="xsd:string" nxsd:style="terminated"
                nxsd:terminatedBy=", " nxsd:quotedBy="&quot;">
            </xsd:element>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```



```

        <xsd:element name="Laptop" type="xsd:string"
            nxsd:style="terminated" nxsd:terminatedBy=", "
            nxsd:quotedBy="&quot;">
    </xsd:element>
    <xsd:element name="Location" type="xsd:string"
        nxsd:style="terminated" nxsd:terminatedBy=", "
        nxsd:quotedBy="&quot;">
    </xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
<!--NXSDWIZ:D:\work\jDevProjects\Temp_BPEL_process\Sample2\note.txt:-->
<!--USE-HEADER:false:-->

```

Default Error Directory

If you do not set the `rejectedMessageHandlers` property in Oracle BPEL Process Manager, then errored records during translation are placed by default in the following directory:

Oracle_Home\bpel\domains\domain_name\jca\project_directory\rejectedMessages

If you do not set the `rejectedMessageHandlers` property in Oracle Enterprise Service Bus, errored records during translation are placed by default in the following directory:

Oracle_Home\j2EE\jca\event_name\rejectedMessages

2.2.1.9 Guaranteed Delivery and Recovery from Server Failures

For Oracle BPEL Process Manager, the file and FTP adapters guarantee once-only delivery of inbound files. This includes guaranteed delivery of large files through FTP. If your system has to shut down for some reason, then the read functionality of the file adapter upon restart avoids creating duplicate messages.

In case of debatching, the file and FTP adapters internally stores the line, column, or byte position of the files being processed and in case of a BPEL server failure, the file and FTP adapters start reading from where it had left off. If the FTP server fails, the inbound adapter goes into an error-recovery mode and keeps checking if the server is up periodically.

For FTP adapter, in case of non-debatching, the file being processed gets deleted only after it has been retrieved from the FTP server and published to Oracle BPEL Process Manager. In case of failure, the file is either rejected or it does not get deleted. If the file gets rejected, it goes to the default or user-specified rejected directory as mentioned in the [rejectedMessageHandlers Property](#) and [Default Error Directory](#) sections. However, if the file is not deleted, it is picked up again when the server restarts.

If your system server shuts down while inbound messages are being processed, you must manually perform recovery when the server restarts to ensure that all message records are recovered. For example, a file has ten messages and the server shuts down after three messages have been processed. This causes the fourth message to go undelivered. When the server restarts and begins processing with message five (the offset of the last successfully rejected message), you must manually recover message four to ensure that all messages are preserved.

Perform the following procedures to recover the rejected message record.

1. Log in to Oracle BPEL Control.
2. Click the **BPEL Processes** tab.
3. Click **Perform Manual Recovery** under the **Related Tasks** section.
4. Click **Recover**.

2.2.1.10 Inbound Service Name WSDL File

When you finish configuring the file adapter, a WSDL file is generated for the inbound direction. The file is named after the service name you specified on the Service Name window of the Adapter Configuration Wizard shown in [Figure 1-4](#). You can rerun the wizard at any time to change your operation definitions.

The `ActivationSpec` parameter holds the inbound configuration information. The `ActivationSpec` and a set of inbound file adapter properties are part of the inbound WSDL file.

The inbound WSDL contains the following information:

```
<pc:inbound_binding />
  <operation name="Read">
    <jca:operation
      PhysicalDirectory="C:/ora_
home/integration/bpm/samples/tutorials/121.FileAdapter/ComplexStructure/inputDir/"
      ActivationSpec="oracle.tip.adapter.file.inbound.FileActivationSpec"
      PhysicalArchiveDirectory="C:/ora_
home/integration/bpm/samples/tutorials/121.FileAdapter/ComplexStructure/archiveDir/"
      DeleteFile="true"
      IncludeFiles=".+\\.txt"
      PollingFrequency="5"
      MinimumAge="0"
      OpaqueSchema="false" >
    </jca:operation>
  </operation>
</binding>
```

The `ActivationSpec` parameters are specified in the Adapter Configuration Wizard during design time and appear in the binding element of the inbound WSDL. The inbound file adapter uses the following configuration parameters:

- **PollingFrequency**
This parameter specifies how often to poll a given input directory for new files. The parameter is of type `int` and is mandatory. The default value is 1.
- **PhysicalDirectory**
This parameter specifies the physical input directory to be polled. The parameter is of type `String`. The inbound directory where the files appear is mandatory. You must specify the physical directory or logical directory.
- **LogicalDirectory**
This parameter specifies the logical input directory to be polled. The parameter is of type `String`.

- PublishSize

This parameter indicates whether the file contains multiple messages, and how many messages to publish to the BPEL process at a time. The parameter is of type `int` and is not mandatory. The default value is 1.

Note: Do not enter a negative value for the number of batches in which to publish messages in the adapter service WSDL file. For example, `PublishSize="-1"`. This causes a validation error when you attempt to deploy your process from Oracle JDeveloper. Note that the Adapter Configuration Wizard does not allow you to enter negative values for this parameter.

- PhysicalArchiveDirectory

This parameter specifies where to archive successfully processed files. The parameter is of type `String` and is not mandatory.

- LogicalArchiveDirectory

This parameter specifies the logical directory in which to archive successfully processed files. The parameter is of type `String` and is not mandatory.

- IncludeFiles

This parameter specifies the pattern for types of files to pick up during polling. The parameter is of type `String` and is not mandatory.

- ExcludeFiles

This parameter specifies the pattern for types of files to be excluded during polling. The parameter is of type `String` and is not mandatory.

2.2.1.11 Inbound Header WSDL File

The WSDL file shown in [Section 2.2.1.10, "Inbound Service Name WSDL File"](#) includes two attributes that indicate which message and part define the operation headers:

```
<jca:header message="hdr:InboundHeader_msg" part="inboundHeader" />
```

The `fileAdapterInboundHeader.wsdl` file defines these attributes. This file also provides information such as the name of the file being processed and its directory path. This file is created along with the service name WSDL file and is displayed in the Applications section of Oracle JDeveloper.

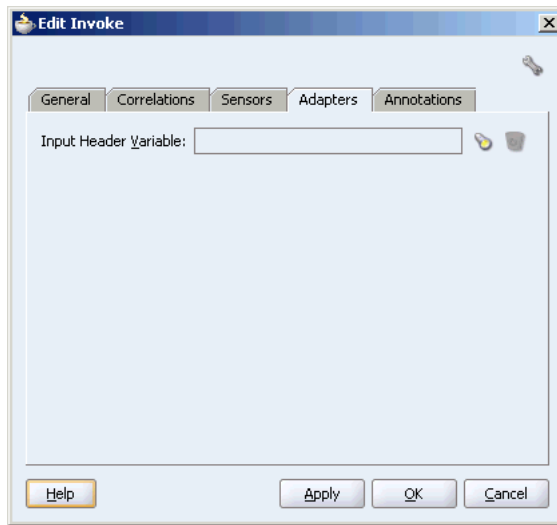
```
<definitions
  name="fileAdapter"
  targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/file/"
  xmlns:tns="http://xmlns.oracle.com/pcbpel/adapter/file/"
  xmlns="http://schemas.xmlsoap.org/wsdl/" >
  <types>
    <schema attributeFormDefault="qualified" elementFormDefault="qualified"
      targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/file/"
      xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:FILEAPP="http://xmlns.oracle.com/pcbpel/adapter/file/">
      <element name="InboundFileType">
        <complexType>
          <sequence>
            <element name="fileName" type="string"/>
            <element name="directory" type="string"/>
          </sequence>
        </complexType>
      </element>
    </types>
  </definitions>
```

```
        </complexType>
      </element>
    </schema>
  </types>

  <!-- Header Message -->
  <message name="InboundHeader_msg">
    <part element="tns:InboundFileHeaderType" name="inboundHeader"/>
  </message>
</definitions>
```

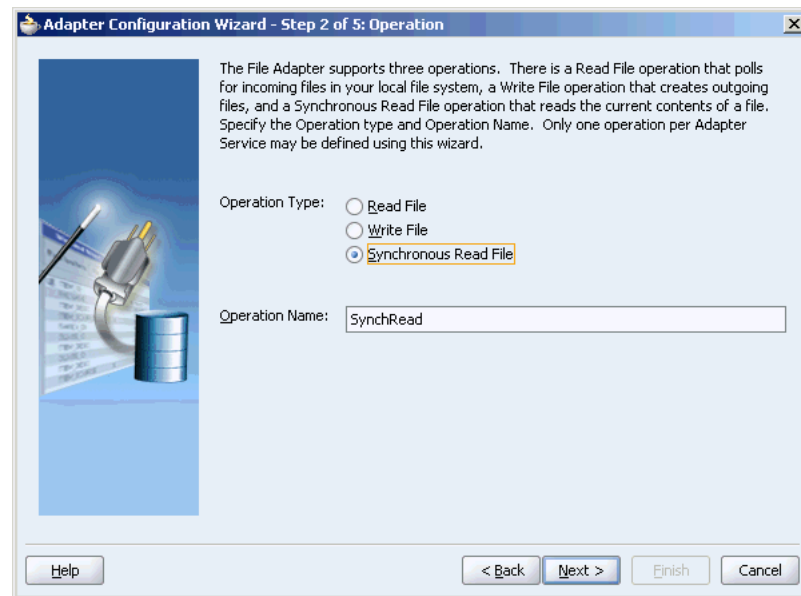
Refer to the online help of an invoke activity in Oracle JDeveloper for more information about header variables. Click **Help** as shown in the [Figure 2–10](#).

Figure 2–10 *Edit Invoke Dialog Box*

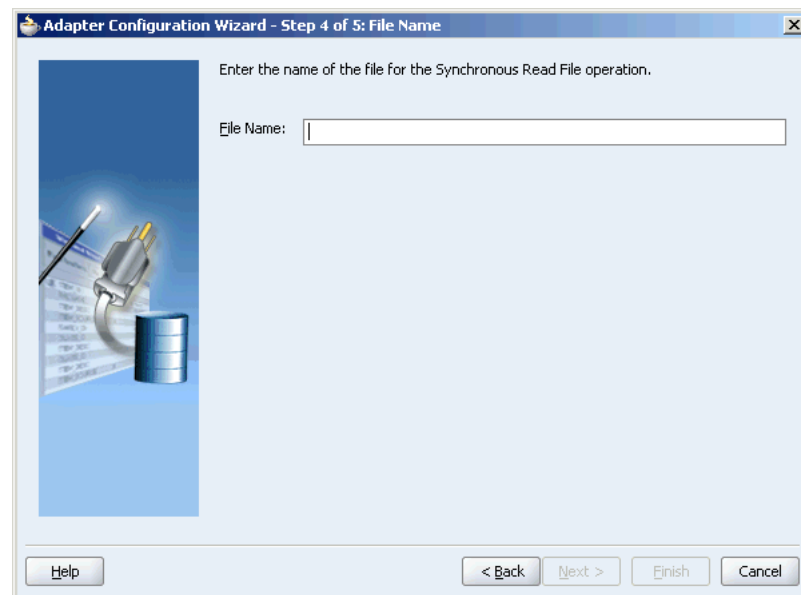


2.2.1.12 Synchronous File Reading Capabilities

The file adapter can also synchronously read a single file using an invoke activity. For reading a file synchronously, you select Synchronous Read File operation as shown in [Figure 2–2](#).

Figure 2–11 Synchronous Read Operation Window

All the windows of the Adapter Configuration Wizard are similar to the Read File operation except the File Name window shown in [Figure 2–12](#). You can specify the name of the file to be read in the **File Name** field.

Figure 2–12 File Name Window

2.2.2 File Adapter Write File Concepts

In the outbound direction, the file adapter receives messages from the BPEL process or ESB service and writes the messages to a file in a file system. This section provides an overview of the outbound file writing capabilities of the file adapter. You use the Adapter Configuration Wizard to configure the file adapter for use with a BPEL process or an ESB Service. This creates an outbound WSDL file named after the service

name you specify with the Adapter Configuration Wizard. An outbound header file named `fileAdapterOutboundheader.wsdl` is also created.

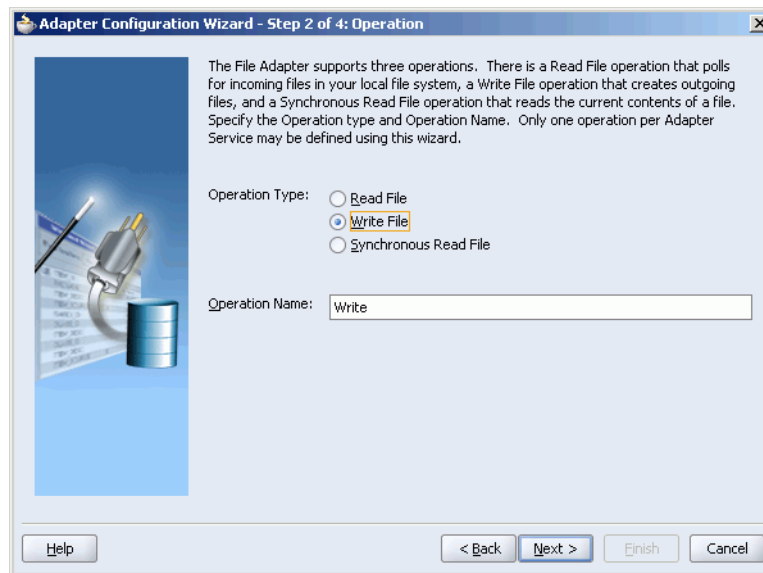
This section contains the following topics:

- [Outbound Operation](#)
- [Outbound File Directory Creation](#)
- [Native Data Translation](#)
- [Outbound Error Handling](#)
- [Outbound Service Name WSDL File](#)
- [Outbound Header WSDL File](#)

2.2.2.1 Outbound Operation

For outbound operations with the file adapter, you select **Write File** operation. [Figure 2–13](#) shows this selection.

Figure 2–13 *Selecting the Write File Operation*



2.2.2.2 Outbound File Directory Creation

For the outbound operation, you can specify the outbound directory, outbound file naming convention to use, and, if necessary, the batch file conventions to use.

The File Configuration Window of the Adapter Configuration Wizard shown in [Figure 2–14](#) enables you to specify the directory for outgoing files and the outbound file naming convention.

Figure 2–14 Adapter Configuration Wizard-Parameters for Outgoing Files

Adapter Configuration Wizard - Step 3 of 4: File Configuration

Specify the parameters for the Write File operation.

Directory specified as ☒ Physical Path ☐ Logical Name

Directory for Outgoing Files (physical path):

File Naming Convention (po_%SEQ%.txt):

Write to new file when existing file meets any of these conditions

☒ Number of Messages Equals:

☒ Elapsed Time Exceeds:

☒ File Size Exceeds:

The following sections describe the file configuration information to specify:

- [Specifying Outbound Physical or Logic Directory Paths in Oracle BPEL Process Manager](#)
- [Specifying the Outbound File Naming Convention](#)
- [Specifying a Dynamic Outbound File Name](#)
- [Batching Multiple Outbound Messages](#)
- [Purging Files](#)

Specifying Outbound Physical or Logic Directory Paths in Oracle BPEL Process Manager

You can specify outbound directory names as physical or logical paths. Physical paths are values such as `c:\outputDir`.

If you specify logical parameters, then the generated WSDL file looks as follows for the logical outbound directory name `OutputFileDir`.

```
<jca:binding />
  <operation name="Write">
    <jca:operation
      InteractionSpec="oracle.tip.adapter.file.outbound.FileInteractionSpec"
      LogicalDirectory="OutputFileDir"
      FileNamingConvention="po_%SEQ%.xml">
    </jca:operation>
  </operation>
  ....
```

In the BPEL partner link in the `bpel.xml` file, you then specify an outbound partner link binding property by using the Property tab of the partner link. This resolves the mapping between the logical directory name and the actual physical directory name.

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<BPELSuitecase>
  <BPELProcess id="ComplexStructure" src="ComplexStructure.bpel">
    <partnerLinkBindings>
```

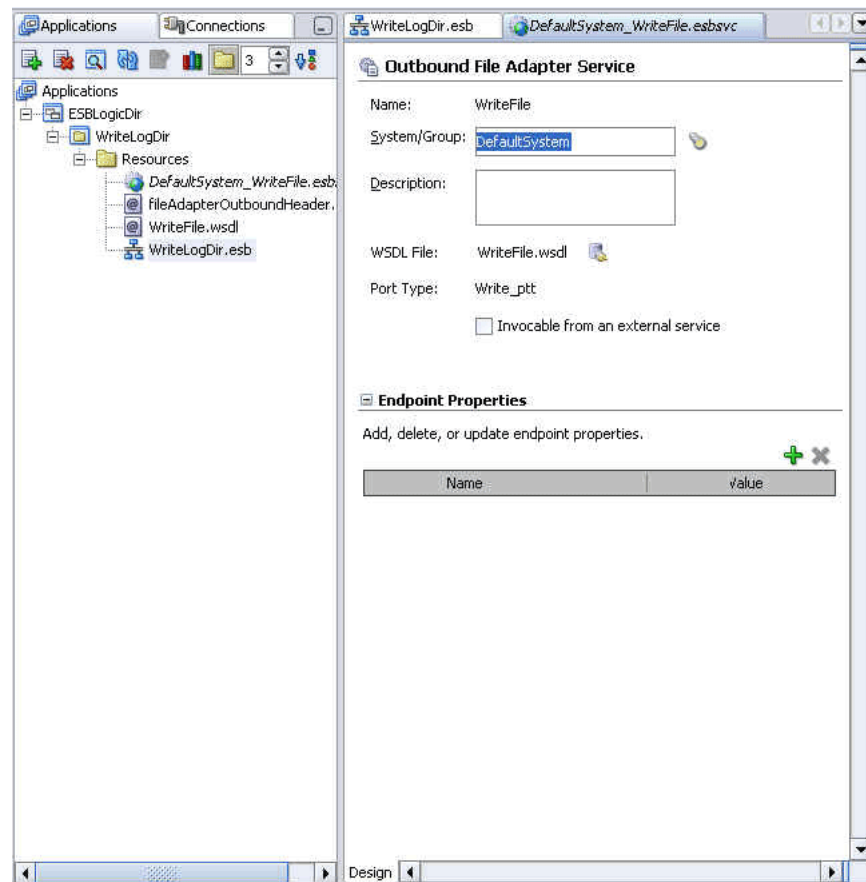
```
<partnerLinkBinding name="InboundPL">
  <property name="wsdlLocation">ComplexStructureInbound.wsdl</property>
</partnerLinkBinding>
<partnerLinkBinding name="OutboundPL">
  <property name="wsdlLocation">ComplexStructureOutbound.wsdl</property>
  <property name="OutputFileDir">C:/ora_
home/integration/bpm/samples/tutorials/
121.FileAdapter/ComplexStructure/outputDir/</property>
</partnerLinkBinding>
</partnerLinkBindings>
<activationAgents>
  <activationAgentclassName=
"oracle.tip.adapter.fw.agent.jca.JCAActivationAgent"partnerLink="InboundPL">
  <property name="portType">Read_ptt</property>
  </activationAgent>
</activationAgents>
</BPELProcess>
</BPELSuitcase>
```

Note: Ensure that you limit the length of outbound file names (the file name, plus the complete directory path) to 200 characters. This is not an exact limit, but rather a recommendation. When an outbound file name is long (for example, 215 characters), a blank file with that name is created in the outbound directory.

Specifying Outbound Physical or Logical Directory Paths in Oracle Enterprise Service Bus

You can specify outbound directory names as physical or logical paths in Oracle Enterprise Service Bus. Physical paths are values such as `c:\inputDir`.

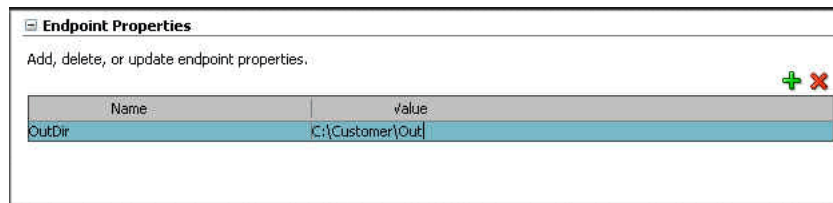
You can specify the logical names at design time in the File Directories window shown in [Figure 2-3](#) and then provide logical-physical mapping by using the Endpoint properties. For example, `WriteFile` is an outbound adapter service. You have specified `OutDir` as the logical directory name at design time. After completing the Adapter Configuration Wizard, when you click the `WriteFile_esbsvc` file, an outbound file adapter window similar to the one shown in [Figure 2-4](#) is displayed.

Figure 2–15 Sample Outbound File Adapter Window

When you click the plus sign (+) in Endpoint Properties section, the Endpoint Property Chooser dialog box shown in [Figure 2–5](#) is displayed.

Figure 2–16 Endpoint Property Chooser Dialog Box

Select the **OutDir** property and click **OK**. Now, you can specify the physical directory name in the **Value** field as shown in [Figure 2–6](#).

Figure 2–17 Endpoint Properties Section

Specifying Dynamic Outbound Directory Name

For outbound operation, you can specify a dynamic outbound directory name. You can use variables to specify dynamic outbound directory names through use of the `directory` parameter in the `fileAdapterOutboundHeader.wsdl` as shown in the following example.

```
<definitions
  name="fileAdapter"
  targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/file/"
  xmlns:tns="http://xmlns.oracle.com/pcbpel/adapter/file/"
  xmlns="http://schemas.xmlsoap.org/wsdl/" >
  <types>
    <schema attributeFormDefault="qualified" elementFormDefault="qualified"
      targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/file/"
      xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:FILEAPP="http://xmlns.oracle.com/pcbpel/adapter/file/">
      <element name="OutboundFileHeaderType">
        <complexType>
          <sequence>
            <element name="fileName" type="string"/>
            <element name="directory" type="string"/>
          </sequence>
        </complexType>
      </element>
    </schema>
  </types>
  <!-- Header Message -->
  <message name="OutboundHeader_msg">
    <part element="tns:OutboundFileHeaderType" name="outboundHeader"/>
  </message>
</definitions>
```

You need to add the `directory` parameter in the `fileAdapterOutboundHeader.wsdl` file. The `fileAdapterOutboundHeader.wsdl` file is read only and cannot be modified in Oracle JDeveloper. So, you need to open the file and modify it separately.

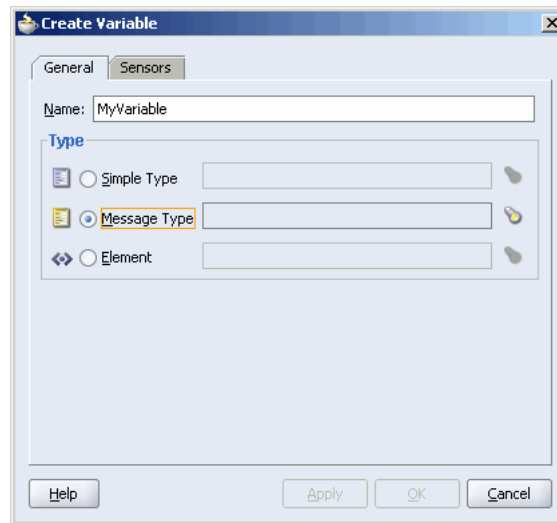
After modifying the `fileAdapterOutboundHeader.wsdl` file, create a variable of message type `OutboundHeader_msg` and assign it a value. Perform the following steps for this:

1. Double-click the invoke activity.
2. Click the **Adapters** tab.
3. Click the Browse Variables icon.
4. In the Variable Chooser dialog box, click the **Create an Object** icon.



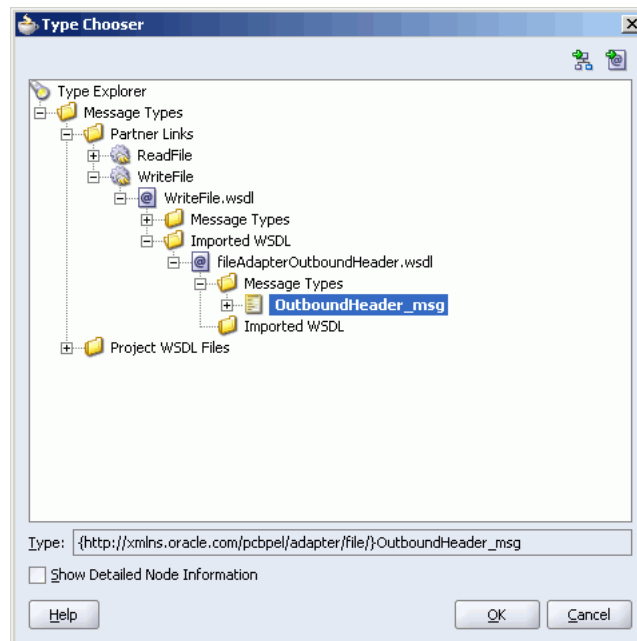
5. In the Create Variable dialog box shown in [Figure 2–18](#), select **Message Type** and click the **Browse Message Type** icon.

Figure 2–18 Create Variable dialog box

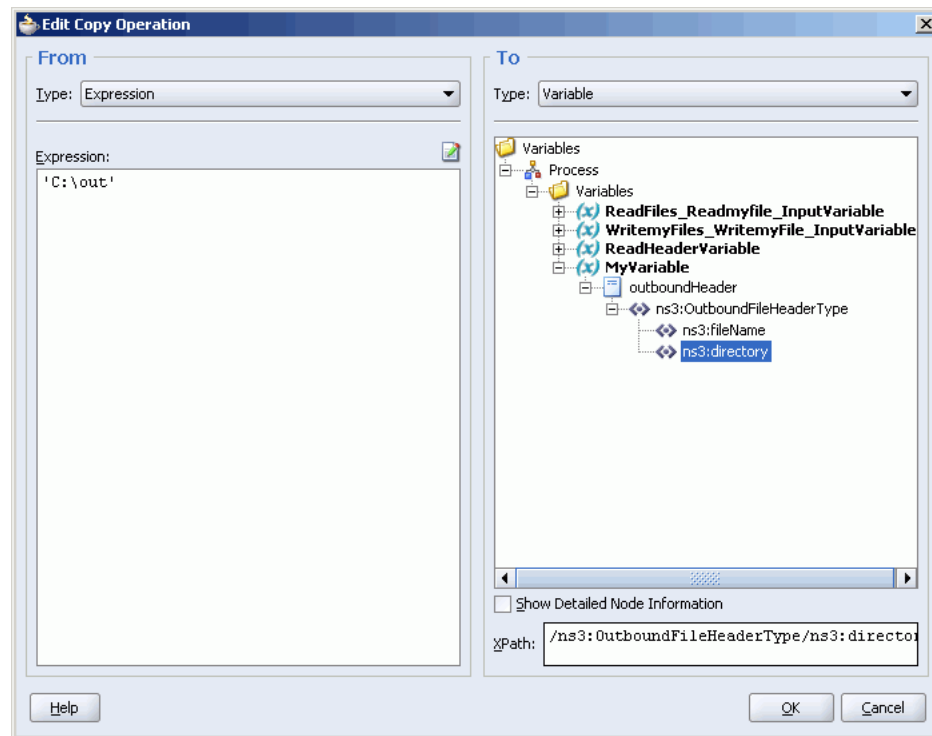


6. Select **OutboundHeader_msg** as shown in [Figure 2–19](#) and click **OK** till you exit the invoke activity dialog box.

Figure 2–19 Type Chooser Dialog Box



7. Double-click the assign activity and click the **Copy Operation** tab.
8. Click **Create** and then **Copy Operation**.
9. In the Create Copy Operation dialog box, you can either select a variable or an expression. For expression, select **Expression** from Type and specify the directory name and path as shown in [Figure 2–20](#). The output file will be written to this directory.

Figure 2–20 Create Copy Operation Dialog Box

10. Click **OK** till you exit the assign activity dialog box.

Note: When using dynamic directories, ensure that parameters such as `NumberMessages`, `ElapsedTime`, and `FileSize` are not defined in the outbound adapter service wsdl file. These parameters are not supported with dynamic directories.

Specifying the Outbound File Naming Convention

Specify the naming convention to use for outgoing files. You cannot enter completely static names such as `po.txt`. This is to ensure the uniqueness of outgoing file names, which prevents files from being inadvertently overwritten. Instead, outgoing file names must be a combination of static and dynamic portions.

The prefix and suffix portions of the file example shown in [Figure 2–14](#) are static (for example, `po_` and `.xml`). The `%SEQ%` variable of the name is dynamic and can be a sequence number or a time stamp (for example, `po_%yyMMddHHmmss%.xml` to create a file with a time stamp).

The sequence number is written to a file if the system shuts down unexpectedly. If you choose a name starting with `po_`, followed by a sequence number and the extension `txt` as the naming convention of the outgoing files, then you must specify `po_%SEQ%.txt`.

If you choose a name starting with `po_`, followed by a time stamp with the pattern `yyyy.MM.dd` and the extension `txt` as the naming convention of the outgoing file, then you must specify `po_%yyyy.MM.dd%.txt`. For example, the outgoing file name can be `po_2004.11.29.txt`.

You cannot use a regular expression for outbound synchronous reads. In these cases, the exact file name must be known.

A time stamp is specified by date and time pattern strings. Within date and time pattern strings, unquoted letters from 'A' to 'Z' and from 'a' to 'z' are interpreted as pattern letters representing the components of a date or time string. Text can be quoted using single quotes (') to avoid interpretation. The characters "''" represent a single quote. All other characters are not interpreted.

The Java pattern letters are defined in [Table 2–6](#).

Table 2–6 Java Pattern Letters

Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
Y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day in week	Text	Tuesday; Tue
a	AM/PM marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in AM/PM (0-11)	Number	0
h	Hour in AM/PM (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
z	Time zone	General Time Zone	Pacific Standard Time; PST; GMT-08:00
Z	Time zone	RFC 822 Time Zone	-0800

Different presentations in the pattern are as follows:

- Text

For formatting, if the number of pattern letters is four or more, then the full form is used; otherwise, a short or abbreviated form is used if available. For parsing, both forms are accepted, independent of the number of pattern letters.

- Number

For formatting, the number of pattern letters is the minimum number of digits, and shorter numbers are zero-padded to this amount. For parsing, the number of pattern letters is ignored unless it is needed to separate two adjacent fields.

- Year

For formatting, if the number of pattern letters is two, then the year is truncated to two digits; otherwise, it is interpreted as a number.

For parsing, if the number of pattern letters is more than two, then the year is interpreted literally, regardless of the number of digits. Using the pattern `MM/dd/yyyy`, `01/11/12` parses to Jan 11, 12 A.D.

For parsing with the abbreviated year pattern (`y` or `yy`), the abbreviated year is interpreted relative to some century. The date is adjusted to be within 80 years before and 20 years after the time instance is created. For example, using a pattern of `MM/dd/yy` and Jan 1, 1997 is created; the string `01/11/12` is interpreted as Jan 11, 2012, while the string `05/04/64` is interpreted as May 4, 1964. During parsing, only strings consisting of exactly two digits are parsed into the default century. Any other numeric string, such as a one-digit string, a three-or-more-digit string, or a two-digit string that is not all digits (for example, `-1`), is interpreted literally. So `01/02/3` or `01/02/003` is parsed, using the same pattern, as Jan 2, 3 AD. Likewise, `01/02/-3` is parsed as Jan 2, 4 BC.

- **Month**

If the number of pattern letters is 3 or more, then the month is interpreted as text; otherwise, it is interpreted as a number.

- **General time zone**

Time zones are interpreted as text if they have names. For time zones representing a GMT offset value, the following syntax is used:

```
GMTOffsetTimeZone:
    GMT Sign Hours : Minutes
Sign: one of
    + -
Hours:
    Digit
    Digit Digit
Minutes:
    Digit Digit
Digit: one of
    0 1 2 3 4 5 6 7 8 9
```

Hours must be between 0 and 23, and Minutes must be between 00 and 59. The format is locale-independent and digits must be taken from the Basic Latin block of the Unicode standard.

For parsing, RFC 822 time zones are also accepted.

For formatting, the RFC 822 4-digit time zone format is used:

```
RFC822TimeZone:
    Sign TwoDigitHours Minutes
TwoDigitHours:
    Digit Digit
```

TwoDigitHours must be between 00 and 23. Other definitions are the same as for general time zones.

For parsing, general time zones are also accepted.

Specifying a Dynamic Outbound File Name

You can also use variables to specify dynamic outbound file names through use of the `OutboundHeader_msg` message name in the `fileAdapterOutboundHeader.wsdl`.

```

<definitions
  name="fileAdapter"
  targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/file/"
  xmlns:tns="http://xmlns.oracle.com/pcbpel/adapter/file/"
  xmlns="http://schemas.xmlsoap.org/wsdl/" >
  <types>
    <schema attributeFormDefault="qualified" elementFormDefault="qualified"
      targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/file/"
      xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:FILEAPP="http://xmlns.oracle.com/pcbpel/adapter/file/">
      <element name="OutboundFileHeaderType">
        <complexType>
          <sequence>
            <element name="fileName" type="string"/>
          </sequence>
        </complexType>
      </element>
    </schema>
  </types>
  <!-- Header Message -->
  <message name="OutboundHeader_msg">
    <part element="tns:OutboundFileHeaderType" name="outboundHeader"/>
  </message>
</definitions>

```

Create a variable of message type `OutboundHeader_msg`, assign it a value, and use it in the Adapter tab of an invoke activity. For example:

```

<variables>
  <variable name="fileHeader" messageType="file:OutboundHeader_msg"/>
[...]
```

```

<assign>
  <copy>
    <from>*testfile.txt*</from>
    <to variable="fileHeader" part="outboundHeader"
      query="/file:OutboundFileHeaderType/file:fileName"/>
  </copy>
</assign>

<invoke name="FileSend" partnerLink="outboundPL"
  portType="out:FileWrite_PortType" operation="Write"
  inputVariable="payload"
  bpelx:inputHeaderVariable="fileHeader"/>

```

See the online Help that is included with the **Adapter** tab in JDeveloper BPEL Designer for more information.

Batching Multiple Outbound Messages

In the simplest scenario, you specify writing a single file to a single message. You can also specify the outbound method for batch file writing. This enables you to specify the number of messages in one batch file to publish. The following batch file settings are provided in the File Configuration window shown in [Figure 2-14](#):

- Number of messages equals
Specify a value that, when equaled, causes a new outgoing file to be created.
- Elapsed time exceeds
Specify a time that, when equaled, causes a new outgoing file to be created.

- File size exceeds

Specify a file size that, when equaled, causes a new outgoing file to be created. For example, assume that you specify a value of three for the number of messages received and a value of 1 MB for the file size. When you receive two messages that when combined equal or exceed 1 MB, or three messages that are less than 1 MB, an output file is created.

If the file adapter encounters some problem during batching, then it starts batching at the point at which it left off on recovery.

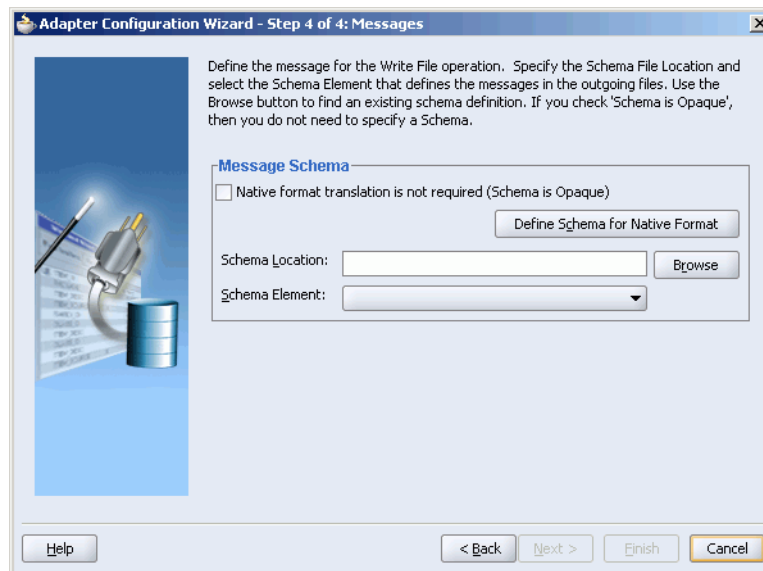
Purging Files

You cannot purge control files specific to a BPEL process. For example, you cannot start anew with sequence number 1 for outbound files or reprocess a (debatching) file with errors from the beginning instead of from the last published message.

2.2.2.3 Native Data Translation

The next Adapter Configuration Wizard window that appears is the Messages window shown in [Figure 2–21](#). This window enables you to select the XSD schema file for translation.

Figure 2–21 Specifying the Schema



As with specifying the schema for the inbound direction, you can perform the following tasks in this window:

- Specify if native format translation is not required
- Select the XSD schema file for translation
- Start the Native Format Builder Wizard to create an XSD file from file formats such as CSV, fixed-length, DTD, and COBOL Copybook

See [Section 2.2.1.7, "Native Data Translation"](#) for more information about the Messages window.

2.2.2.4 Outbound Error Handling

As with inbound files, the file adapter guarantees once-only delivery of outbound files. This includes guaranteed delivery of large files through FTP. If your system shuts down unexpectedly, then the write functionality of the file adapter on restart has the information to proceed from where it left off in the previous run, thereby avoiding duplicate messages. If the target host is unavailable, the file adapter supports retrying to send documents. For example, if the directory to which it is trying to write is read-only, then the file adapter tries to write again.

For Oracle BPEL Process Manager, you must configure two partner link retry properties from the **Property** tab of the partner link (updates the `bpel.xml` file):

```
<partnerLinkBinding name="WriteFile">
  <property name="wsdlLocation">WriteFile.wsdl</property>
  <property name="retryMaxCount">10</property>
  <property name="retryInterval">60</property>
```

The write operation eventually succeeds (if the problem is resolved in a timely fashion). If the problem is not resolved within the retry period, then a binding fault is sent back to the BPEL process.

For Oracle Enterprise Service Bus, you can configure the following endpoint properties:

- OutboundRetryCount
- OutboundRetryInterval
- OutboundRetryEnabled

These properties can be configured through the `esb_config.ini` file, through ESB Console, or through the endpoint properties section of an outbound adapter service as shown in [Figure 2–6](#). Following are the default values as specified in the `esb_config.ini` file:

```
OutboundRetryCount = 3
OutboundRetryInterval = 5
OutboundRetryEnabled = true
```

2.2.2.5 Outbound Service Name WSDL File

When you complete configuration of the file adapter with the Adapter Configuration Wizard, a WSDL file is generated for the outbound direction. The file is named after the service name you specified on the Service Name window of the Adapter Configuration Wizard shown in [Figure 1–4](#). You can rerun the wizard at any time to change your operation definitions.

The `InteractionSpec` parameters in the WSDL file contain the outbound configuration information that you specified with the Adapter Configuration Wizard during design time. The `InteractionSpec` parameters and a set of outbound file adapter properties are part of the outbound WSDL file. The outbound WSDL includes the following information:

```
<jca:binding />
  <operation name="Write">
    <jca:operation
      PhysicalDirectory="E:\Customer\out"
      InteractionSpec="oracle.tip.adapter.file.outbound.FileInteractionSpec"
      FileNamingConvention="po_%SEQ%.xml"
      NumberMessages="1"
      ElapsedTime="60"
```

```
        FileSize="1000000"  
        OpaqueSchema="false" >  
    </jca:operation>  
    <input>  
        <jca:header message="hdr:OutboundHeader_msg" part="outboundHeader"/>  
    </input>  
    </operation>  
</binding>
```

The outbound file adapter uses the following configuration parameters:

- **PhysicalDirectory**
This parameter specifies the physical directory in which to write output files. The parameter is of type `String`. The outbound directory where the outgoing files are written is mandatory. You must specify the physical directory or logical directory.
- **LogicalDirectory**
This parameter specifies the logical directory in which to write output files. The parameter is of type `String`.
- **NumberMessages**
This parameter is used for outbound batching. The outgoing file is created when the number of messages condition is met. The parameter is of type `String` and is not mandatory. The default value is 1.
- **ElapsedTime**
This parameter is used for outbound batching. When the time specified elapses, the outgoing file is created. The parameter is of type `String` and is not mandatory. The default value is 1.
- **FileSize**
This parameter is used for outbound batching. The outgoing file is created when the file size condition is met. The parameter is of type `String` and is not mandatory. The default value is 1000 KB.
- **FileNamingConvention**
This parameter is for the naming convention for the outbound write operation file. The parameter is of type `String` and is mandatory.

2.2.2.6 Outbound Header WSDL File

The WSDL file shown in [Section 2.2.2.5, "Outbound Service Name WSDL File"](#) includes two attributes that indicate which message and part define the operation headers:

```
<jca:header message="hdr:OutboundHeader_msg" part="outboundHeader" />
```

The `fileAdapterOutboundHeader.wsdl` file defines these attributes, as well as information about the outbound file name. This file is created along with the service name WSDL file, and is displayed in the **Applications** navigator of Oracle JDeveloper.

```
<definitions  
    name="fileAdapter"  
    targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/file/"  
    xmlns:tns="http://xmlns.oracle.com/pcbpel/adapter/file/"  
    xmlns="http://schemas.xmlsoap.org/wsdl/" >  
    <types>  
        <schema attributeFormDefault="qualified" elementFormDefault="qualified"
```

```

        targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/file/"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:FILEAPP="http://xmlns.oracle.com/pcbpel/adapter/file/"
        <element name="OutboundFileHeaderType">
            <complexType>
                <sequence>
                    <element name="fileName" type="string"/>
                </sequence>
            </complexType>
        </element>
    </schema>
</types>

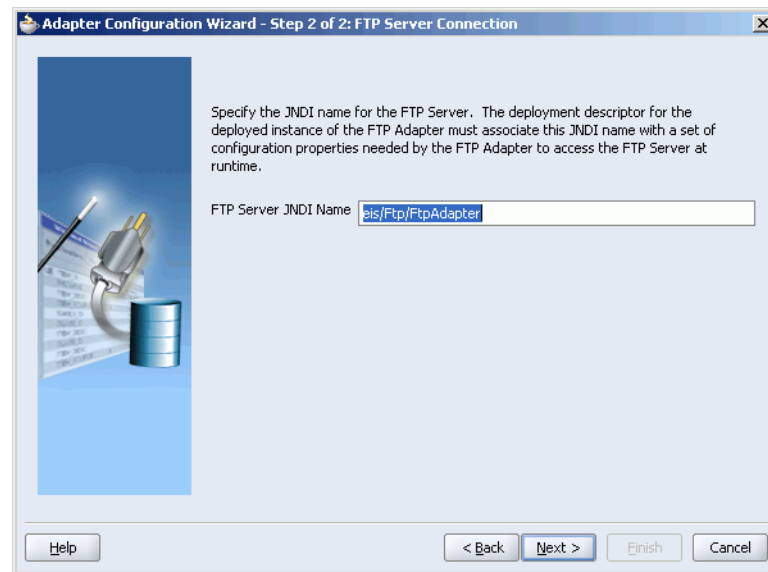
<!-- Header Message -->
<message name="OutboundHeader_msg">
    <part element="tns:OutboundFileHeaderType" name="outboundHeader"/>
</message>
</definitions>

```

2.2.3 FTP Adapter for Get File Concepts

In the inbound direction, the FTP adapter works the same way as the Read File operations of the file adapter in that it polls and gets files from a file system for processing. The major difference is that the FTP adapter is used for remote file exchanges. Because of this, the Adapter Configuration Wizard asks for connection information to an FTP server to be used later, as shown in [Figure 2-22](#).

Figure 2-22 Specifying FTP Server Connection Information



To create the FTP server connection that you specify in [Figure 2-22](#), you must edit the `oc4j-ra.xml` deployment descriptor file for adapter instance JNDI name and FTP server connection information. The location of the `oc4j-ra.xml` file depends on the installation type. [Table 2-7](#) lists the installation type and corresponding `oc4j-ra.xml` file location.

Table 2–7 *oc4j-ra.xml File Location*

Installation Type	File Location
SOA Basic Installation	<i>Oracle_</i> <i>Home\j2ee\home\application-deployments\default\</i> <i>FtpAdapter</i>
Oracle Enterprise Service Bus on Oracle Application Server Middle tier	<i>Oracle_</i> <i>Home\j2ee\homemid\application-deployments\defau</i> <i>lt\FtpAdapter</i>
Oracle BPEL Process Manager on Oracle Application Server Middle tier	<i>Oracle_</i> <i>Home\j2ee\homemid\application-deployments\defau</i> <i>lt\FtpAdapter</i>

A sample of `oc4j-ra.xml` is as follows:

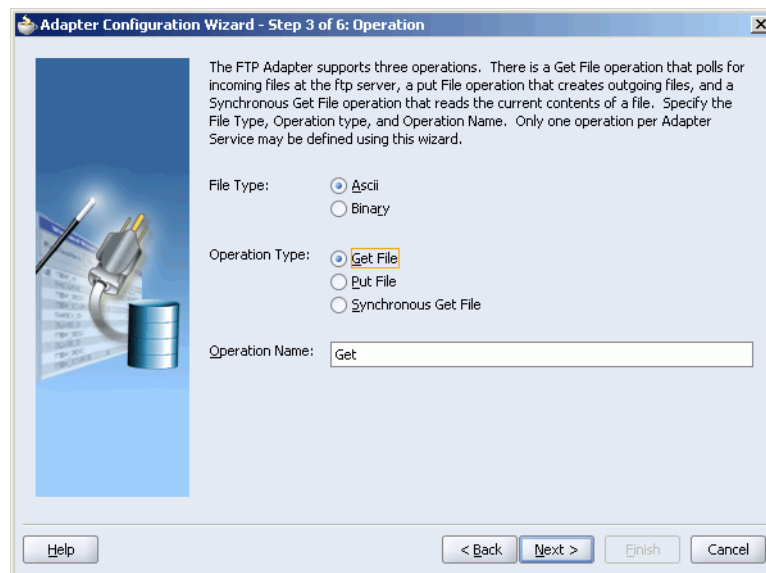
```
<?xml version="1.0"?>
<!DOCTYPE oc4j-connector-factories PUBLIC "-//Oracle//DTD Oracle Connector
  9.04//EN" "http://xmlns.oracle.com/ias/dtds/oc4j-connector-factories-9_04.dtd">

<oc4j-connector-factories>
  <connector-factory location="eis/Ftp/FtpAdapter" connector-name="FTP Adapter">
    <config-property name="host" value="stada55.us.oracle.com"/>
    <config-property name="port" value="21"/>
    <config-property name="username" value="anonymous"/>
    <config-property name="password" value="password"/>
  </connector-factory>
</oc4j-connector-factories>
```

The adapter instance JNDI name is specified as **eis/Ftp/FtpAdapter** and connection information such as host, port, username, and password are provided as configuration properties.

Note: The FTP adapter does not support the FTP commands RESTART and RECOVERY during the transfer of large files.

After logging in, you select the Get File (read) operation and the type of file to deliver. [Figure 2–23](#) shows this selection.

Figure 2–23 Selecting the Get File Operation

For inbound and outbound file transfers, the `ra.xml` file includes a `serverType` property. The location of the `ra.xml` file depends on the installation type. Table 2–8 lists the installation type and corresponding `ra.xml` file location.

Table 2–8 `ra.xml` File Location

Installation Type	File Location
SOA Basic Installation	<code>Oracle_Home\j2ee\home\connectors\FtpAdapter\FtpAdapter\META-INF</code>
Oracle Enterprise Service Bus on Oracle Application Server Middle tier	<code>Oracle_Home\j2ee\homemid\connectors\FtpAdapter\FtpAdapter\META-INF</code>
Oracle BPEL Process Manager on Oracle Application Server Middle tier	<code>Oracle_Home\j2ee\homemid\connectors\FtpAdapter\FtpAdapter\META-INF</code>

The `serverType` property is automatically used to determine line separators when you transfer data. You can specify `unix`, `win`, or `mac` as property values. These values represent the operating system on which the FTP server is running. By default, the `serverType` property contains `unix`.

```
<config-property-name>serverType</config-property-name>
<config-property-type>java.lang.String</config-property-type>
<config-property-value>unix</config-property-value>
```

When you specify `mac` as the value, `\r` is used as line separator. For `unix`, `\n` is used and for `win`, `\r\n` is used.

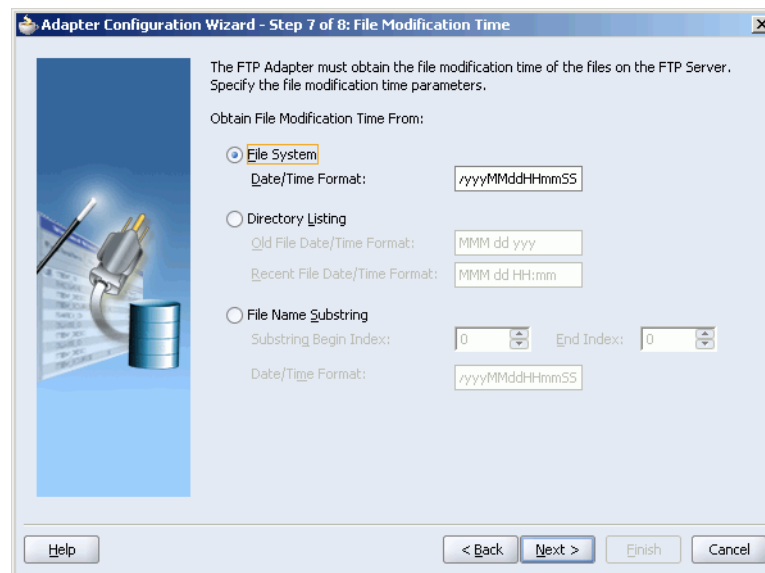
From this point onward, the windows of the Adapter Configuration Wizard for the Get File operation are the same as those for the Read File operation of the FTP adapter. Table 2–9 lists the windows that are displayed and provides references to sections that describe their functionality.

Table 2–9 Adapter Configuration Wizard Windows for Get File Operation

Window	See Section...
File Directories (Figure 2–3)	Section 2.2.1.2, "Inbound File Directory Specifications"
File Filtering (Figure 2–7)	Section 2.2.1.3, "File Matching and Batch Processing"
File Polling (Figure 2–8)	Section 2.2.1.4, "File Polling"
Messages (Figure 2–9)	Section 2.2.1.7, "Native Data Translation"

An additional Adapter Configuration Wizard window is also available for advanced users. This window is shown in Figure 2–24 and appears only after you make either or both of the following selections on the File Polling window shown in Figure 2–8:

- Do not select the **Delete Files After Successful Retrieval** check box.
- Set the value of the **Minimum File Age** field to a value greater than 0.

Figure 2–24 File Modification Time

This window enables you to specify one of the following methods for obtaining the modification time of files on the remote FTP server:

- **File System**

This option enables you to obtain the date/time format of the file modification time with the file system listing command. However, this option is rarely used and is not supported by all FTP servers. See your FTP server documentation to determine whether your server supports the file system listing command, which command-line syntax to use, and how to interpret the output.

For example, if the file system listing command `quote mdm filename` is supported and returns the following information:

```
213 20050602102633
```

specify the start index, end index, and date/time format of the file modification time in the **Data/Time Format** field as a single value separated by commas (for example, **4,18,yyyyMMddHHmmss**).

Where:

- 4 is the start index of the file modification time.
- 18 is the end index of the file modification time.
- yyyyMMddHHmmss is the data/time format of the file modification time obtained with the quote mdtm filename command.

The resulting *service_name.wsdl* file includes the following parameters and values:

```
FileModificationTime="FileSystem"
ModificationTimeFormat="4,18,yyyyMMddHHmmss"
```

To handle the time zone issue, you must also be aware of the time stamp difference. The time zone of the FTP server is determined by using the Windows date/time properties (for example, by double-clicking the time being displayed in the Windows task bar). You must then convert the time difference between the FTP server and the system on which the FTP adapter is running to milliseconds and add the value as a property in the *bpel.xml* file:

```
<activationAgents>
  <activationAgent ...>
    <property name="timestampOffset">259200000</property>
```

■ Directory Listing

This option enables you to obtain the date/time format from the file modification time with the FTP directory listing command. For example, if the directory listing command (*ls -l*) returns the following information:

```
12-27-04 07:44AM                2829 NativeData2.txt
```

specify the start index, end index, and date/time format of the file modification time as a single value separated by commas in either the **Old File Date/Time Format** field or the **Recent File Date/Time Format** field (for example, **0,17, MM-dd-yy hh:mm**).

Where:

- 0 is the start index of the file modification time.
- 17 is the end index of the file modification time.
- MM-dd-yy hh:mm is the date/time format of the file modification time obtained with the *ls -l* command. For this example, the value is entered in the **Recent File Date/Time Format** field. This field indicates that the format is obtained from the most recent file adhering to the naming convention, whereas the **Old File Date/Time Format** field obtains the format from the oldest file.

The resulting *service_name.wsdl* file includes the following parameters and values:

```
FileModificationTime="DirListing"
ModificationTimeFormat="0,17, MM-dd-yy hh:mm"
```

To handle the time zone issue, you must also be aware of the time stamp difference. The time zone of the FTP server is determined by using the Windows date/time properties (for example, by double-clicking the time being displayed in the Windows task bar). You must then convert the time difference between the FTP

server and the system on which the FTP adapter is running to milliseconds and add the value as a property in the `bpel.xml` file:

```
<activationAgents>
  <activationAgent ...>
    <property name="timestampOffset">259200000</property>
```

■ File Name Substring

This option enables you to obtain the modification time from the file name. For example, if the name of the file is `fixedLength_20050324.txt`, you can specify the following values:

- The start index in the **Substring Begin Index** field (for example, **12**).
- The end index in the **End Index** field (for example, **20**).
- The date and time format in the **Date/Time Format** field conforming to the Java `SimpleDateFormat` to indicate the file modification time in the file name (for example, `yyyyMMdd`).

The resulting `service_name.wsdl` file includes the following parameters and values:

```
FileModificationTime="Filename"
FileNameSubstringBegin="12"
FileNameSubstringEnd="20"
ModificationTimeFormat="yyyyMMdd"
```

After the completion of Adapter Configuration Wizard, configuration files are created in the Applications section of Oracle JDeveloper.

The inbound service WSDL file name that is created is also similar to that of the file adapter. The main differences include the operation type and the file type.

```
<pc:inbound_binding />
  <operation name="Get">
<jca:operation
  FileType="binary"
```

The inbound header WSDL file named `ftpAdapterInboundHeader.wsdl` looks as follows:

```
<definitions
  name="fileAdapter"
  targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/ftp/"
  xmlns:tns="http://xmlns.oracle.com/pcbpel/adapter/ftp/"
  xmlns="http://schemas.xmlsoap.org/wsdl/" >
  <types>
    <schema attributeFormDefault="qualified" elementFormDefault="qualified"
      targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/ftp/"
      xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:FTPAPP="http://xmlns.oracle.com/pcbpel/adapter/ftp/">
      <element name="InboundFTPHeaderType">
        <complexType>
          <sequence>
            <element name="fileName" type="string"/>
          </sequence>
        </complexType>
      </element>
    </schema>
  </types>
```



```

<!-- Header Message -->
<message name="InboundHeader_msg">
  <part element="tns:InboundFTPHeaderType" name="inboundHeader"/>
</message>
</definitions>

```

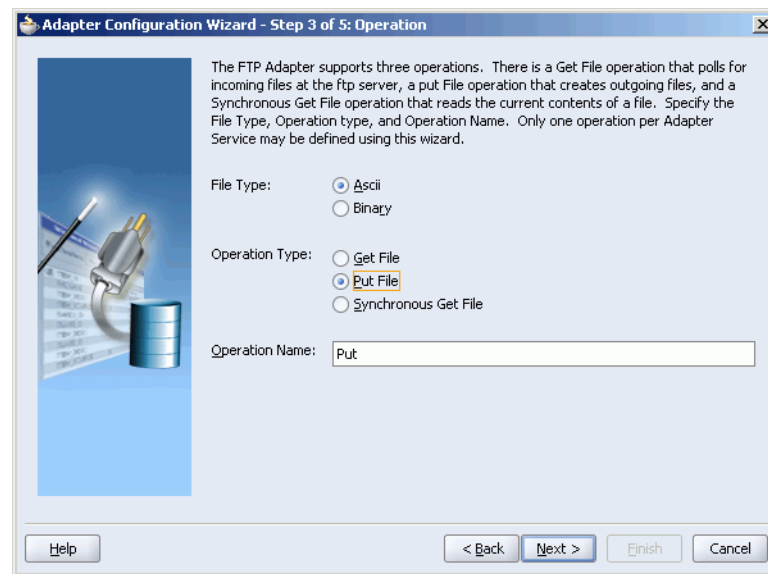
See [Section 2.2.1.8, "Error Handling"](#) and [Section 2.2.1.9, "Guaranteed Delivery and Recovery from Server Failures"](#) for more information about error handling and guaranteed delivery capabilities.

2.2.4 FTP Adapter for Put File Concepts

In the outbound direction, the FTP adapter works the same as the Write File operations of the file adapter. The FTP adapter receives messages from a BPEL process or ESB service and writes the messages in a file to a file system (in this case, remote). Because of this, the Adapter Configuration Wizard prompts you to connect to the FTP server with the adapter instance JNDI name, as shown in [Figure 2–22](#).

After logging in, you select the Put File (write) operation and the type of file to deliver. [Figure 2–25](#) shows this selection.

Figure 2–25 Selecting the Put File Operation



From this point onward, the windows of the Adapter Configuration Wizard window for the Put File operation are the same as those for the Write File operation of the file adapter. [Table 2–10](#) lists the windows that display and provides references to sections that describe their functionality.

Table 2–10 Adapter Configuration Wizard Windows for Put File Operation

Window	See Section...
File Configuration (Figure 2–14)	Section 2.2.2.2, "Outbound File Directory Creation"
Messages (Figure 2–21)	Section 2.2.2.3, "Native Data Translation"

After the completion of the Adapter Configuration Wizard, configuration files are created in the Applications section of Oracle JDeveloper.

The outbound service WSDL file name that is created is also similar to that of the file adapter. The main differences include the operation type and the file type.

```
<pc:inbound_binding />
  <operation name="Put">
<jca:operation
  FileType="binary"
```

The outbound header WSDL file named `ftpAdapterOutboundHeader.wsdl` looks as follows:

```
<definitions
  name="fileAdapter"
  targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/ftp/"
  xmlns:tns="http://xmlns.oracle.com/pcbpel/adapter/ftp/"
  xmlns="http://schemas.xmlsoap.org/wsdl/" >
<types>
  <schema attributeFormDefault="qualified" elementFormDefault="qualified"
    targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/ftp/"
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:FTPAPP="http://xmlns.oracle.com/pcbpel/adapter/ftp/" >
    <element name="OutboundFTPHeaderType">
      <complexType>
        <sequence>
          <element name="fileName" type="string"/>
        </sequence>
      </complexType>
    </element>
  </schema>
</types>

<!-- Header Message -->
<message name="OutboundHeader_msg">
  <part element="tns:OutboundFTPHeaderType" name="outboundHeader" />
</message>
</definitions>
```

See [Section 2.2.2.4, "Outbound Error Handling"](#) for more information about error handling capabilities.

2.3 Using Secure FTP with the FTP Adapter

The FTP adapter supports the use of the secure FTP feature on Solaris. This section provides an overview of secure FTP functionality and describes how to install and configure this feature.

This section contains the following tasks:

- [Secure FTP Overview](#)
- [Installing and Configuring OpenSSL](#)
- [Installing and Configuring vsftpd](#)
- [Creating an Oracle Wallet](#)
- [Setting Up the FTP Adapter](#)

Note: The FTP adapter supports the secure FTP feature on Solaris only.

2.3.1 Secure FTP Overview

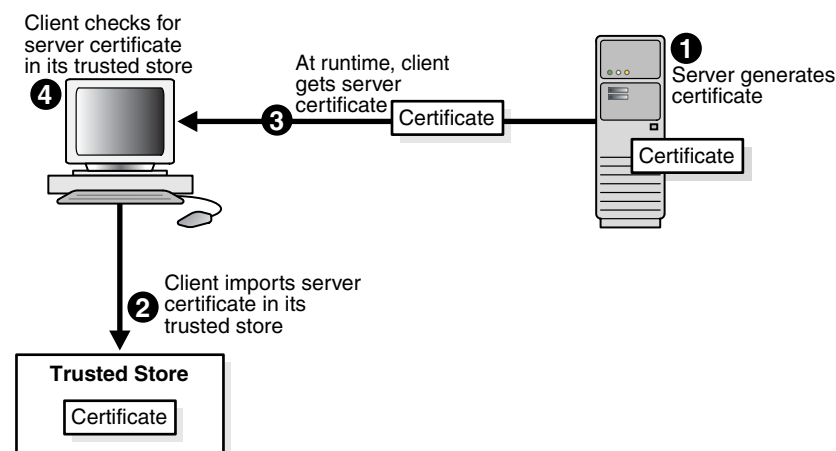
In environments in which sensitive data is transferred to remote servers (for example, sending credit card information to HTTP servers), the issue of security is very important. Security in these cases primarily refers to two requirements:

- Trust in the remote server with which you are exchanging data
- Protection from third parties trying to intercept the data

Secure socket layer (SSL) certificates and encryption focus on satisfying these two security requirements. When SSL is used in the context of FTP, the resulting security mechanism is known as FTPS (or FTP over SSL).

To gain the trust of clients in SSL environments, servers obtain certificates (typically, X.509 certificates) from recognized certificate authorities. When you set up the FTP server `vsftpd` in [Section 2.3.3, "Installing and Configuring vsftpd"](#), you use `openssl` to create a certificate for the server. Every client trusts a few parties to begin. If the server is one of these trusted parties, or if the server's certificate was issued by one of these parties, you have established trust, even indirectly. For example, if the server's certificate was issued by authority A, which has a certificate issued by authority B, and the client trusts B, that is good enough. For the setup shown in [Figure 2-7](#), the server's certificate is directly imported into the client's certificate store (or Oracle Wallet) as a trusted certificate.

Figure 2-26 Establishing Trust



You make the data being transferred immune to spying by encrypting it before sending it and decrypting it after receiving it. Symmetric encryption (using the same key to encrypt and decrypt data) is much faster for large amounts of data than the public key and private key approach. Symmetric encryption is the approach used by FTPS. However, before the client and server can use the same key to encrypt and decrypt data, they must agree on a common key. This is typically done by the client performing the following tasks:

- Generating a session key (to be used to encrypt and decrypt data)
- Encrypting this session key using the server's public key that is part of the server's certificate
- Sending the key to the server

The server decrypts this session key using its private key and subsequently uses it to encrypt file data before sending it to the client.

The remaining subsections describe how to install and configure secure FTP.

2.3.2 Installing and Configuring OpenSSL

OpenSSL is an open source implementation of the SSL protocol. OpenSSL implements basic cryptographic functions and provides utility functions. Install and configure OpenSSL on the Solaris host to use as the FTP server.

1. Go to the following URL:

```
http://www.openssl.org/source
```

2. Locate `openssl-0.9.7g.tar.gz` in the list of available files. For example:

```
3132217 Apr 11 17:21:51 2005 openssl-0.9.7g.tar.gz (MD5) (PGP sign)
```

3. Download the following files:

- `openssl-0.9.7g.tar.gz`
- `openssl-0.9.7g.tar.gz.md5` (under the MD5 link)
- `openssl-0.9.7g.tar.gz.asc` (under the PGP sign link)

4. Unzip the following file using `gunzip`.

```
gunzip openssl-0.9.7g.tar.gz
```

5. Untar the following file:

```
tar xvf openssl-0.9.7g.tar
```

6. Change directories to the following location:

```
cd openssl-0.9.7g
```

7. Run the following command:

```
./config --prefix=/usr --openssldir=/usr/local/openssl
```

8. Change to the Bourne shell (if not already using it):

```
sh
```

9. Configure and export the `PATH`:

```
PATH=${PATH}:/usr/ccs/bin; export PATH
```

10. Run the following command:

```
make
```

11. Exit the Bourne shell:

```
exit
```

12. Run the following command:

```
make test
```

13. Log in as the super user:

```
msu
```

14. Enter the password when prompted.

15. Run the following command:

```
make install
```

2.3.3 Installing and Configuring vsftpd

The vsftpd server is a secure and fast FTP server for UNIX systems. Install and configure vsftpd on the Solaris host to use as the FTP server.

1. Go to the following location:

```
ftp://vsftpd.beasts.org/users/cevans/
```

2. Download vsftpd-2.0.3 (You need the tar and signature file (.asc file)). For example:

[BINARY]	vsftpd-2.0.3.tar.gz	[Mar 19 21:26]	149K
[FILE]	vsftpd-2.0.3.tar.gz.asc	[Mar 19 21:26]	189B

3. Unzip the following file using gunzip.

```
gunzip vsftpd-2.0.3.tar.gz
```

4. Unzip the tar file:

```
tar xvf vsftpd-2.0.3.tar
```

5. Change directories to the following location:

```
cd vsftpd-2.0.3
```

6. Make the following change in the builddefs.h file:

```
#undef VSF_BUILD_SSL

to

#define VSF_BUILD_SSL
```

7. Log in as the super user:

```
msu
```

8. Enter the password when prompted.

9. Create a file named vsftpd.conf with the following settings in the /etc directory:

```
# Standalone mode
listen=YES
max_clients=200
max_per_ip=4
# Access rights
anonymous_enable=YES
#chroot_local_user=YES
#userlist_enable=YES
ftp_username=ftp
local_enable=YES
write_enable=YES
anon_upload_enable=YES
anon_mkdir_write_enable=YES
anon_other_write_enable=YES
chown_uploads=YES
chown_username=ftp
```

```
# Security
anon_world_readable_only=NO
allow_anon_ssl=YES
ssl_enable=YES
connect_from_port_20=YES
hide_ids=YES
pasv_min_port=50000
pasv_max_port=60000
# Features
ftpd_banner="Welcome to the FTP Service"
xferlog_enable=YES
ls_recurse_enable=NO
ascii_download_enable=NO
async_abor_enable=YES
# Performance
one_process_model=NO
idle_session_timeout=120
data_connection_timeout=300
accept_timeout=60
connect_timeout=60
anon_max_rate=50000
```

Note: Copies of the `vsftpd.conf` file appear in several locations in the `vsftpd-2.0.3` directory structure. If you use one of those files to create the `vsftpd.conf` file in the `/etc` directory, then ensure that it only includes the parameters and settings described in Step 9.

10. Run the following commands:

```
mkdir /var/ftp
useradd -d /var/ftp ftp
chown root /var/ftp
chmod og-w /var/ftp
mkdir /usr/share/empty
mkdir /usr/share/ssl
mkdir /usr/share/ssl/certs
```

11. Run the following command:

```
openssl req -x509 -nodes -newkey rsa:1024 -keyout
/usr/share/ssl/certs/vsftpd.pem -out /usr/share/ssl/certs/vsftpd.pem
```

12. Run the `vsftpd` daemon from the `vsftpd-2.0.3` directory:

```
./vsftpd
```

2.3.4 Creating an Oracle Wallet

Oracle Wallet Manager is an application for managing and editing security credentials in Oracle wallets. A wallet is a password-protected container that stores authentication and signing credentials, including private keys, certificates, and trusted certificates, all of which are used by SSL for strong authentication.

1. Create a new wallet in Oracle Wallet Manager.
2. Import `vsftpd.pem` from Step 11 of [Section 2.3.3, "Installing and Configuring vsftpd"](#) as a trusted certificate in this wallet.
3. Save this wallet in PKCS # 12 (`.p12`) format.

See the *Oracle Application Server Administrator's Guide* for details about using Oracle Wallet Manager.

2.3.5 Setting Up the FTP Adapter

Perform the following tasks to set up the FTP adapter:

1. On your Solaris host, run the following commands:

```
mkdir /var/ftp/inDir
mkdir /var/ftp/outDir
chmod 777 /var/ftp/inDir /var/ftp/outDir
```

2. Specify the FTP connection parameters in the FTP adapter `oc4j-ra.xml` file. The location of this file is based on the installation type you selected. Refer to [Table 2-7](#) for information about location of the `oc4j-ra.xml` file.

A sample `oc4j-ra.xml` is as follows:

```
<connector-factory location="eis/Ftp/FtpAdapter" connector-name="FTP Adapter">
  <config-property name="host" value="usunnbf29.us.oracle.com"/>
  <config-property name="port" value="21"/>
  <config-property name="username" value="ftp"/>
  <config-property name="password" value="password"/>
  <config-property name="useFtps" value="true"/>
  <config-property name="walletLocation" value="D:\wallet\ewallet.p12"/>
  <config-property name="walletPassword" value="welcome1"/>
  <config-property name="channelMask" value="both"/>
  <config-property name="securePort" value="990"/>
</connector-factory>
```

Where...	Is...
useFtps	Set to True. This setting is required to use FTP over SSL. The default is False.
walletLocation	The location of the wallet created in Section 2.3.4, "Creating an Oracle Wallet"
walletPassword	The password of the wallet
channelMask	The type of channel: control channel or data channel. Possible values are both, control, data, or none. The default is both.
securePort	The port for FTP over SSL. The default is 990.

3. Restart Oracle BPEL Server after changing the `oc4j-ra.xml` file.

You have now installed and configured secure FTP and are ready to use this feature with the FTP adapter.

2.4 Using SFTP with the FTP Adapter

SSH file transfer protocol (SFTP) is a network protocol that enables secure file transfer over the network. The FTP adapter supports the use of the SFTP feature on Windows and Linux. This section provides an overview of SFTP functionality and describes how to install and configure this feature.

This section contains the following tasks:

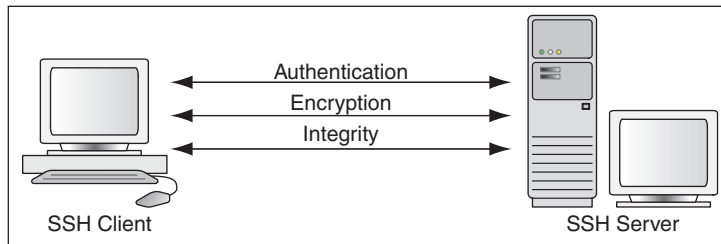
- [SFTP Overview](#)

- [Install and Configure OpenSSH for Windows](#)
- [Set up the FTP Adapter for SFTP](#)

2.4.1 SFTP Overview

SFTP is the network protocol that enables clients to securely transfer files over the underlying SSH transport. SFTP is not similar to FTP over SSH or File Transfer Protocol (FTP). [Figure 2-27](#) displays the communication process between an SSH client and an SSH server.

Figure 2-27 SFTP Communication



SFTP has the following features:

- [Encryption](#)
- [Authentication](#)
- [Integrity](#)
- [Data Compression](#)

2.4.1.1 Encryption

The SSH protocol uses public key cryptography for encryption. The following list explains how the data is encrypted:

1. The SSH subsystem uses one of the symmetric key ciphers such as Data Encryption Standard (DES) or Blowfish to generate a session key. The SSH protocol currently uses the Diffie-Hellman Key Exchange Algorithm to derive the symmetric key for the session.
2. The data is encrypted using the session key
3. The session key is encrypted using the recipient's public key. Because the recipient already has the private key, it can decrypt the message using its preferred PKI algorithm such as Rivest-Shamir-Adleman (RSA) or Digital Signature Algorithm (DSA).

2.4.1.2 Authentication

The SSH protocol inherently supports password authentication by encrypting passwords or session keys as they are transferred over the network. In addition, SSH protocol uses a mechanism known as 'known hosts' to prevent threats such as IP Spoofing. By using this mechanism, both the client and the server have to prove their identity to each other before any kind of communication exchange.

2.4.1.3 Integrity

The SSH protocol makes use of widely trusted bulk hashing algorithms such as Message Digest Algorithm 5 (MD5) or Secure Hash Algorithm (SHA-1) to prevent

insertion attacks. Implementation of data integrity checksum by using the algorithms mentioned in [Section 2.4.1.1, "Encryption"](#), prevents deliberate tampering of data during transmission.

2.4.1.4 Data Compression

The SSH protocol supports zlib, an open-source cross-platform algorithm for data-compression. SSH uses zlib to compress in-flight data in order to reduce network bandwidth.

2.4.2 Install and Configure OpenSSH for Windows

OpenSSH for Windows is the free implementation of SSH protocol on Windows. Perform the following steps to install and configure OpenSSH on Windows XP:

1. Login as a user with Administrator privileges.
2. Download the `setup.exe` from the following location:
`http://www.cygwin.com`
3. Run `setup.exe`. The Cygwin Net Release Setup window is displayed.
4. Click **Next**. The Choose Installation type window is displayed.
5. Select **Install from Internet** as the download source and click **Next**. The Choose Installation Directory window is displayed.
6. Leave the Root Directory as `C:\cygwin`. Also, keep the default options for the Install For and the Default Text File Type fields.
7. Click **Next**. The Select Local Package Directory window is displayed.
8. Click **Browse** and select `C:\cygwin` as the local package directory.
9. Click **Next**. The Select Connection Type window is displayed.
10. Select a setting for Internet connection and click **Next**. The Choose Download Site(s) window is displayed.
11. Select a site from the Available Download Sites list and click **Next**. The Select Packages window is displayed.
12. Click **View** to see the complete list of packages available for installation.
13. Select **openssh** if it is not selected by default.
14. Select the **Binaries** box for openssh.
15. Click **Next** to start the installation.
16. On Windows XP desktop, right-click **My Computer** and select **Properties**.
17. Click the **Advanced** tab and click **Environment Variables**.
18. Click **New** and enter `CYGWIN` in the **Variable Name** field and `ntsec` in the **Variable Value** field.
19. Add `C:\cygwin\bin` to the system path.
20. Open the cygwin window.
21. Type `ssh-host-config`.
22. You will be prompted with following questions:
 - a. Shall privilege separation be used? (yes/no)

Enter yes.

- b. Shall this script create a local user 'sshd' on this machine?

Enter yes.

- c. Do you want to install sshd as service?

(Say "no" if it's already installed as service) (yes/no)

Enter yes.

- d. Which value should the environment variable CYGWIN have when sshd starts? It's recommended to set at least "ntsec" to be able to change user context without password. Default is "binmode ntsec tty".

Enter ntsec.

23. Type `net start sshd` to start the sshd service.

24. Run the following command in the cygwin window to replicate the Windows local user accounts to cygwin:

```
mkpasswd --local > /etc/passwd
mkgroup --local > /etc/group
```

25. To test the setup, type `ssh localhost` in the cygwin window.

2.4.3 Set up the FTP Adapter for SFTP

To use the SFTP functionality, you need to modify the `oc4j-ra.xml` file. Refer to [Table 2-7](#) for information about location of the `oc4j-ra.xml` file.

[Table 2-11](#) lists the parameters for which you need to specify a value in the `oc4j-ra.xml` file. The values of these parameters depend on the type of authentication and the location of OpenSSH.

Table 2-11 SFTP Parameters

Parameter	Description	Example
useSftp	Specify true. Mandatory: Yes Default value: false	<code><config-property name="useSftp" value="false"/></code>

Table 2–11 (Cont.) SFTP Parameters

Parameter	Description	Example
authenticationType	<p>Specify PASSWORD for password-based authentication or PUBLICKEY for public key authentication.</p> <p>For Password-based authentication, the user name and password specified in the oc4j-ra.xml file are used. Ensure that there is a Windows user with the same name and password as specified in the oc4j-ra.xml file. In addition, the user should have administrative privileges.</p> <p>For Public key authentication, the privateKeyFile parameter must be set to the location of the private key file.</p> <p>Mandatory: Yes</p>	<pre><config-property name="authenticationType" value="password" /></pre>
preferredKeyExchangeAlgorithm	<p>Specify diffie-hellman-group1-sha1 or diffie-hellman-group-exchange-sha1.</p> <p>This is an optional parameter where the user can select the default key exchange protocol for negotiating the session key for encrypting the message.</p> <p>Mandatory: No</p> <p>Default value: diffie-hellman-group1-sha1</p>	<pre><config-property name="preferredKeyExchangeAlgorithm" value="diffie-hellman-group1-sha1" /></pre>
preferredCompressionAlgorithm	<p>Specify none or zlib.</p> <p>This parameter enables the user to choose whether in-flight data should be compressed or not.</p> <p>Mandatory: No</p>	<pre><config-property name="preferredCompressionAlgorithm" value="none" /></pre>
preferredDataIntegrityAlgorithm	<p>Specify hmac-md5 or hmac-sha1.</p> <p>This parameter enables the user to select the bulk-hashing algorithm for data integrity checks.</p> <p>Mandatory: No</p> <p>Default value: hmac-md5</p>	<pre><config-property name="preferredDataIntegrityAlgorithm" value="hmac-md5" /></pre>

Table 2–11 (Cont.) SFTP Parameters

Parameter	Description	Example
preferredPKIAlgorithm	Specify ssh-rsa or ssh-dsa. This parameter enables the user to configure the asymmetric cipher for the communication. Mandatory: No Default value: ssh-rsa	<pre><config-property name="preferredPKIAlgorithm" value="ssh-rsa"/></pre>
privateKeyFile	Specify the path to the private key file. This is required if the authenticationType parameter is set to PUBLICKEY. Mandatory: No	<pre><config-property name="privateKeyFile" value=" " /></pre>
preferredCipherSuite	Specify one of the following ciphers: <ul style="list-style-type: none"> ■ twofish192-cbc ■ cast128-cbc ■ twofish256-cbc ■ aes128-cbc ■ twofish128-cbc ■ 3des-cbc ■ blowfish-cbc ■ aes256-cbc ■ aes192-cbc Mandatory: No Default value: blowfish-cbc	<pre><config-property name="preferredCipherSuite" value="blowfish-cbc"/></pre>
transportProvider	Specify socket or HTTP. Specify socket if the SSH server is inside the firewall. Specify HTTP if the SSH server is outside the firewall or the server is exposed through a HTTP server. If you select HTTP, then you must provide values for the following parameters: <ul style="list-style-type: none"> ■ proxyHost ■ proxyPort ■ proxyUser ■ proxyPassword ■ useProxy Mandatory: Yes	<pre><config-property name="transportProvider" value="socket"/></pre>

2.4.3.1 Configuring the FTP Adapter for Password Authentication

Perform the following steps to set up the FTP adapter for password authentication:

1. In the `oc4j-ra.xml` file, specify the values of the parameters listed in [Table 2–11](#). Ensure that the `authenticationType` parameter is set to `password`.
2. Restart the server after modifying the `oc4j-ra.xml` file.

A sample `oc4j-ra.xml` file with password authentication parameters is shown in the following example:

```
<config-property name="useSftp" value="true"/>
<config-property name="authenticationType" value="password"/>
<config-property name="preferredKeyExchangeAlgorithm"
value="diffie-hellman-group1-sha1"/>
<config-property name="preferredCompressionAlgorithm" value="none"/>
<config-property name="preferredDataIntegrityAlgorithm" value="hmac-md5"/>
<config-property name="preferredPKIAlgorithm" value="ssh-rsa"/>
<config-property name="privateKeyFile" value=""/>
<config-property name="preferredCipherSuite" value="blowfish-cbc"/>
<config-property name="transportProvider" value="socket"/>
```

2.4.3.2 Configuring the FTP Adapter for Public Key Authentication

For public key authentication, you first need to configure OpenSSH and then set up the FTP adapter. The FTP adapter set up depends on whether the OpenSSH is running inside the firewall or outside the firewall. If OpenSSH is running inside the firewall, then refer to the following sections:

- [Configuring the OpenSSH for Public-Key Authentication](#)
- [Configuring the FTP Adapter for Public Key Authentication with OpenSSH Running Inside the Firewall](#)

If OpenSSH is running outside the firewall, then refer to the following sections:

- [Configuring the OpenSSH for Public-Key Authentication](#)
- [Configuring the FTP Adapter for Public Key Authentication with OpenSSH Running Outside the Firewall](#)

Configuring the OpenSSH for Public-Key Authentication

Perform the following steps:

1. Go to `C:\cygwin\etc` folder. If required, configure the `sshd_config` file to force public key authentication. Refer to `openssh help` or manual for the information.
2. Go to the `C:\cygwin\bin` directory.
3. Run the following command to generate the key pair:

```
ssh-keygen -t rsa
```

4. Enter `/etc/id_rsa` when prompted for the file in which the key should be saved.
5. Enter the passphrase.
6. Enter the passphrase again.
7. Go to the `/etc` directory and verify that both the public key file (`id_rsa.pub`) and the private key file (`id_rsa`) are generated.
8. Run the following command to create a copy of the public key file:

```
cp id_rsa.pub authorized_keys
```

9. Create a copy of the private key file to a secured location such as `C:\my-secured-folder\`. The FTP adapter configuration will refer to this private key file.
10. Restart the OpenSSH server by running the following commands:

```
net stop sshd
net start sshd
```

Configuring the FTP Adapter for Public Key Authentication with OpenSSH Running Inside the Firewall

Perform the following steps to set up the FTP adapter for public key authentication:

1. In the `oc4j-ra.xml` file, specify the values of the parameters listed in [Table 2–11](#). Ensure that the `authenticationType` parameter is set to `publickey` and the `transportProvider` parameter is set to `socket`. The `privateKeyFile` parameters should contain the location of the private key file.
2. Restart the server after modifying the `oc4j-ra.xml` file.

A sample `oc4j-ra.xml` file with public key authentication parameters is shown in the following example:

```
<config-property name="useSftp" value="true"/>
<config-property name="authenticationType" value="publickey"/>
<config-property name="preferredKeyExchangeAlgorithm"
value="diffie-hellman-group1-sha1"/>
<config-property name="preferredCompressionAlgorithm" value="none"/>
<config-property name="preferredDataIntegrityAlgorithm" value="hmac-md5"/>
<config-property name="preferredPKIAlgorithm" value="ssh-rsa"/>
<config-property name="privateKeyFile" value="C:\my-secured-folder\id_rsa"/>
<config-property name="preferredCipherSuite" value="blowfish-cbc"/>
<config-property name="transportProvider" value="socket"/>
```

Configuring the FTP Adapter for Public Key Authentication with OpenSSH Running Outside the Firewall

Perform the following steps to set up the FTP adapter for public key authentication when OpenSSH is running outside the firewall:

1. In the `oc4j-ra.xml` file, specify the values of the parameters listed in [Table 2–11](#). Ensure that the `authenticationType` parameter is set to `publickey` and the `transportProvider` parameter is set to `HTTP`. The `privateKeyFile` parameter contains the location of the private key file.
2. In the `oc4j-ra.xml` file, also specify the following proxy-related parameters:
 - `proxyHost`: The name of the proxy host.
 - `proxyPort`: The port number of the proxy.
 - `proxyUsername`: The user name for the proxy.
 - `proxyPassword`: The password for the proxy.
 - `useProxy`: Specify `true` to use proxy.
3. Restart the server after modifying the `oc4j-ra.xml` file.

A sample `oc4j-ra.xml` file with public key authentication parameters and proxy parameters is shown in the following example:

```
<config-property name="proxyHost" value="proxy.host.com"/>
```

```

<config-property name="proxyPort" value="80"/>
<config-property name="proxyUsername" value="anonymous"/>
<config-property name="proxyPassword" value="tiger@scott.com"/>
<config-property name="useProxy" value="true"/>
<config-property name="useSftp" value="true"/>
<config-property name="authenticationType" value="publickey"/>
<config-property name="preferredKeyExchangeAlgorithm"
value="diffie-hellman-group1-sha1"/>
<config-property name="preferredCompressionAlgorithm" value="none"/>
<config-property name="preferredDataIntegrityAlgorithm" value="hmac-md5"/>
<config-property name="preferredPKIAlgorithm" value="ssh-rsa"/>
<config-property name="privateKeyFile" value="C:\my-secured-folder\id_rsa"/>
<config-property name="preferredCipherSuite" value="blowfish-cbc"/>
<config-property name="transportProvider" value="HTTP"/>

```

2.5 Configuring the FTP Adapter for HTTP Proxy

The FTP adapter provides proxy support for HTTP proxy only. The HTTP proxy support is available in the following two modes: plain FTP mode and SFTP mode. This section explains how to configure the FTP adapter for running in plain FTP mode and SFTP mode. It contains following topics:

- [Configuring for Plain FTP Mode](#)
- [Configuring for SFTP Mode](#)

2.5.1 Configuring for Plain FTP Mode

For running the FTP adapter in plain FTP mode, you need to specify the value of certain parameters in the `oc4j-ra.xml` file. Refer to [Table 2-7](#) for information about the location of the `oc4j-ra.xml` file. [Table 2-12](#) lists the parameters that you need to modify.

Table 2-12 Plain FTP Mode Parameters

Parameter	Description	Example
host	The remote FTP server name.	<code><config-property name="host" value="my.host.com"/></code>
port	The FTP control port number.	<code><config-property name="port" value="21"/></code>
username	The FTP user name.	<code><config-property name="username" value="scott"/></code>
password	The FTP password.	<code><config-property name="password" value="password"/></code>
proxyHost	The proxy host name.	<code><config-property name="proxyHost" value="proxy.host.com"/></code>
proxyPort	The proxy port number.	<code><config-property name="proxyPort" value="80"/></code>

Table 2–12 (Cont.) Plain FTP Mode Parameters

Parameter	Description	Example
proxyUsername	The proxy user name.	<code><config-property name="proxyUsername" value="anonymous" /></code>
proxyPassword	The proxy password.	<code><config-property name="proxyPassword" value="tiger@scott.co m" /></code>
proxyType	The proxy type. Only HTTP proxy type is supported.	<code><config-property name="proxyType" value="http" /></code>
proxyDefinitionFile	The absolute path of the proxy definition file. This parameter is not mandatory. See Section 2.5.1.1, "Proxy Definition File" for more information.	<code><config-property name="proxyDefinition File" value="c:\ proxydefinitions.xml " /></code>
useProxy	Specify true to use proxy.	<code><config-property name="useProxy" value="true" /></code>

A sample `oc4j-ra.xml` file is shown in the following example:

```
<connector-factory location="eis/Ftp/FtpAdapter" connector-name="Ftp Adapter">
<config-property name="host" value="my.host.com"/>
<config-property name="port" value="21"/>
<config-property name="username" value="user"/>
<config-property name="password" value="password"/>
<config-property name="proxyHost" value="proxy.host.com"/>
<config-property name="proxyPort" value="80"/>
<config-property name="proxyUsername" value="anonymous"/>
<config-property name="proxyPassword" value="tiger@scott.com"/>
<config-property name="proxyType" value="http"/>
<config-property name="proxyDefinitionFile" value="c:\proxydefinitions.xml"/>
<config-property name="useProxy" value="true"/>
</connector-factory>
```

2.5.1.1 Proxy Definition File

You can specify all proxy-specific information in a proxy definition file and configure the adapter to use this file with the `proxyDefinitionFile` parameter of the `oc4j-ra.xml` file. A proxy definition file is written in XML format and is based on XML schema. The XML schema for the proxy definition file is shown in [Example 2–1](#). Your proxy definition file must be based on this XML schema.

Example 2–1 Proxy Definition File XML Schema

```
<?xml version = \"1.0\" encoding = \"UTF-8\"?>
<schema targetNamespace = \"http://ns.oracle.com/ip/af/ftp/proxy\" xmlns =
\"http://www.w3.org/2001/XMLSchema\"
xmlns:proxy=\"http://ns.oracle.com/ip/af/ftp/proxy">

    <element name="ProxyDefinitions" type="proxy:ProxyDefinitionsType"/>
    <complexType name="ProxyDefinitionsType">
```



```

        <sequence>
            <element name="Proxy" type="proxy:ProxyDefinition"
                minOccurs="0" maxOccurs="unbounded" />
        </sequence>
    </complexType>

    <complexType name="ProxyDefinition">
        <sequence>
            <element name="Step" type="proxy:StepType"
                minOccurs="1" maxOccurs="unbounded" />
        </sequence>
        <attribute name="key" type="ID" use="required" />
        <attribute name="description" type="string"
            use="required" />
        <attribute name="type" type="proxy:Protocol"
            use="optional" />
    </complexType>

    <complexType name="StepType">
        <simpleContent>
            <extension base="string">
                <attribute name="command" type="string" use="required" />
                <attribute name="args" type="string" use="required" />
            </extension>
        </simpleContent>
    </complexType>

    <simpleType name="Protocol">
        <restriction base="string">
            <enumeration value="ftp" />
            <enumeration value="http" />
        </restriction>
    </simpleType>
</schema>

```

A sample proxy definition file, based on the XML schema in [Example 2-1](#), would look as shown in [Example 2-2](#):

Example 2-2 Proxy Definition File

```

<?xml version = '1.0' standalone = 'yes'?>
<proxy:ProxyDefinitions xmlns:proxy="http://ns.oracle.com/ip/af/ftp/proxy">
  <Proxy key="http" description="http" type="http">
    <Step command="USER" args="remote_username" />
    <Step command="PASS" args="remote_password" />
  </Proxy>
</proxy:ProxyDefinitions>

```

When you use the file in [Example 2-2](#), the FTP adapter sends the following sequence of commands to login:

1. USER remote_username
2. PASS remote_password

You can also direct the proxy definition file to pick values from the `oc4j-ra.xml` file. You can use the following expressions for this:

- `$proxy.user`: This corresponds to the value of the `proxyUsername` parameter in the `oc4j-ra.xml` file.

- `$proxy.pass`: This corresponds to the value of the `proxyPassword` parameter in the `oc4j-ra.xml` file.
- `$remote.user`: This corresponds to the value of the `username` parameter in the `oc4j-ra.xml` file.
- `$remote.pass`: This corresponds to the value of the `password` parameter in the `oc4j-ra.xml` file.
- `$remote.host`: This corresponds to the value of the `host` parameter in the `oc4j-ra.xml` file.
- `$remote.port`: This corresponds to the value of the `port` parameter in the `oc4j-ra.xml` file.

A sample proxy definition file based on the XML schema in [Example 2-2](#) and taking values from the `oc4j-ra.xml` file is shown in [Example 2-3](#):

Example 2-3 Proxy Definition File Taking Values from the `oc4j-ra.xml` File

```
<?xml version = '1.0' standalone = 'yes'?>
<proxy:ProxyDefinitions xmlns:proxy="http://ns.oracle.com/ip/af/ftp/proxy">
<Proxy key="http" description="http" type="http">
<Step command="USER" args="$remote.user" />
<Step command="PASS" args="$remote.pass" />
</Proxy>
</proxy:ProxyDefinitions>
```

2.5.2 Configuring for SFTP Mode

For running the FTP adapter in SFTP mode, you need to specify the value of certain parameters in the `oc4j-ra.xml` file. Refer to [Table 2-7](#) for information about the location of the `oc4j-ra.xml` file. [Table 2-13](#) lists the parameters which you need to modify.

Table 2-13 SFTP Mode Parameters

Parameter	Description	Example
host	The remote FTP server name.	<code><config-property name="host" value="my.host.com" /></code>
port	The FTP control port number.	<code><config-property name="port" value="22" /></code>
username	The SFTP user name.	<code><config-property name="username" value="scott" /></code>
password	The SFTP password.	<code><config-property name="password" value="password" /></code>
proxyHost	The proxy sever host name.	<code><config-property name="proxyHost" value="proxy.host.com" /></code>
proxyPort	The proxy port number.	<code><config-property name="proxyPort" value="80" /></code>

Table 2–13 (Cont.) SFTP Mode Parameters

Parameter	Description	Example
proxyUsername	The proxy user name.	<code><config-property name="proxyUsername" value="anonymous" /></code>
proxyPassword	The proxy password.	<code><config-property name="proxyPassword" value="tiger@scott.com" /></code>
useSftp	Specify true for SFTP mode. This value is required to use the SFTP feature.	<code><config-property name="useSftp" value="true" /></code>
authenticationType	Specify either PASSWORD or PUBLICKEY. PASSWORD Refer to Section 2.4.3, "Set up the FTP Adapter for SFTP"	<code><config-property name="authenticationType" value="password" /></code>
transportProvider	Specify http as value. Only HTTP transport provider is supported.	<code><config-property name="transportProvider" value="http" /></code>

A sample `oc4j-ra.xml` file is shown in the following example:

```
<connector-factory location="eis/Ftp/FtpAdapter" connector-name="Ftp Adapter">
<config-property name="host" value="my.host.com" />
<config-property name="port" value="22" />
<config-property name="username" value="user" />
<config-property name="password" value="password" />
<config-property name="proxyHost" value="proxy.host.com" />
<config-property name="proxyPort" value="80" />
<config-property name="proxyUsername" value="anonymous" />
<config-property name="proxyPassword" value="password" />
<config-property name="useSftp" value="true" />
<config-property name="authenticationType" value="password" />
<config-property name="transportProvider" value="http" />
</connector-factory>
```

2.6 File and FTP Adapters Use Cases for Oracle BPEL Process Manager

Oracle BPEL Process Manager includes a number of demonstrations of file and FTP adapters. Some of these demonstrations are described in Readme files. Others are described in the Oracle BPEL Process Manager documentation set. This section provides an overview of these demonstrations and a reference to documentation that more fully describes the scenarios.

This section contains the following topics:

- [File Adapter Use Cases for Oracle BPEL Process Manager](#)
- [FTP Adapter Use Case for Oracle BPEL Process Manager](#)

2.6.1 File Adapter Use Cases for Oracle BPEL Process Manager

This section provides an overview of file adapter demonstrations.

This section contains the following topics:

- [File Reading](#)
- [Message Debatching](#)
- [Reading Delimited Content Files](#)
- [Reading Positional \(Fixed Length\) Content Files](#)
- [File Writing](#)

2.6.1.1 File Reading

Several file reading demonstrations are available:

- A complex structure demonstration shows how to use the file read and write functionality of the file adapter. The sample for a complex structure is available at the following location:

`Oracle_Home\bpel\samples\tutorials\121.FileAdapter\ComplexStructure`

- A simple file reading demonstration is provided as part of a larger tutorial that guides you through the design and execution of a sophisticated process that uses synchronous and asynchronous services, parallel flows of execution, conditional branching logic, fault handling and exceptions management, transformations, the file adapter, the database adapter, human workflow, notification, and sensor functionality. In the file reading portion, you configure the file adapter to read an inbound purchase order request from a file in a directory. See Oracle BPEL Process Manager Order Booking Tutorial for a tutorial that uses the file reading functionality of the file adapter.

2.6.1.2 Message Debatching

This demonstration shows how the file adapter processes native data containing multiple messages defined in a custom format. The file adapter takes a single inbound file of two records and writes each record to its own file. For a demonstration of message debatching, go to

`Oracle_Home\bpel\samples\tutorials\121.FileAdapter\Debatching`

2.6.1.3 Reading Delimited Content Files

This demonstration shows how the file adapter reads CSV-formatted entries in an address book, transforms the file contents using XSLT, and stores the data in a fixed-length formatted file. For a demonstration of delimited content files, go to

`Oracle_Home\bpel\samples\tutorials\121.FileAdapter\FlatStructure`

2.6.1.4 Reading Positional (Fixed Length) Content Files

This demonstration shows how the file adapter reads CSV-formatted entries in an address book, transforms the file contents using XSLT, and stores the data in a fixed-length formatted file. For a positional (fixed length) demonstration, go to

`Oracle_Home\bpel\samples\tutorials\121.FileAdapter\FlatStructure`

2.6.1.5 File Writing

A simple file writing demonstration is provided as part of the larger tutorial described in [Section 2.6.1.1, "File Reading"](#). In the file writing portion, you configure the file adapter to write an outbound purchase order acknowledgment to a file in a directory.

See *Oracle BPEL Process Manager Order Booking Tutorial* for a demonstration that uses the file writing functionality of the file adapter.

2.6.2 FTP Adapter Use Case for Oracle BPEL Process Manager

This demonstration shows how the file adapter processes native data containing multiple messages defined in a custom format. The native data instance contains an invoice and purchase order.

In the inbound direction, the FTP adapter retrieves a remote file, processes the file, and publishes the invoice and purchase order separately to the debatching BPEL process.

In the outbound direction, only purchase orders are generated. The debatching BPEL process transforms an invoice to a purchase order. The purchase record is simply copied over. All purchase orders are then written in separate remote output files. For an FTP demonstration, go to

`Oracle_Home\bpel\samples\tutorials\129.FTPAdapter\FTPDebatching`

2.7 File and FTP Adapters Use Case for Oracle Enterprise Service Bus

In this use case, Oracle Enterprise Service Bus receives the customer data from a file system as a text file, through an inbound file adapter service named `ReadFile`. The `ReadFile` adapter service sends the message to a routing service named `ReadFile_RS`. The `ReadFile_RS` sends the message to the outbound adapter service `WriteFTP`. The `WriteFTP` service delivers the message to its associated external application.

This use case consists of following sections:

- [Section 2.7.1, "Prerequisites"](#)
- [Section 2.7.4, "Creating the Inbound File Adapter Service"](#)
- [Section 2.7.5, "Creating the Outbound FTP Adapter Service"](#)
- [Section 2.7.6, "Creating the Routing Rule"](#)
- [Section 2.7.8, "Run-Time Task"](#)

2.7.1 Prerequisites

This example assumes that you are familiar with basic ESB constructs, such as services, routing service, and Oracle JDeveloper environment for creating and deploying ESB services.

To define the schema of the messages, you require an `address-csv.xsd` file. This file can be copied from the following location:

`ORAHOME\bpel\samples\tutorials\121.FileAdapter\FlatStructure`

2.7.2 Creating an Oracle Enterprise Service Bus Application and Project

To create an application and a project for the use case, follow these steps:

1. In the **Application Navigator** of Oracle JDeveloper, right-click **Applications** and select **New Application**. The Create Application dialog box is displayed.
2. Enter `FileFTP_RW` in the **Application Name** field and click **OK**. The Create Project dialog box is displayed.
3. Click **Cancel**.
4. In the Application window, select **Applications** and then right-click `FileFTP_RW`.
5. Select **New Project**. The New Gallery dialog box is displayed.

6. From Categories, select **General** and then **Projects**.
7. From Items, select **ESB Project** and click **OK**. The Create ESB Project dialog box is displayed.
8. Enter `FileRead_FTPWrite` in the **Project Name** field.
9. Click **OK**.

2.7.3 Importing the Schema Definition (.XSD) Files

Perform the following steps to import the XSD files that will define the structure of the messages:

1. Create a **Schema** folder and copy the `address-csv.xsd` file to this folder.
2. In the **Application Navigator**, select **FileRead_FTPWrite**.
3. From the **File** menu, select **Import**. The Import dialog box is displayed.
4. From the **Select What You Want to Import** list, select **Web Source**, and then click **OK**. The Web Source dialog box is displayed.
5. To the right of the **Copy From** field, click **Browse**. The Choose Directory dialog box is displayed.
6. Navigate to the **Schema** folder and click **Select**. The Web Source dialog box with the directory selected is displayed.
7. Click **OK**.

2.7.4 Creating the Inbound File Adapter Service

Perform the following steps to create an inbound file adapter service to read the file from a local directory

1. Drag a File Adapter service from Components Palette to the design area. The Create File Adapter Service dialog box is displayed.
2. Enter `ReadFile` in the **Name** field.
3. Click the Configure adapter service wsdl icon next to the WSDL field. The Adapter Configuration Wizard Welcome window is displayed.
4. Click **Next**. The Service Name window is displayed.
5. Click **Next**. The Operations window is displayed.
6. Select **Read File** and click **Next**. The File Directories window is displayed
7. Select **Physical Path** option, and click **Browse** and select a polling directory.
8. Click **Next**. The File Filtering window is displayed.
9. Enter `*.txt` in the **Include Files with Name Pattern** field and click **Next**. The File Polling window is displayed.
10. Click **Next**. The Messages window is displayed.
11. Click **Browse**. The Type Chooser dialog box is displayed.
12. Select **Project Schema Files**, `address-csv.xsd`, and then **Root-Element**.
13. Click **OK**.
14. Click **Next** in the Messages window.
15. Click **Finish**. The Create File Adapter Service dialog box is displayed.

16. Click **OK**. A routing service `ReadFile_RS` will be created along with the `ReadFile` adapter service.

2.7.5 Creating the Outbound FTP Adapter Service

Perform the following steps to create an outbound FTP adapter service which will write the file to an FTP server:

1. Drag an FTP Adapter service from Components Palette to the design area. The Create FTP Adapter Service dialog box is displayed.
2. Enter `WriteFTP` in the **Name** field.
3. Click the Configure adapter service wsdl icon next to the WSDL field. The Adapter Configuration Wizard Welcome window is displayed.
4. Click **Next**. The Service Name window is displayed.
5. Click **Next**. The FTP Server Connection window is displayed.
6. Specify the JNDI Name of the FTP Server in the **FTP Server JNDI Name** field and click **Next**. The Operations window is displayed.
7. Select **Ascii** option as File Type.
8. Select **Put File** option as the Operation Type and click **Next**. The File Configuration window is displayed.
9. Specify the directory to which file will be written in the **Directory for Outgoing Files(physical path)** field.
10. Specify the naming convention for the output file name in the File Naming Convention field. For example, `po_%SEQ%.txt`.
11. Click **Next**. The Messages window is displayed.
12. Click **Browse**. The Type Chooser dialog box is displayed.
13. Select **Project Schema Files, address-csv.xsd**, and then **Root-Element**.
14. Click **OK**.
15. Click **Next** in the Messages window.
16. Click **Finish**.

2.7.6 Creating the Routing Rule

Perform the following steps to create a routing service:

1. Double-click the **ReadFile_RS** routing service.
2. Click the Plus icon in the Routing rules area. The Browse Target Service Operation dialog box is displayed.
3. Select **ESB, Services in project, DefaultSystem, WriteFTP**, and then **Put**.
4. Click **OK**.
5. Click the icon next to the **<<Transformation Map>>** box. The request Transformation Map dialog box is displayed.
6. Select **Create New Mapper File** and click **OK**.

A `Root_To_Root.xsl` tab is added to Oracle JDeveloper. This tab enables you to graphically create a document transformation file to convert the structure of the file data to a canonical data structure.

7. Drag and drop the **imp1:Address** source element to the **imp1:Address** target element. The Auto Map Preferences dialog box is displayed.
8. From the **During Auto Map** options, deselect **Match Elements Considering Their Ancestor Names**.
9. Click **OK**.
10. From the **File** menu, click **Save**.

2.7.7 Registering Services with Oracle Enterprise Service Bus

Perform the following steps to register the services with Oracle Enterprise Service Bus Server:

1. In the **Applications Navigator**, right-click `FileRead_FTPWrite`, choose **Register with ESB**, and then click *LocalIntegrationServer*.

A message is displayed to indicate that the following services were successfully registered:

- *System.WriteFTP* Created
- *System.ReadFile_RS* Created
- *System.ReadFile* Created

2. Click **OK**.
3. View the Oracle Enterprise Service Bus configuration in the Oracle Enterprise Service Bus Control Console, as follows:
 - If the Oracle Enterprise Service Bus Control Console is currently open, then:
Click the Refresh button, and then click **ReadFile_RS**.
 - If the Oracle Enterprise Service Bus Control Console is not open, then:
Select **Start, All Programs, Oracle - Oracle_Home, Oracle ESB, ESB Control**, to open the Oracle Enterprise Service Bus Control Console.

2.7.8 Run-Time Task

At run time, copy a text file to the polling directory. Once the file adapter picks the file, the file will be written to the directory that you specified at design time.

Oracle Application Server Adapter for Advanced Queuing

This chapter describes how to use the Oracle Application Server Adapter for Advanced Queuing (AQ adapter), which enables a BPEL process or an ESB service to interact with a queue.

This chapter contains the following topics:

- [Section 3.1, "Introduction to the AQ Adapter"](#)
- [Section 3.2, "Use Cases for the AQ Adapter"](#)
- [Section 3.3, "AQ Adapter Use Cases for Oracle BPEL Process Manager"](#)
- [Section 3.4, "AQ Adapter Use Cases for Oracle Enterprise Service Bus"](#)

3.1 Introduction to the AQ Adapter

Oracle Streams Advanced Queuing (AQ) provides a flexible mechanism for bidirectional, asynchronous communication between participating applications. Because advanced queues are an Oracle database feature, they are scalable and reliable. Backup and recovery (including any-point-in-time recovery), logging, transactional services, and system management are all inherent features of the database and, therefore, advanced queues. Multiple queues can also service a single application, partitioning messages in a variety of ways and providing another level of scalability through load balancing. See *Oracle Streams Advanced Queuing User's Guide and Reference* for more information.

This section comprises the following topics:

- [AQ Adapter Features](#)
- [AQ Adapter Integration with Oracle BPEL Process Manager](#)
- [AQ Adapter Integration with Oracle Enterprise Service Bus](#)

3.1.1 AQ Adapter Features

The AQ adapter is both a producer and consumer of AQ messages. The enqueue operation is exposed as a JCA outbound interaction. The dequeue operation is exposed as a JCA inbound interaction.

The AQ adapter supports both ADT (Oracle object type) and RAW queues as payloads. It also supports extracting a payload from one ADT member column. The AQ adapter does not support the AQ XML type.

The AQ adapter supports headers for enqueue and dequeue options. Headers are comprised of an AQ header and a payload header. The AQ header consists of the standard message properties that are present in every queue.

You access the AQ adapter from the Adapter Configuration Wizard, which you use to browse queues and expose the underlying metadata as a WSDL with JCA extensions.

For AQ adapter samples, go to

`Oracle_Home\bpel\samples\tutorials\124.AQAdapter`

3.1.1.1 Enqueue-Specific Features (Message Production)

The AQ adapter supports the following features of Oracle Streams AQ:

- Correlation identifier

In the Adapter Configuration Wizard, you can specify a correlation identifier when defining an enqueue operation, which you use to retrieve specific messages.

- Multiconsumer queue

In Oracle Streams AQ, more than one consumer can process and consume a single message. To use this feature, you must create multiconsumer queues and enqueue the messages into these queues. In this configuration, a single message can be consumed by more than one AQ consumer (dequeue operation), either through the default subscription list or with an override recipient list. Under this scenario, a message remains in the queue until it is consumed by all of its intended consumer agents. The AQ adapter enqueue header enables you to specify the override recipient list (string values separated by commas) that can retrieve messages from a queue. All consumers that are added as subscribers to a multiconsumer queue must have unique values for the `Recipient` parameter. This means that two subscribers cannot have the same values for the `NAME`, `ADDRESS`, and `PROTOCOL` attributes.

- Message priority

If you specify the priority of enqueued messages, then the messages are dequeued in priority order. If two messages have the same priority, the order in which they are dequeued is determined by the enqueue time. You can also create a first-in, first-out (FIFO) priority queue by specifying the enqueue time priority as the sort order of the messages. This priority is a property of the AQ adapter enqueue header. The enqueue time is set automatically by the underlying AQ application.

Here is an example of how to create the FIFO queue:

```
EXECUTE DBMS_AQADM.CREATE_QUEUE_TABLE( \
queue_table => 'OE_orders_pr_mqtab', \
sort_list => 'priority,enq_time', \
comment => 'Order Entry Priority \
MultiConsumer Orders queue table', \
multiple_consumers => TRUE, \
queue_payload_type => 'BOLADM.order_typ', \
compatible => '8.1', \
primary_instance => 2, \
secondary_instance => 1);
EXECUTE DBMS_AQADM.CREATE_QUEUE ( \
queue_name => 'OE_bookedorders_que', \
queue_table => 'OE_orders_pr_mqtab');
```

- Time specification and scheduling

In Oracle Streams AQ, you can specify a delay interval and an expiration interval. The delay interval determines when an enqueued message is marked as available to the dequeuers after the message is enqueued. When a message is enqueued with a delay time set, the message is marked in a `WAIT` state. Messages in a `WAIT` state are masked from the default dequeue calls. The expiration time property is used to specify an expiration time and the message is automatically moved to an exception queue if the message is not consumed before its expiration. The delay interval, expiration time, and exception queue property are properties of the AQ adapter enqueue header.

3.1.1.2 Dequeue and Enqueue Features

Oracle Streams AQ provides the following dequeuing options:

- Poll option
- Notification option

The poll option involves processing the messages as they arrive and polling repeatedly for messages. The AQ adapter supports a polling mechanism for consuming AQ messages.

The AQ adapter supports the following features of Oracle Streams AQ:

- Multiconsumer queue

The AQ adapter can retrieve messages from a multiconsumer queue.

- Navigation of messages for dequeuing

Messages do not have to be dequeued in the same order in which they were enqueued. You can use a correlation identifier to specify dequeue order. The Adapter Configuration Wizard defines the correlation ID for the dequeue operation.

- Retries with delays

The number of retries is a property of the AQ adapter dequeue header. If the number of retries exceeds the limit, the message is moved to an exception queue that you specify. The exception queue is a property of the AQ adapter enqueue header.

- Rule-based subscription

Oracle Streams AQ provides content-based message filtering and subject-based message filtering. A rule defines one or more consumers interest in subscribing to messages that conform to that rule. For a subject-based rule, you specify a Boolean expression using syntax similar to the `WHERE` clause of a SQL query. This Boolean expression can include conditions on message properties (currently priority and correlation ID), user data properties (object payloads only), and functions (as specified in the `WHERE` clause of a SQL query).

- AQ headers

[Table 3–1](#) summarizes the AQ header properties for the enqueue and dequeue operations.

Table 3–1 AQ Header Message Properties

Message Property	Datatype	Default If Not Specified	Description	Include in Message Header for Enqueue	Include in Message Header for Dequeue
Priority	Integer	1	Priority of the message. A smaller number indicates a higher priority.	Yes	Yes
Delay	Integer	0	The number of seconds after which the message is available for dequeuing	Yes	No
Expiration	Integer	-1 (never expires)	The number of seconds before the message expires. This parameter is an offset from the delay.	Yes	No
Correlation	String	-	User-assigned correlation ID	Yes	Yes
RecipientList	String	-	The list of recipients for this message, separated by commas. This overrides RecipientList in the InteractionSpec	Yes	No
ExceptionQueue	String	-	The exception queue name	Yes	No
EnqueueTime	String	-	The time at which the message was enqueued	No	Yes
MessageID	String	-	The hexadecimal representation of the message ID	No	Yes
OrigMessageId	String	-	The hexadecimal representation of the original message ID	No	Yes
Attempts	Integer	-	The number of failed attempts at dequeuing the message	No	Yes

See Oracle Application Server Adapter Concepts for information on AQ adapter architecture, adapter integration with Oracle BPEL Process Manager, and adapter deployments.

3.1.1.3 Supported ADT Payload Types

The AQ adapter supports the following RAW types:

- BLOB
- CHAR
- CLOB
- DATE
- DECIMAL
- DOUBLE PRECISION
- FLOAT
- INTEGER
- NUMBER
- REAL
- SMALLINT

- `TIMESTAMP`
- `VARCHAR2`

In addition, the primitive types, varrays of objects, or primitives are also supported.

Note: The AQ Adapter does not currently support the following data types for ADT columns: `TIMESTAMP WITH LOCAL TIMEZONE` and `TIMESTAMP WITH TIMEZONE`.

If choosing a payload field instead of the whole ADT, choose only one of the following datatypes as the payload field:

- `CLOB`, either XSD or opaque schema
- `VARCHAR2`, either XSD or opaque schema
- `BLOB`, opaque schema only

3.1.1.4 Native Format Builder Wizard

JDeveloper BPEL Designer provides the Native Format Builder Wizard to define XSD files of various formats, including for the AQ RAW payload. See [Chapter 7, "Native Format Builder Wizard"](#) for more information. For Native Format Builder examples, go to

```
Oracle_Home\bpel\samples\tutorials\124.AQAdapter\
RawQueuePayloadUsingNativeFormat
```

Payload Schema

The payload schemas depend on the payload type. In the whole ADT case, the schema is completely generated by the Adapter Configuration Wizard. In the ADT with `BLOB` selected as the payload case, an opaque schema must be used, which is defined as:

```
<element name="opaqueElement" type="base64Binary" />
```

In all other cases, you can either provide a schema or use an opaque schema, as shown in [Table 3–2](#).

Table 3–2 Payload Schema

Payload Type	Supported Schema
RAW	User-provided schema or opaque schema
Whole ADT	Must use a schema generated by the Adapter Configuration Wizard, which is based on the queue structure
ADT with <code>VARCHAR2</code> picked as payload	User-provided schema or opaque schema
ADT with <code>CLOB</code> picked as payload user-provided schema or opaque schema	User-provided schema or opaque schema
ADT with <code>BLOB</code> picked as payload opaque schema	Opaque schema

If you do not have an XSD, but the payload data is formatted in some way (for example, in a comma-separated value (CSV) format), then the Native Format Builder Wizard can be used to generate an appropriate XSD. The Adapter Configuration Wizard is integrated with the Native Format Builder Wizard. In the Adapter Configuration Wizard - Messages window, click **Define Schema for Native Format** to access the Native Format Builder Wizard.

3.1.2 AQ Adapter Integration with Oracle BPEL Process Manager

Adapter Framework is used for the bidirectional integration of the J2CA 1.5 resource adapters with BPEL Process Manager. Adapter Framework is based on standards and employs the Web service Invocation Framework (WSIF) technology for exposing the underlying J2CA interactions as Web services.

See *Oracle Application Server Adapter Concepts* for information on AQ adapter architecture, adapter integration with Oracle BPEL Process Manager, and adapter deployments.

3.1.3 AQ Adapter Integration with Oracle Enterprise Service Bus

Oracle Enterprise Service Bus Server supports the Oracle adapters and enables you to define inbound and outbound adapter services for each. An inbound adapter service receives data from an external data source and transforms it into an XML message. An outbound adapter service sends data to a target application by transforming an XML message into the native format of the given adapter.

Using Oracle Enterprise Service Bus Server you can send or receive messages from Oracle Advanced Queuing single or multiconsumer queues.

BPEL pre-dates ESB and most of this guide and the samples implicitly assume use with BPEL. However, the adapters work equally well with either BPEL or ESB. For any mention of BPEL here you may substitute ESB instead.

3.2 Use Cases for the AQ Adapter

The following use cases include a general walkthrough of the Adapter Configuration Wizard, followed by examples of how you modify the general procedure in different situations. Each example shows relevant parts of the generated WSDL file.

This section comprises the following topics:

- [Adapter Configuration Wizard Walkthrough](#)
- [Dequeuing and Enqueuing Object and ADT Payloads](#)
- [Dequeuing One Column of the Object/ADT Payload](#)
- [Processing Large Numbers of Messages](#)
- [Using Correlation ID for Filtering Messages During Dequeue](#)
- [Enqueuing and Dequeuing from Multisubscriber Queues](#)
- [Rule-Based Subscription for Multiconsumer Queues](#)

3.2.1 Adapter Configuration Wizard Walkthrough

The following use cases include a configuring AQ adapter, followed by examples of how you modify the generic procedure in different situations. Each example shows relevant parts of the generated WSDL file.

In this example you will create an adapter service that enqueues messages to the CUSTOMER_IN_QUEUE queue, with a payload that is one field within the CUSTOMER_TYPE object (the customer's e-mail address), and with a user-defined schema.

This section describes the tasks required to configure AQ adapter using the Adapter Configuration Wizard in Oracle JDeveloper.

This section comprises the following:

- [Meeting Prerequisites](#)
- [Creating a New BPEL Project](#)
- [Adding a Partner Link](#)
- [Generated WSDL File](#)

3.2.1.1 Meeting Prerequisites

The following prerequisites have to be met before you create the sample service:

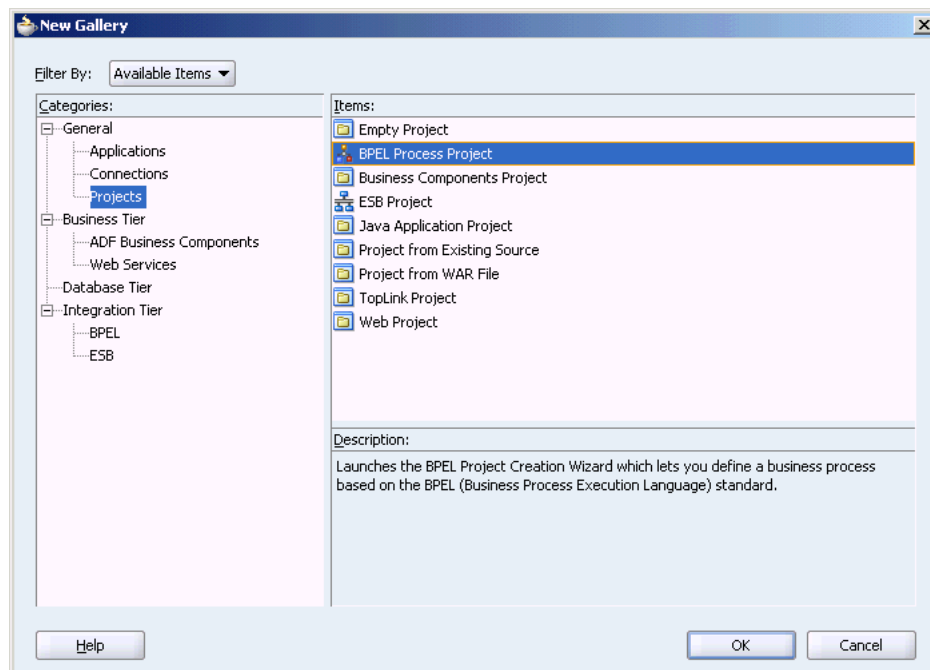
1. Access SCOTT schema in some database.
2. Find data-sources.xml in
C:\product\10.1.3.1\OracleAS_1\bpel\system\appserver\oc4j\j2ee\home\config\data-sources.xml
3. Add the following code to data-sources.xml to define the data source for jdbc/aqSample. Note that URL should refer to the endpoint database that we use.

```
<connection-pool name="aqSample_CONNECTION_POOL">
  <connection-factory factory-class="oracle.jdbc.OracleDriver"
url="jdbc:oracle:thin:scott/tiger@stadd14:1521:db5617"/>
</connection-pool>
<managed-data-source name="AQSamplesDataSource"
connection-pool-name="aqSample_CONNECTION_POOL"
jndi-name="eis/AQ/MyConnection" />
```

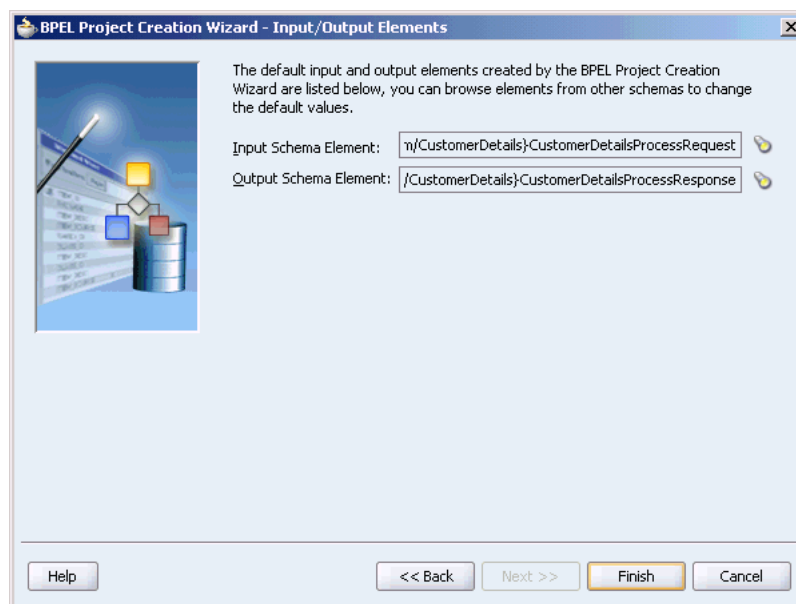
3.2.1.2 Creating a New BPEL Project

The first configuration task is to create a new BPEL project. Use the following steps to create a new BPEL project:

1. Open Oracle JDeveloper
2. From the **File** menu, select **New**. The New Gallery dialog box is displayed.
3. Select **All Technologies** from the **Filter By** box. This displays a list of available categories.
4. Expand the **General** node, and then select **Projects**.
5. Select **BPEL Process Project** from the **Items** group, as shown in [Figure 3–1](#).

Figure 3–1 Creating a New BPEL Process Project

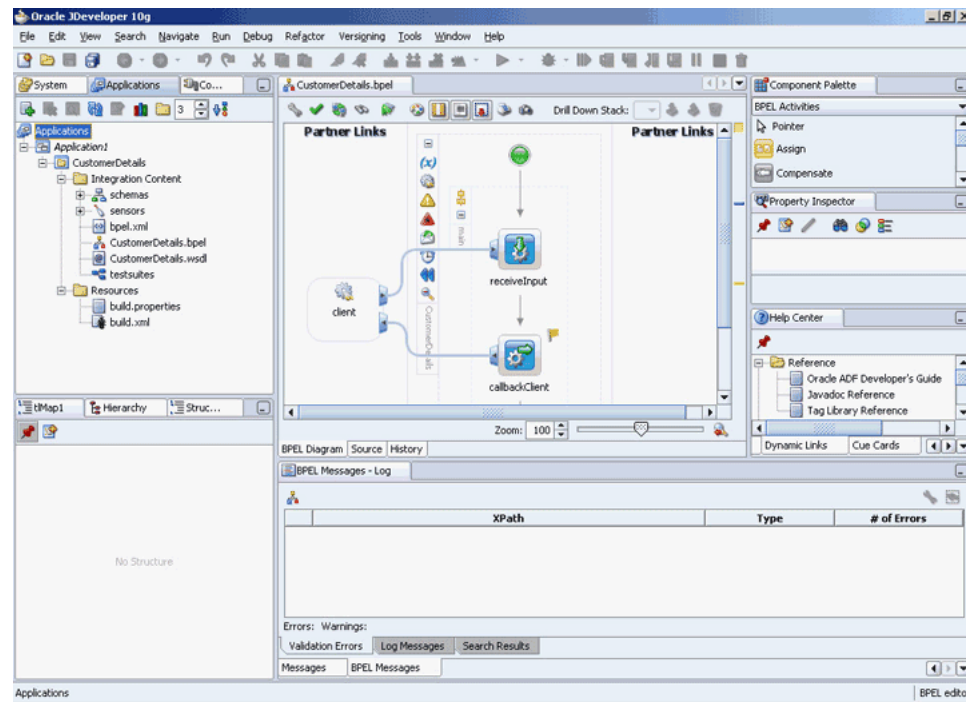
6. Click **OK**. The BPEL Process Project dialog box is displayed.
7. In the **BPEL Process Name** field, enter a descriptive name. For example, CustomerDetails.
8. From the **Template** box, select **Asynchronous BPEL Process**.
The Input/Output Elements dialog box is displayed, as shown in [Figure 3–2](#).

Figure 3–2 The Input/Output Elements Dialog Box

9. Retain the default values for the schemas, and then click **Finish**.

10. A new BPEL process, with the required source files including `bpel.xml`, `CustomerDetails.bpel`, and `CustomerDetails.wsdl` is created, as shown in Figure 3–3.

Figure 3–3 New BPEL Process Project

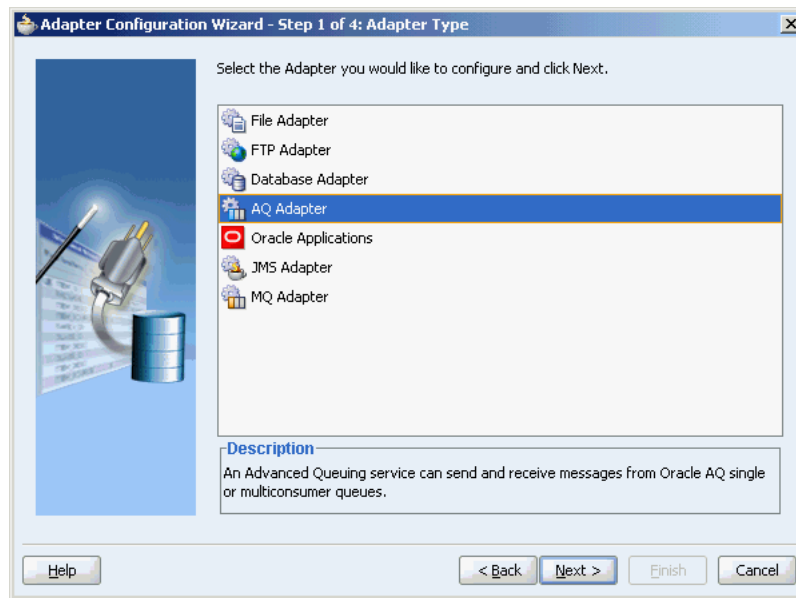


3.2.1.3 Adding a Partner Link

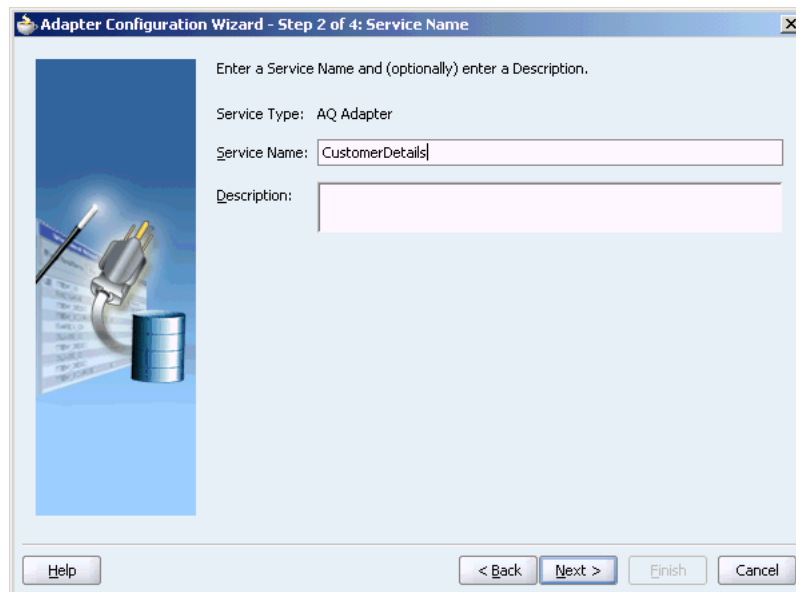
The next task is to add a partner link to the BPEL process. A partner link defines the link name, type, and the role of the BPEL process that interacts with the partner service.

Use the following steps to add a partner link:

1. Select **BPEL Activities** from the Component Palette, and then drag and drop **PartnerLink** into the border area of the process diagram. The Create Partner Link dialog box is displayed.
2. Click the **Define Adapter Service** icon (third icon) in WSDL Settings. The Adapter Configuration Wizard is displayed.
3. Click **Next**. The Adapter Type dialog box is displayed.
4. Select **AQ Adapter**, as shown in Figure 3–4, and then click **Next**.

Figure 3–4 Selecting OracleAS Adapter for Oracle Applications

5. The Service Name dialog box is displayed, as shown in [Figure 3–5](#). Enter the following information:
 - a. In the **Service Name** field, enter a service name.
 - b. In the **Description** field, enter a description for the service. This is an optional field. Click **Next**.

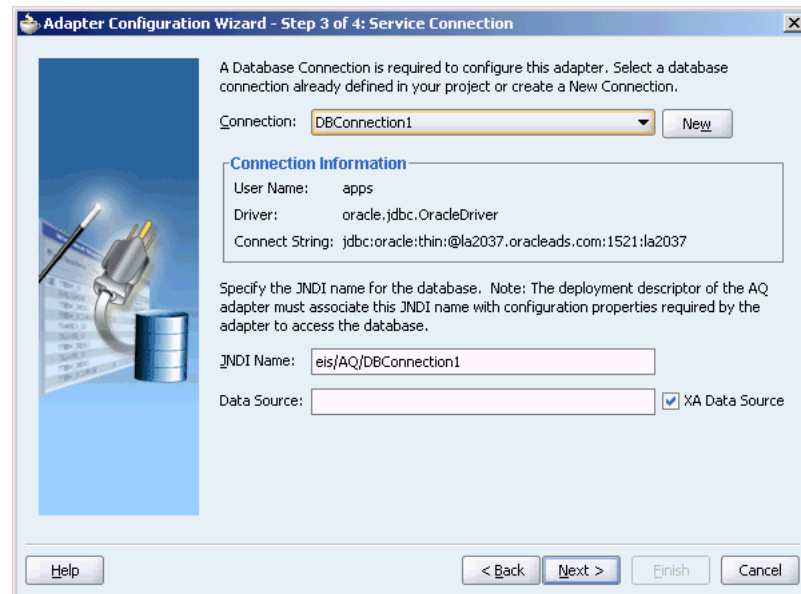
Figure 3–5 Specifying the Service Name

6. The Service Connection dialog box is displayed. Enter the Java Naming and Directory Interface (JNDI) name in the **Database Server JNDI Name** field. The JNDI name acts as a placeholder for the connection used when your service is deployed to the BPEL server. This enables you to use different databases for development and later production.

See Also: *Oracle Application Server Adapter Concepts Guide* for understanding JNDI concepts

Figure 3–6 shows how to create a new database connection.

Figure 3–6 Creating a New Database Connection

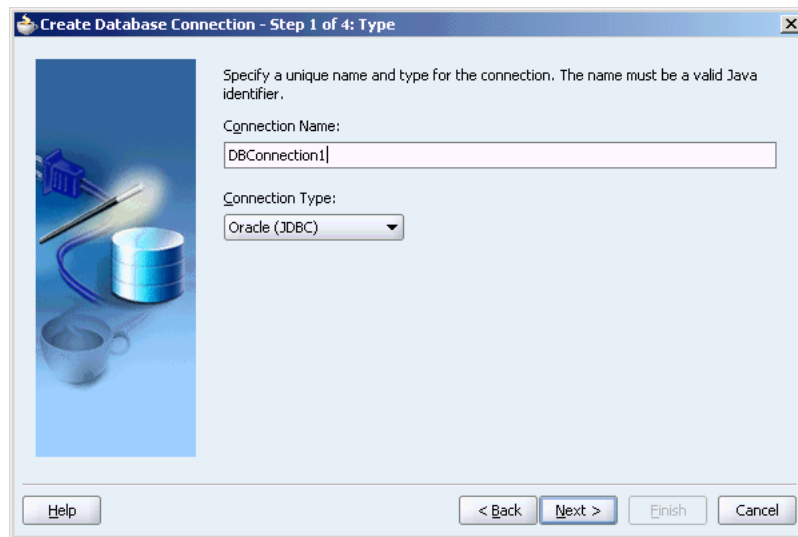


7. Click **New** to define a database connection. The Create Database Connection Wizard is displayed.

Note: You must connect to the database where Oracle Applications is running.

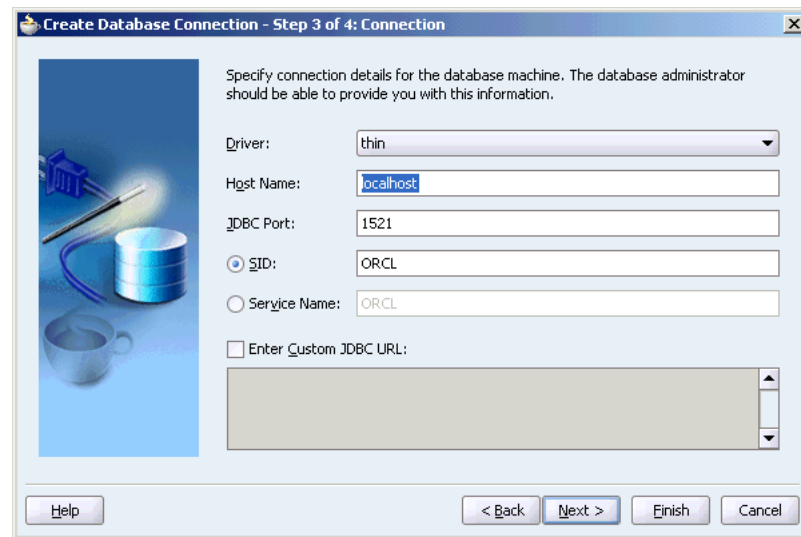
8. Enter the following information in the Type dialog box:
 - a. In the **Connection Name** field, specify a unique name for the database connection. In this example, type **DBConnection1**.
 - b. From the **Connection Type** box, select **Oracle (JDBC)**.

Figure 3–7 shows the Type dialog box.

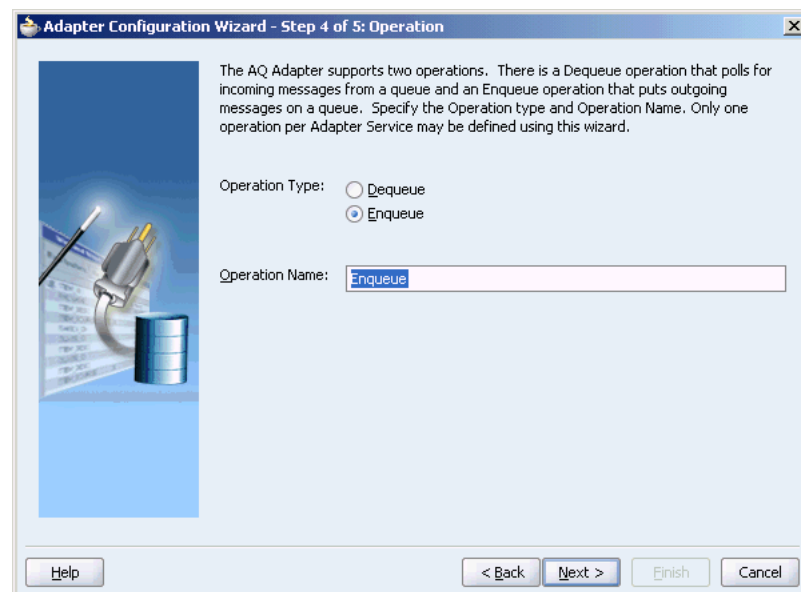
Figure 3–7 Specifying the Connection Name and Type of Connection

9. Click **Next**. The Authentication dialog box is displayed.
10. Enter information in the following fields:
 - a. In the **UserName** field, specify a unique name for the database connection. In this example, type **scott**.
 - b. In the **Password** field, specify a password for the database connection. In this example, type **tiger**.
 - c. Leave the **Role** field blank.
 - d. Select **Deploy Password**.
11. Click **Next**. The Connection dialog box is displayed.
12. Enter information in the following fields:
 - a. From the **Driver** list, select **Thin**.
 - b. In the **Host Name** field, retain the default value, **localhost**.
 - c. In the **JDBC Port** field, specify the port number for the database connection. In this example, type **1521**.
 - d. In the **SID** field, specify a unique SID value for the database connection. In this example, type **ORCL**.

Figure 3–8 shows the Connection dialog box.

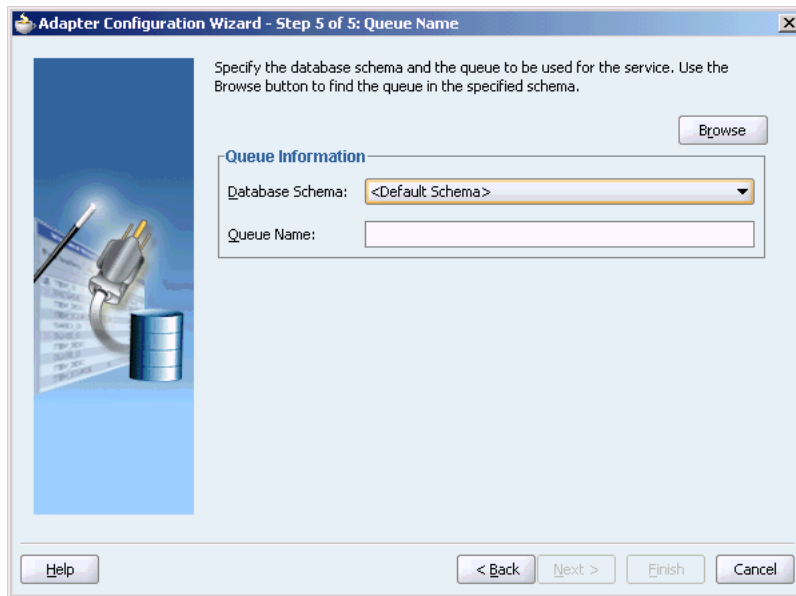
Figure 3–8 Specifying the New Database Connection Information

13. Click **Next**. The Test dialog box is displayed.
14. Click **Test Connection** to determine whether the specified information establishes a connection with the database.
15. Click **Next**. The Service Connection dialog box is displayed, providing a summary of the database connection.
16. Click **Finish** to complete the process of creating a new database connection.
17. Click **Next** in the Service Connection dialog box.
The Operation dialog box is displayed.
18. Select **Enqueue**, to send messages to a queue, or **Dequeue**, to receive messages from a queue. In this example, select **Enqueue**, as show in [Figure 3–9](#), and then click **Next**.

Figure 3–9 The Operation Dialog Box

The Queue Name dialog box is displayed, as shown in [Figure 3–10](#).

Figure 3–10 The Queue Name Dialog Box



19. Select a database schema from the drop-down list, or click **Browse** to browse for the schema. In this example, click **Browse**.

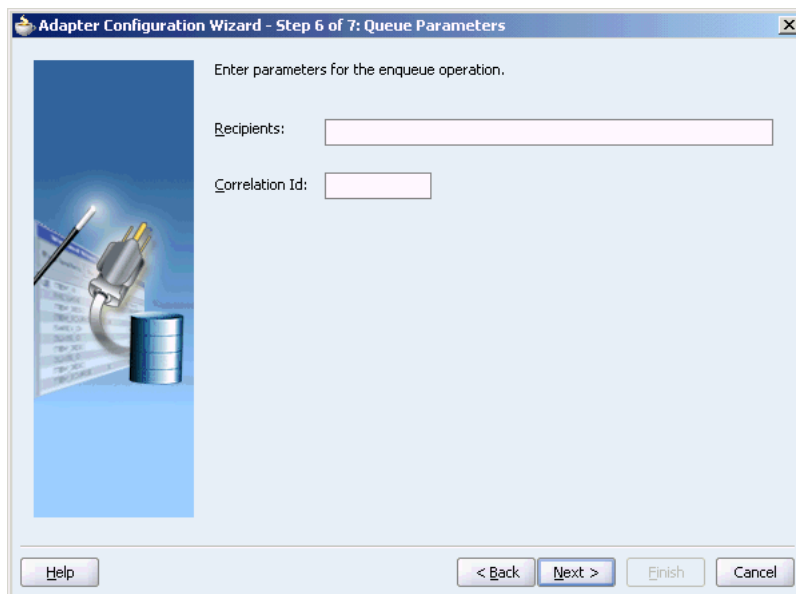
The Select Queue dialog box is displayed.

20. Select the required queue, and then click **Ok**. In this example, click **Customer_In_Queue**.

21. Once you have selected the database schema, click **Next**.

The Queue Parameters dialog box is displayed, as shown in [Figure 3–11](#).

Figure 3–11 The Queue Parameters Dialog Box



22. Enter values for the parameters, and then click **Next**.

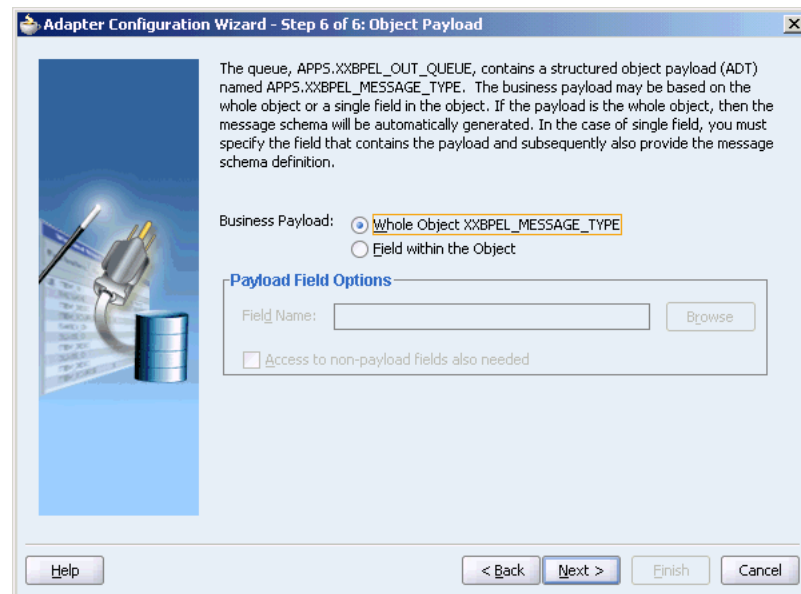
- **Recipients:** Enter the consumer name or names separated by commas that are the intended recipients for the messages enqueued by the adapter. The message remains in the queue until all recipients have dequeued the message. If the **Recipients** field is left empty, then all currently active consumers are recipients. This field can be overridden on a per message basis by setting the `RecipientList` field described in [Table 3-1](#) in the outbound header.
- **Correlation ID:** Enter an optional correlation ID from 1 to 30 characters in length. This is used to identify messages that can be retrieved at a later time by a dequeue activity using the same correlation ID.

The value to enter is agreed upon between the enqueueing sender and the dequeuing receiver for asynchronous conversations. The correlation ID maps to an AQ header. Correlation IDs in the inbound direction enable you to be selective about the message to dequeue. This field is optional. If you do not enter a value, all messages in the queue are processed.

If you enter a value for the Correlation ID in the outbound direction, all outbound messages have the correct ID set to the value entered. You can override this value on a per message basis in the correlation field of the outbound header.

The Object Payload dialog box is displayed, as shown in [Figure 3-12](#).

Figure 3-12 The Object Payload Dialog Box



23. Select the business payload: either the entire object, or just one field within the object. In this example, select **Field within the Object**.

- If you select **Whole Object CUSTOMER_TYPE**, click **Next** and go to Step 26. The message schema is automatically generated.
- If you select **Field within the Object**, the business payload is contained in a single field in the object. Specify the correct field name, either by entering the name or by browsing for it using the **Browse** button. The field containing the payload you select must be a CLOB, BLOB, or VARCHAR2.

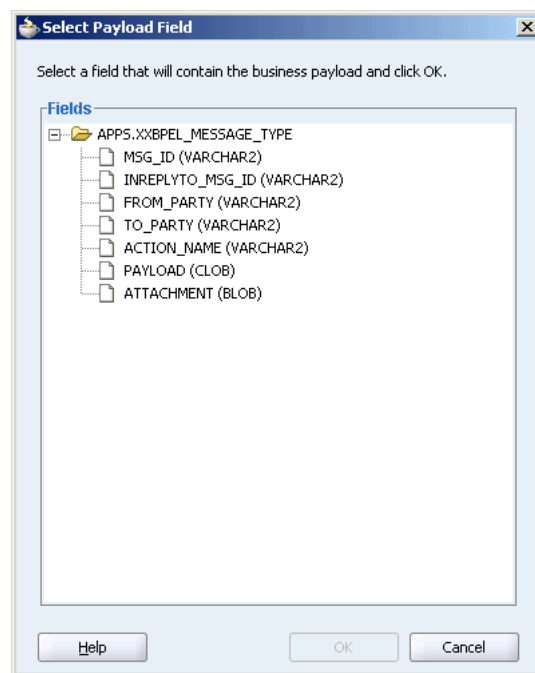
The **Access to non-payload fields also needed** check box is available as an option. Select this check box if you need to specify additional information in a header field that is separate from the object payload. This selection generates the additional header schema but does not automatically create the header variable. Header variable is an optional field used to pass the header to and from the partnerlink. This header variable is created in the adapter tab of corresponding receive or invoke activity. See section [Header Variables in JDeveloper BPEL Designer](#) for more information.

For example, your payload may be a JPG image. You may want to specify a person's name in the nonpayload field. This selection generates an additional header schema file (*object_name.xsd*, where *object_name* is the structured payload object used by the queue). Using this information is discussed in more detail in [Rule-Based Subscription for Multiconsumer Queues](#).

24. Under **Payload Field Options**, enter a file name or click **Browse**. In this example, click **Browse**.

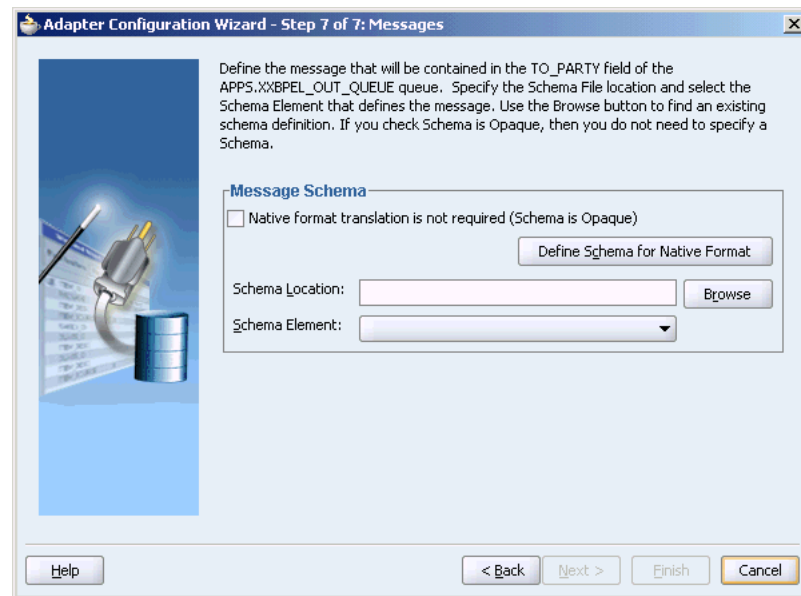
The Select Payload Field dialog box is displayed, as shown in [Figure 3–13](#).

Figure 3–13 The Select Payload Field Dialog Box



25. Select **TO_PARTY** from the list of payload fields, and then click **OK**.
26. Click **Next**.

The Messages dialog box is displayed, as shown in [Figure 3–14](#).

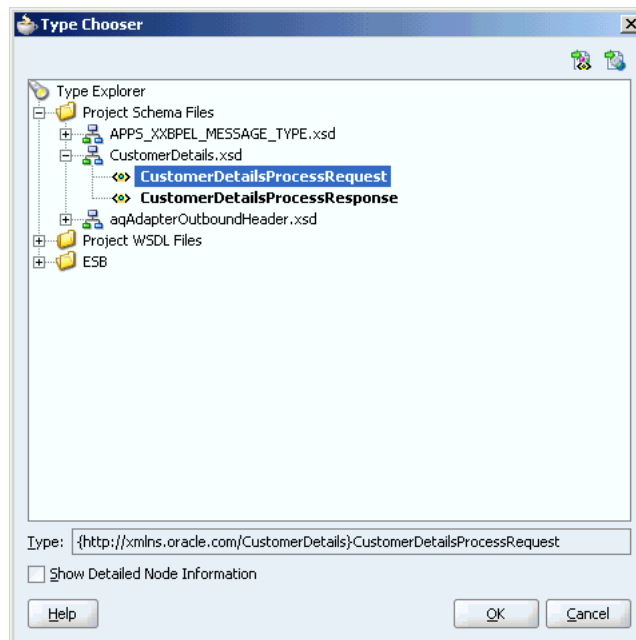
Figure 3–14 The Messages Dialog Box

The Message dialog box has the following options:

- **Native format translation is not required (Schema is Opaque):** Select this option if you do not want to specify a schema. Selecting this option disables all the other fields under Message Schema.
- **Define Schema for Native Format:** Click this to start the Native Format Builder wizard, which guides you through the process of defining the native format.
- **Schema Location:** You can enter the path for the schema file URL or click **Browse** to browse for the path.

27. In this example, click **Browse** to browse for the schema file URL.

The Type Chooser dialog box is displayed, as shown in [Figure 3–15](#).

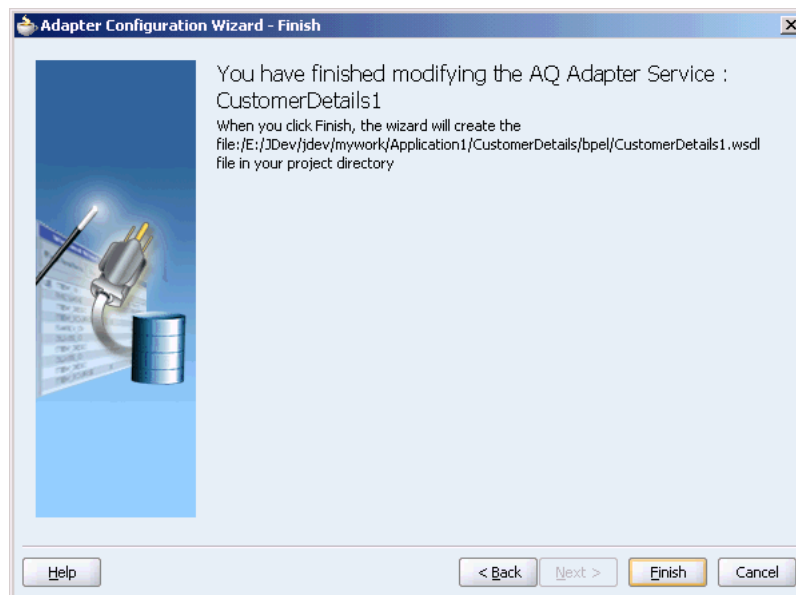
Figure 3–15 The Type Chooser Dialog Box

28. Select **BPELProcessProcessRequest** from the list, and then click **OK**.

The Messages window reappears, with the Schema Location and Schema Element fields completed.

29. Click **Next**.

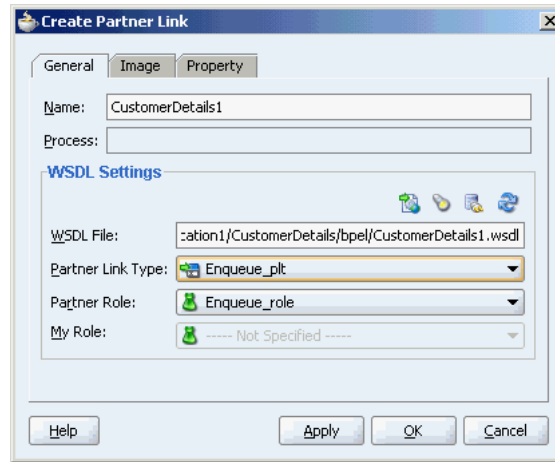
The Finish screen is displayed, as shown in [Figure 3–16](#). This dialog box shows the path and name of the adapter file that the wizard creates.

Figure 3–16 The Finish screen

30. In the Finish window, click **Finish**.

The Create Partner Link dialog box is displayed, with the fields filled up, as shown in [Figure 3–17](#).

Figure 3–17 The Create Partner Link Dialog Box



31. Click OK.

3.2.1.4 Generated WSDL File

The adapter service generates a WSDL file to serve as the defined adapter interface. This section contains explanation about each code segment in the WSDL file created by the preceding example:

This part of the code segment defines the name of the adapter, and the locations of various necessary schemas and other definition files.

```
<definitions
  name="CustomerDetails1"
  targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/aq/CustomerDetails/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:tns="http://xmlns.oracle.com/pcbpel/adapter/aq/CustomerDetails/"
  xmlns:plt="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
  xmlns:jca="http://xmlns.oracle.com/pcbpel/wsdl/jca/"
  xmlns:impl="http://xmlns.oracle.com/BPELProcess"
  xmlns:hdr="http://xmlns.oracle.com/pcbpel/adapter/aq/outbound/"
>
```

The following code segment imports the necessary namespaces:

```
<import namespace="http://xmlns.oracle.com/pcbpel/adapter/aq/outbound/"
location="aqAdapterOutboundHeader.wsdl"/>
<import namespace="http://xmlns.oracle.com/BPELProcess"
location="bpelprocess.wsdl"/>
```

The following code segment defines the message name and port type for the partner link:

```
<message name="BPELProcessProcessRequest_msg">
  <part name="BPELProcessProcessRequest"
element="impl:BPELProcessProcessRequest"/>
</message>
<portType name="Enqueue_ptt">
  <operation name="Enqueue">
    <input message="tns:BPELProcessProcessRequest_msg"/>
```

```
    </operation>
  </portType>
```

The following code segment defines the necessary bindings for the enqueue operation and the target queue, and identifies the message header:

```
    <binding name="Enqueue_binding" type="tns:Enqueue_ptt">
      <jca:binding />
      <operation name="Enqueue">
        <jca:operation
          InteractionSpec="oracle.tip.adapter.aq.outbound.AQEnqueueInteractionSpec"
            QueueName="CUSTOMER_IN_QUEUE"
            ObjectFieldName="CONTACTS.EMAIL"
            OpaqueSchema="false" />
        </jca:operation>
      <input>
        <jca:header message="hdr:Header" part="Header"/>
      </input>
    </operation>
  </binding>
</service name="AQ_Example">
  <port name="Enqueue_pt" binding="tns:Enqueue_binding">
```

This part defines the database connection, the connection factory (as defined in the `oc4j-ra.xml` file), and the name and role of the `partnerLinkType` and `portType`.

```
<!--Your runtime connection is declared in
J2EE_HOME/application-deployments/default/AqAdapter/oc4j-ra.xml
These mcf properties here are from your design time connection and
save you from having to edit that file and restart the application server
if eis/AQ/DBConnection1 is missing.
These mcf properties are safe to remove.-->
  <jca:address location="eis/AQ/DBConnection1"
    UIConnectionName="DBConnection1"

    ManagedConnectionFactory="oracle.tip.adapter.aq.AQManagedConnectionFactory"

    mcf.ConnectionString="jdbc:oracle:thin:@144.25.143.7:1521:shashipc"
      mcf.UserName="apps"
      mcf.Password="53CB0F044A0D3DD2C063679F18F89870" />
  </port>
</service>
<plt:partnerLinkType name="Enqueue_plt" >
  <plt:role name="Enqueue_role" >
    <plt:portType name="tns:Enqueue_ptt" />
  </plt:role>
</plt:partnerLinkType>
</definitions>
```

3.2.2 Dequeuing and Enqueuing Object and ADT Payloads

The Adapter Configuration Wizard walkthrough shows how to enqueue one field in an object. To enqueue or dequeue the entire object as the payload, do the following:

- Select **Enqueue** or **Dequeue** in Step 18.
- Select **Whole Object CUSTOMER_TYPE** in Step 23, and skip to Step 29.

The Adapter Configuration Wizard walkthrough provided an enqueue operation example. For a dequeue operation, the resulting WSDL file differs by including dequeue information instead of enqueue information, and by lacking an `ObjectFieldName` in the `jca:Operation` section of the code. See the following code sample for a dequeue operation and compare it to the WSDL file from the general walkthrough to see these differences:

```
<portType name="Dequeue_ptt">
  <operation name="Dequeue">
    <input message="tns:CUSTOMER_TYPE_msg" />
  </operation>
</portType>
<binding name="Dequeue_binding" type="tns:Dequeue_ptt">
<pc:inbound_binding />
  <operation name="Dequeue">
    <jca:operation
      ActivationSpec="oracle.tip.adapter.aq.inbound.AQDequeueActivationSpec"
      QueueName="CUSTOMER_OUT_QUEUE"
      OpaqueSchema="false" >
    </jca:operation>
    <input>
      <jca:header message="hdr:Header" part="Header" />
    </input>
  </operation>
```

3.2.3 Dequeuing One Column of the Object/ADT Payload

The walkthrough is an example of enqueueing one field or column within an object payload. To create an adapter that dequeues the one field in an object, the steps are the same, except that in Step 18, select **Dequeue**.

The following segment of the generated WSDL file specifies that one field, in this case `CONTACTS.EMAIL`, is dequeued.

```
<jca:operation
  ActivationSpec="oracle.tip.adapter.aq.inbound.AQDequeueActivationSpec"
  QueueName="CUSTOMER_IN_QUEUE"
  ObjectFieldName="CONTACTS.EMAIL"
  OpaqueSchema="true" >
</jca:operation>
```

3.2.4 Processing Large Numbers of Messages

If you want to process large numbers of messages with the AQ adapter, ensure that you set the `cacheWSIFOperation` property to `true` in the `bpel.xml` file:

```
<BPELSuitcase>
  <BPELProcess id="HelloWorld" src="HelloWorld.bpel">
    <partnerLinkBindings>
      <partnerLinkBinding name="Dequeue">
        <property name="wsdlLocation">fileService.wsdl</property>
      </partnerLinkBinding>
      <partnerLinkBinding name="Enqueue">
        <property name="wsdlLocation">fileWriteService.wsdl</property>
        *<property name="cacheWSIFOperation">true</property>*
      </partnerLinkBinding>
    </partnerLinkBindings>
```

Set this property through the **Property** tab of the Create Partner Link or Edit Partner Link window for the partner link.

3.2.5 Using Correlation ID for Filtering Messages During Dequeue

Perform the following steps to set up an adapter that dequeues messages with a certain correlation ID only.

- Select **Dequeue** operation in Step 18.
- Enter the correlation ID in Step 22.

The adapter dequeues messages enqueued with that same correlation ID only.

The resulting WSDL file contains the correlation ID:

```
<jca:operation
  ActivationSpec="oracle.tip.adapter.aq.inbound.AQDequeueActivationSpec"
  QueueName="CUSTOMER_OUT_QUEUE"
  Correlation="147"
  OpaqueSchema="false" >
</jca:operation>
```

3.2.6 Enqueuing and Dequeuing from Multisubscriber Queues

Multisubscriber queues are accessible by multiple users, and sometimes those users are only concerned with certain types of messages within the queue. For example, you may have a multiuser queue for loan applications where loans below \$100,000 can be approved by regular loan-approval staff, whereas loans over \$100,000 must be approved by a supervisor. In this case, the BPEL process can use one adapter to enqueue loan applications for big loans for supervisors, and another adapter to enqueue loan applications for smaller loans for regular staff in the same multisubscriber queue.

Specify an adapter that enqueues to a multisubscriber queue, and include queue parameters in the **Recipients** field and the **Correlation ID** field.

In step 22, specify **Bob** in the Recipients field.

The following code is from a WSDL file generated by defining an AQ adapter that enqueues with a recipient list of Bob:

```
<jca:operation
  InteractionSpec="oracle.tip.adapter.aq.outbound.AQEnqueueInteractionSpec"
  QueueName="IP_IN_QUEUE"
  DatabaseSchema="SCOTT"
  ObjectFieldName="PAYLOAD"
  PayloadHeaderRequired="true"
  RecipientList="Bob"
  OpaqueSchema="true" >
</jca:operation>
```

When dequeuing from a multisubscriber queue, the Queue Parameters window is displayed.

The **Consumer** field is where you specify the consumer name, or the name of the queue subscriber. This must match the **Recipient** entry on the enqueue process for the message to be dequeued. When subscribing to a multiconsumer queue, this field is required.

The following code is from a WSDL file generated by defining an AQ adapter with a consumer name:

```
<jca:operation
  ActivationSpec="oracle.tip.adapter.aq.inbound.AQDequeueActivationSpec"
  QueueName="IP_IN_QUEUE"
```

```

        DatabaseSchema="SCOTT"
        ObjectFieldName="PAYLOAD"
        PayloadHeaderRequired="true"
        Consumer="Bob"
        OpaqueSchema="true" >
    </jca:operation>
    <input>
        <jca:header message="tns:Header_msg" part="Header" />
    </input>
</operation>

```

The **Message Selector Rule** field enables you to enter rules for accepting messages. This is discussed in more detail in the following section.

Correlation Id is a number assigned to a message to identify it to specific dequeuers, as mentioned in Step 22. This differs from a subscription rule in that the queue stores messages with a correlation ID for later use when a subscriber using that correlation ID comes online.

3.2.7 Rule-Based Subscription for Multiconsumer Queues

When a dequeue is performed from a multisubscriber queue, it is sometimes necessary to screen the messages and accept only those that meet certain conditions. This condition may concern header information, as in selecting messages of only priority 1, or some aspect of the message payload, as in selecting only loan applications above \$100,000.

The **Message Selector Rule** field appears in Step 22 if the selected queue is multisubscriber. Enter a subscription rule in the form of a Boolean expression using syntax similar to a SQL WHERE clause, such as `priority = 1`, or `TAB.USER_DATA.amount > 1000`. The adapter only dequeues messages for which this Boolean expression is true.

In Step 23, **Access to non-payload fields also needed** must be checked in order to access header information.

When this field is checked, the generated WSDL has additional code in the `type` section:

```

    <complexType name="HeaderCType" >
        <sequence>
            <!-- static header -->
            <element name="QueueHeader" type="hdr:HeaderType" />
            <!-- payload header -->
            <element name="PayloadHeader" type="obj1:SERVICE_TYPE" />
        </sequence>
    </complexType>
    <element name="Header" type="tns:HeaderCType" />
<element name="Header" type="tns:HeaderCType" />

and the message:
    <message name="Header_msg">
        <part name="Header" element="tns:Header" />
    </message>

```

Note that `PayloadHeader` is the type for the whole ADT of the queue. The payload contains only the chosen payload field. If **Access to non-payload fields also needed** is checked, the `PayloadHeader` contains the whole ADT (including the payload field, which is also present in the header, but ignored by the adapter).

For working examples of rule-based subscription using header and payload information, go to

`Oracle_Home\bpel\samples\tutorials\124.AQAdapter\
RuleBasedSubscription_Header`

`Oracle_Home\bpel\samples\tutorials\124.AQAdapter\
RuleBasedSubscription_Payload`

3.2.8 General Tips when working with AQ Adapter

This section comprises tips that you can use while working with AQ adapter. It includes the following sections:

- [Tips when working with AQ Adapter](#)
- [Tips when working with JMS Adapter For AQ](#)

3.2.8.1 Tips when working with AQ Adapter

While defining the connection pool for managed data source, set `validate-connection` property to `true` when you use `oracle.jdbc.pool.OracleDataSource` as connection factory in order to have OC4J validate the connection before returning it to the adapters, as shown in the following example:

```
<connection-pool name="AQSAMPLE_POOL1" validate-connection='true'>  
  <connection-factory factory-class="oracle.jdbc.pool.OracleDataSource"  
    user="AQAMAN"  
    password="AQAMAN"  
    url="jdbc:oracle:thin:@bstern-sun.us.oracle.com:1521:iasdb" />  
</connection-pool>  
<managed-data-source name="AQSAMPLE_DS1"  
  connection-pool-name="AQSAMPLE_POOL1"  
  jndi-name="jdbc/aqSample1" tx-level="global"  
  manage-local-transactions="false"  
  login-timeout="0"/>
```

3.2.8.2 Tips when working with JMS Adapter For AQ

When defining an OJMS resource provider, set the following property to `true` - `oracle.jms.useNativeXA=true` at the container level when using the thin driver.

3.3 AQ Adapter Use Cases for Oracle BPEL Process Manager

This section comprises the following topics:

- [Using AQ Headers in a BPEL Process](#)
- [Header Variables in JDeveloper BPEL Designer](#)
- [Configuring a Message Rejection Handler for Data Errors](#)
- [Example connectionString for RAC](#)

3.3.1 Using AQ Headers in a BPEL Process

The entries included in AQ headers are described in ["Dequeue and Enqueue Features"](#). These header entries are available for composing subscription rules.

Header information can also be translated into BPEL variables using the **Adapter** tab when defining BPEL activities that send or receive messages. This tab enables you to create header variables for use with the adapters.

You create header variables in **invoke**, **receive**, **reply**, and **pick - OnMessage** branch activities. Information passing through header variables is protocol specific.

Note that header information for messages coming from the AQ adapter is different from header information for corresponding messages flowing into the AQ adapter. You must use headers if you need to get or set any of these protocol-specific properties. The following examples describe when you must use headers:

- The AQ adapter enables you to get and set the priority of the message. For example, you want two different flows in your business process: one for high priority flows and one for low priority flows. This is possible if you use the inbound AQ header of the message flowing into the business process. In a similar fashion, you can set the priority of an outbound message through the outbound AQ headers.
- In a file propagation scenario, files are being moved from one file system to another using the file adapter. In this case, it is imperative that you maintain file names across the two systems. Use file headers in both directions and set the file name in the outbound file header to use the file name in the inbound file header.

3.3.2 Header Variables in JDeveloper BPEL Designer

The following example describes how to create a special header variable in a **receive** activity in the outbound direction for the file adapter. You are *not* restricted to this direction or adapter type. You can also create this variable in either direction (inbound or outbound) with other adapter types that include headers (AQ, JMS, and FTP).

1. Click the **flashlight** icon to display the Variable Chooser window.
2. Select the second **Variables** folder, and then click **Create** to display the Create Variable window.
3. Enter a unique and recognizable name in the **Name** field (for this example, **Variable_Header**).
4. Select **Message Type**, and then click the **flashlight** icon to display the Type Chooser window.
5. Expand **Message Types**, then **Project WSDL Files**, then **service_name.wsdl**, and then **Message Types**, where **service_name** is the name you specified for the service name when you ran the Adapter Configuration Wizard.

A header message named **OutboundHeader_msg** (or a similar name that includes **Header_msg** as part of the name) is displayed.

6. If this name appears here, select it and go to Step 8.
7. If this name does not appear, perform the following additional steps.
 - a. Expand **Imported WSDL**, then **fileAdapterOutboundHeader.wsdl** (or a similar name for the direction and adapter type you are using), and then **Message Types**.
 - b. Select **OutboundHeader_msg** and go to Step 8.

Note: The name that displays here for the AQ adapter may only be **Header**. Select this name. This indicates that the header is static, because you did not need to select the **Access to non-payload fields** check box when configuring the AQ adapter.

8. Click **OK** to close the Type Chooser window, Create Variable window, and Variable Chooser window.
9. Complete the setup of the **Receive** activity.
10. Open the source view of the BPEL process file to view the header variable you created:

```
bpelx:headerVariable="Variable_Header" />
```

3.3.3 Configuring a Message Rejection Handler for Data Errors

Rejected messages (that is, messages with data errors) can be directed to a rejected messages queue, and can also be logged in a directory on the system for later review.

An example of a rejection handler can be found at

Oracle_Home\bpel\samples\tutorials\124.AQAdapter\AQMessageRejectionHandler

The `readme.txt` file in this directory describes a procedure for testing the message rejection handler.

The `Dequeue.wsdl` file in this directory includes the following code:

```
<pc:inbound_binding />
  <operation name="Dequeue">
    <jca:operation
      ActivationSpec="oracle.tip.adapter.aq.inbound.AQDequeueActivationSpec"
      QueueName="REJECTION_TEST_IN"
      DatabaseSchema="SCOTT"
      OpaqueSchema="false" >
    </jca:operation>
    <input>
      <jca:header message="hdr:Header" part="Header" />
    </input>
    </operation>
  </binding>
```

This part of the code directs the errored message to the `REJECTION_TEST_IN` queue.

3.3.4 Example connectionString for RAC

Consider a scenario when an RAC environment has two databases, for example, one is `infra1` and the other `infra2`, and BPEL is using both AQ and DB adapters. Here, the `oc4j-ra.xml` file uses a jdbc connect string that includes the `infra1` db but not `infra2`. So, when `infra1` goes down BPEL will not connect to the db even though `infra2` is available.

You must make the following changes in the `data-sources.xml` file to configure BPEL AQ and DB adapters correctly in this environment:

```
jdbc:oracle:thin:@(DESCRIPTION=
  (ADDRESS_LIST=(LOAD_BALANCE=on)
    (ADDRESS=(PROTOCOL=tc) (HOST=hostname1) (PORT=1521))
```

```

        (ADDRESS=(PROTOCOL=tcp) (HOST=hostname2) (PORT=1521))
    )
    (CONNECT_DATA=(SERVICE_NAME=orcl))
)

```

3.4 AQ Adapter Use Cases for Oracle Enterprise Service Bus

In this sample, the business process receives a message from the AQAdapter, copies the payload to an outbound message and invokes the AQAdapter with the outbound message.

The queues involved are ADT queues. In this scenario, where the user has chosen the use whole ADT as the payload, the AQAdapter wizard has generated the schema in SCOTT_CUSTOMER_TYPE.xsd, according to the queue structure. During runtime, a xml that matches the schema will be created by the adapter for each message.

This section comprises the following topics:

- [Meeting Prerequisites](#)
- [Creating a New ESB Project](#)
- [Creating Inbound AQ Adapter](#)
- [Creating Outbound AQ Adapter](#)
- [Configuring Routing Service](#)
- [Checking the ESB Console](#)
- [Checking Execution in the ESB Control](#)

3.4.1 Meeting Prerequisites

The following prerequisites have to be met before you create the sample service:

1. Access SCOTT schema in some database.
2. Find data-sources.xml in
C:\product\10.1.3.1\OracleAS_1\bpel\system\appserver\oc4j\j2ee\home\config\data-sources.xml
3. Add the following code to data-sources.xml to define the data source for jdbc/aqSample. Note that URL should refer to the endpoint database that we use.

```

<connection-pool name="aqSample_CONNECTION_POOL">
  <connection-factory factory-class="oracle.jdbc.OracleDriver"
url="jdbc:oracle:thin:scott/tiger@stadd14:1521:db5617"/>
</connection-pool>
<managed-data-source name="AQSamplesDataSource"
connection-pool-name="aqSample_CONNECTION_POOL"
jndi-name="eis/AQ/MyConnection" />

```

3.4.2 Creating a New ESB Project

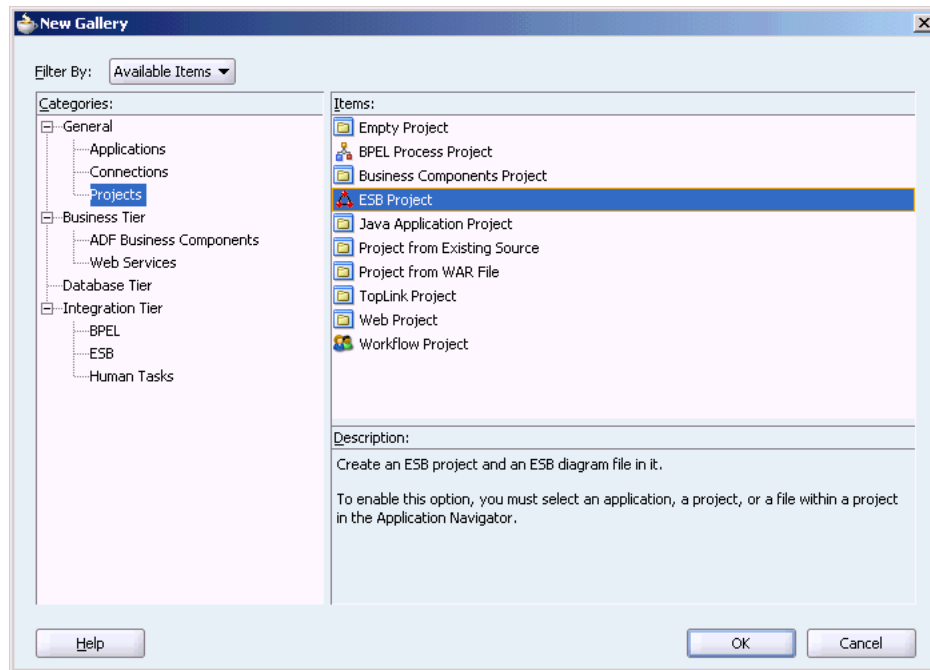
The following are the steps to create a new ESB project:

1. Open Oracle JDeveloper.
2. From the **File** menu, select **New**.

The New Gallery dialog box is displayed.

3. Select **All Technologies** from the **Filter By** box. This displays a list of available categories.
4. Expand the **General** node, and then select **Projects**.
5. Select **ESB Project** from the Items group, as shown in [Figure 3–18](#)

Figure 3–18 Creating a New ESB Project



6. Click **OK**.
7. In the **Project Name** field, enter a descriptive name. For example, `AQADTForESB`.
8. Click **OK**.

You have created a new ESB project, `AQADTForESB`.

3.4.3 Creating Inbound AQ Adapter

The following are the steps to create an inbound AQ adapter service:

1. Select **Adapter Services** from the Component Palette, and then drag and drop **Database Adapter** into the `AQADTForESB.esb` project.

The Create AQ Adapter Service dialog box is displayed, as shown in [Figure 3–19](#).

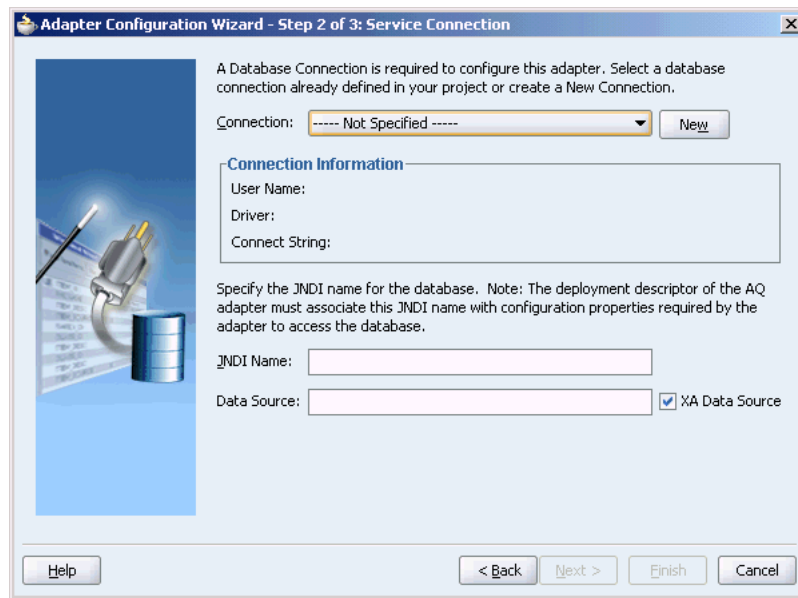
Figure 3–19 Creating an Inbound AQ Adapter Service

2. Specify the following information in AQ Adapter Service dialog box:
 - **Name:** Type a name for the service. In this example, type **Inbound**.
 - **System/Group:** Retain the default value.

Figure 3–20 shows the AQ Adapter Service dialog box with the **Name** and **System/Group** fields, filled up.

Figure 3–20 Defining the AQ Adapter Service

3. Under Adapter Service WSDL, click the **Configure adapter service wsdl** icon.
The Adapter Configuration wizard Welcome page is displayed.
4. Click **Next**.
The Service Name dialog box is displayed with the Service Name field, filled up.
5. Retain the service name, and click **Next**.
The Service Connection dialog box is displayed, as shown in Figure 3–21

Figure 3–21 The Service Connection Dialog Box

6. Click **New** to define a database connection.
The Create Database Connection Wizard Welcome page is displayed.
7. Click **Next**.
The Type dialog box is displayed.
8. Enter the following information in the Type dialog box:
 - a. In the **Connection Name** field, specify a unique name for the database connection. In this example, type **MyConnection**.
 - b. From the **Connection Type** box, select **Oracle (JDBC)**.
9. Click **Next**.
The Authentication dialog box is displayed.
10. Enter the authentication credentials in the following fields:
 - a. In the **UserName** field, specify a unique name for the database connection. In this example, type **scott**.
 - b. In the **Password** field, specify a password for the database connection. In this example, type **tiger**.
 - c. Leave the **Role** field blank.
 - d. Select **Deploy Password**.
11. Click **Next**.
The Connection dialog box is displayed.
12. Enter information in the following fields:
 - a. In the **Driver** list, retain the default value, **Thin**.
 - b. In the **Host Name** field, retain the default value, **localhost**.
 - c. In the **JDBC Port** field, specify the port number for the database connection. In this example, type **1521**.

- d. In the **SID** field, specify a unique SID value for the database connection. In this example, type **ORCL**.

13. Click Next.

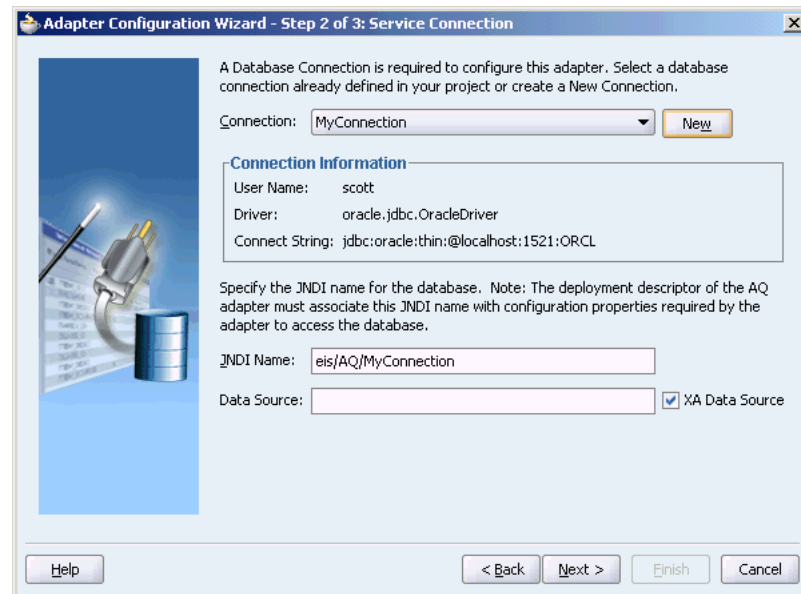
The Test dialog box is displayed.

- 14. Click Test Connection** to determine whether the specified information establishes a connection with the database.

- 15. Click Finish** to complete the process of creating a new database connection.

The Service Connection dialog box is displayed, providing a summary of the database connection, as shown in [Figure 3–22](#).

Figure 3–22 The Service Connection Dialog Box

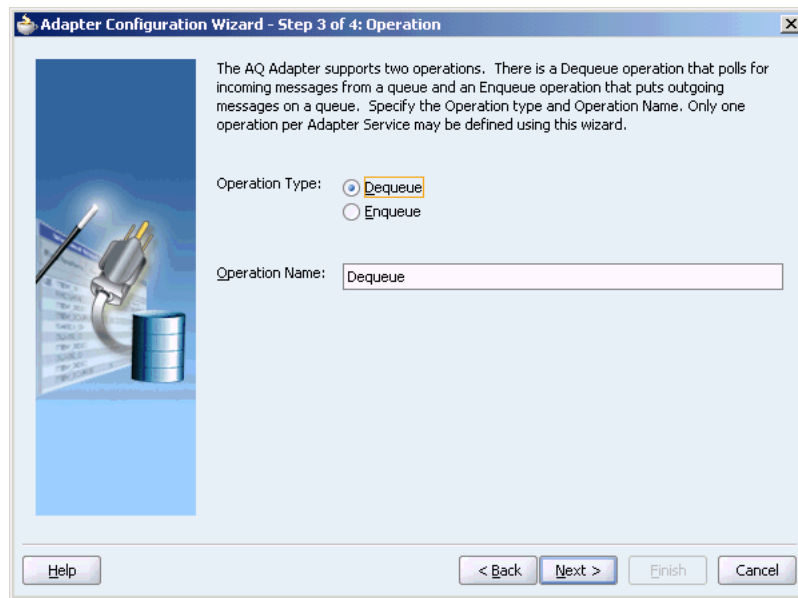


- 16. Click Next.**

The Operation dialog box is displayed.

- 17. Select Dequeue**, as shown in [Figure 3–23](#), and then click **Next**.

The Queue Name dialog box is displayed.

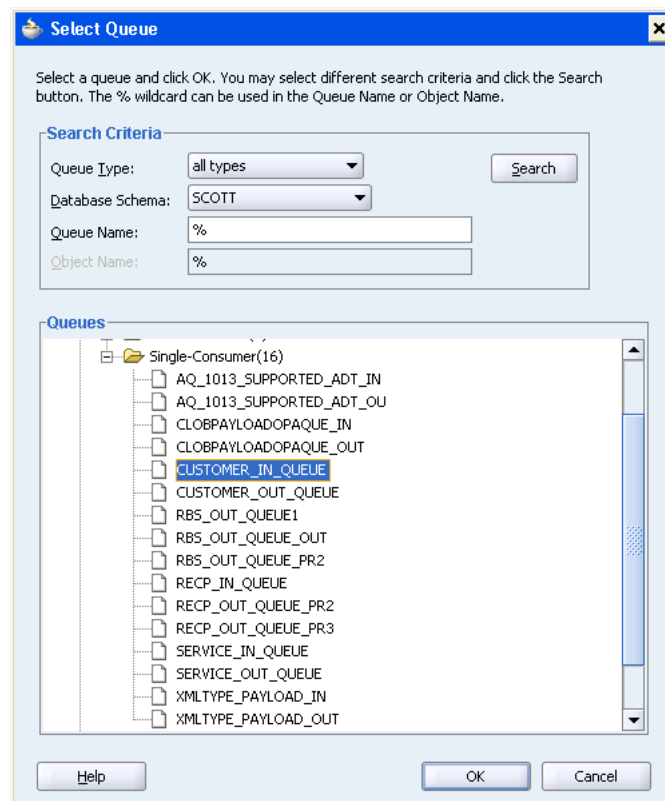
Figure 3–23 The Operation Dialog Box

18. Select a database schema from the drop-down list, or click **Browse** to browse for the schema. In this example, click **Browse**.

The Select Queue dialog box is displayed.

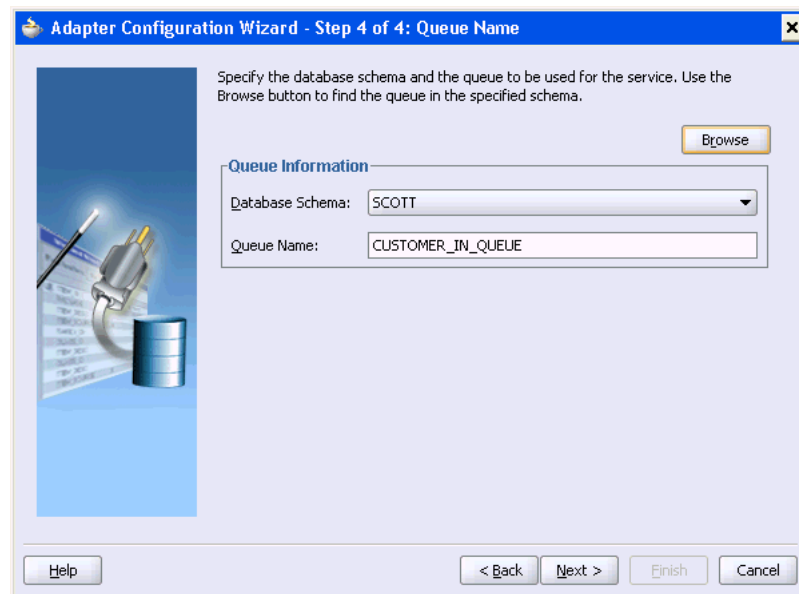
19. In the Select Queue dialog box, perform the following steps:
 - a. For Queue Type, select **all types**.
 - b. For Database Schema, select **Scott**.
 - c. Retain the default values for the other fields.
 - d. Under Queues, select **CUSTOMER_IN_QUEUES**.

Figure 3–24 shows the Select Queue dialog box with all the fields, filled up.

Figure 3–24 Selecting a Queue for the Inbound Operation

20. Click OK.

The Queue Name dialog box with all the fields populated is displayed, as shown in Figure 3–25.

Figure 3–25 The Select Queue Dialog Box

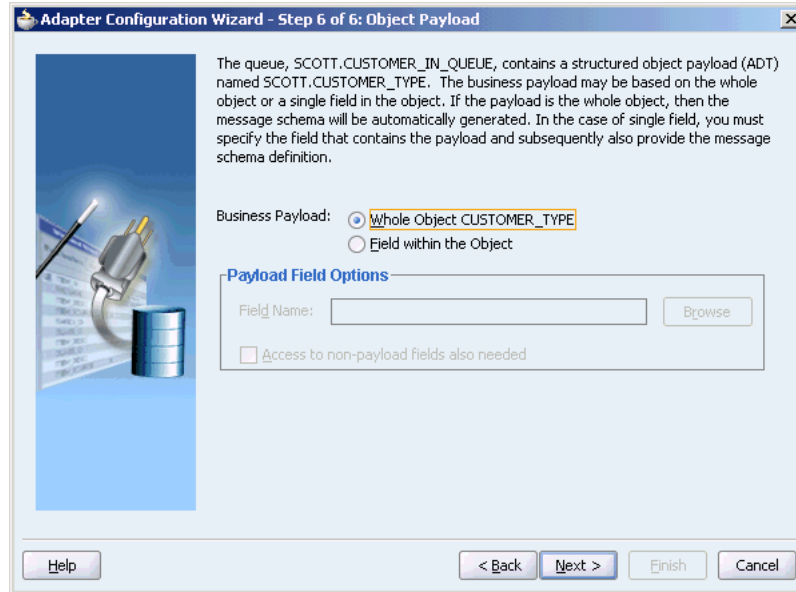
21. Click Next.

The Queue Parameters dialog box is displayed.

22. In the Queue Parameters dialog box, leave the fields empty, and then click **Next**.

The Object Payload dialog box is displayed, as shown in [Figure 3–26](#).

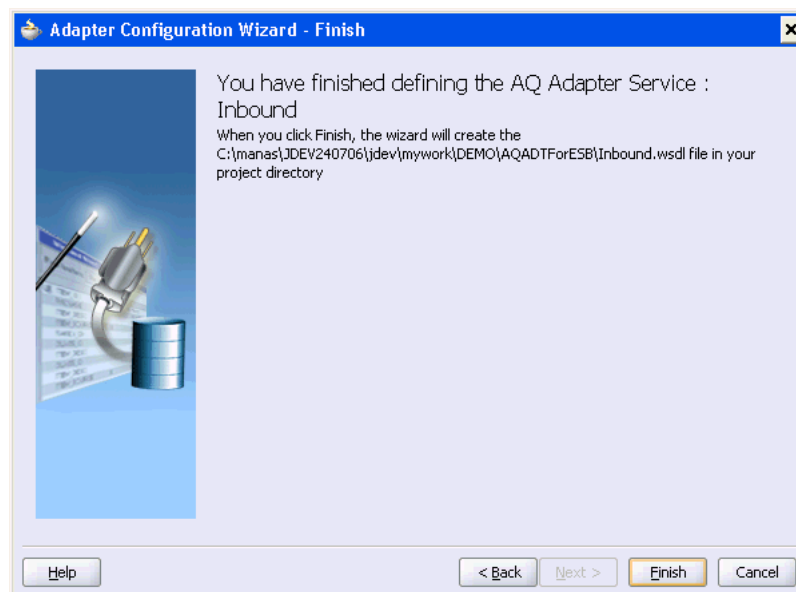
Figure 3–26 The Object Payload Dialog Box



23. Select a business payload: either the entire object, or just one field within the object. In this example, select **Whole Object CUSTOMER_TYPE**.
24. Click **Next**.

The Finish screen is displayed, as shown in [Figure 3–27](#). This dialog box shows the path and name of the adapter file that the wizard creates.

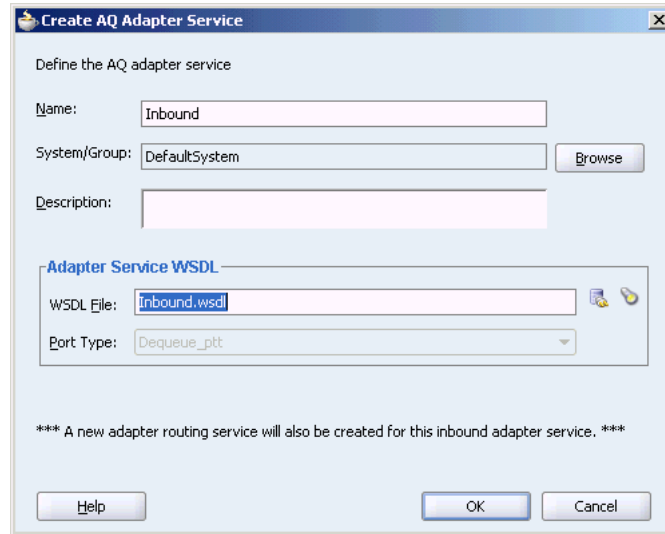
Figure 3–27 The Finish screen



25. In the Finish window, click **Finish**.

The Create AQ Adapter Service dialog box is displayed, with the fields filled up, as shown in [Figure 3-17](#).

Figure 3-28 The Create AQ Adapter Service Dialog Box



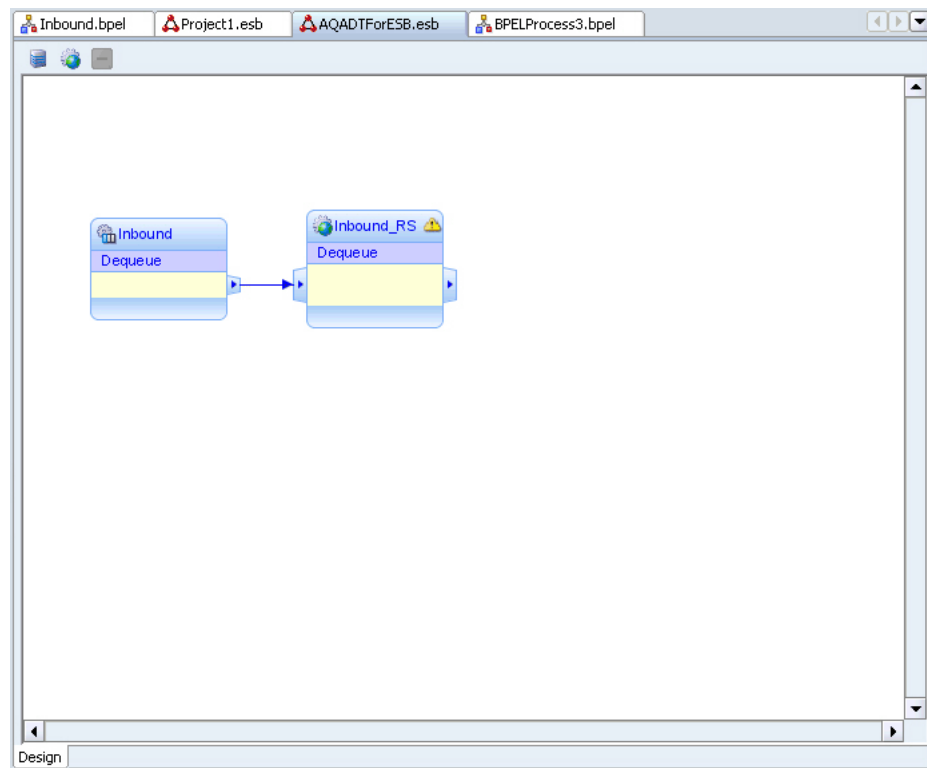
The dialog box titled "Create AQ Adapter Service" contains the following fields and controls:

- Name:** Inbound
- System/Group:** DefaultSystem (with a Browse button)
- Description:** (empty text box)
- Adapter Service WSDL:**
 - WSDL File:** Inbound.wsdl (with a file selection icon)
 - Port Type:** Dequeue_ptt (dropdown menu)
- Footer:**
 - *** A new adapter routing service will also be created for this inbound adapter service. ***
 - Buttons: Help, OK, Cancel

26. Click **OK**.

The mid pane of the window will resemble [Figure 3-29](#).

Figure 3-29 Completing the Inbound AQ Adapter Service



3.4.4 Creating Outbound AQ Adapter

The following are the steps to create an outbound AQ adapter service:

1. Select **Adapter Services** from the Component Palette, and then drag and drop **Database Adapter** into the `AQADTForESB.esb` project.

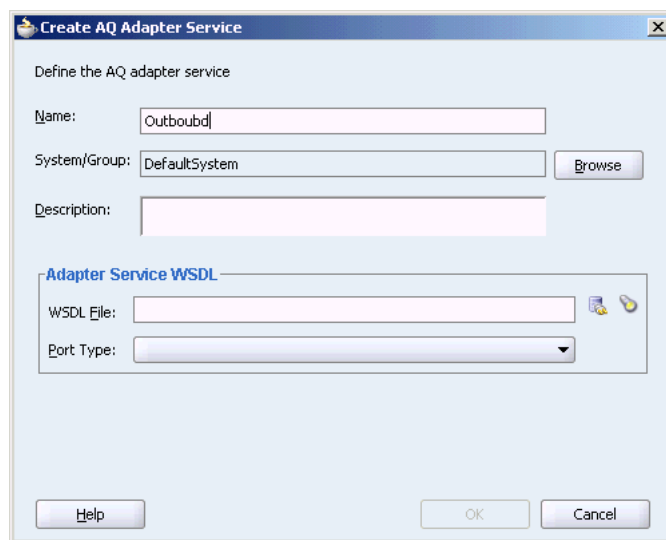
The Create AQ Adapter Service dialog box is displayed.

2. Specify the following information in AQ Adapter Service dialog box:

- **Name:** Type a name for the service. In this example, type `Outbound`.
- **System/Group:** Retain the default value.

Figure 3–30 shows the AQ Adapter Service dialog box with the **Name** and **System/Group** fields, filled up.

Figure 3–30 Defining an Outbound AQ Adapter Service



3. Under Adapter Service WSDL, click the **Configure adapter service wsdl** icon.

The Adapter Configuration wizard Welcome page is displayed.

4. Click **Next**.

The Service Name dialog box is displayed with the Service Name field, filled up.

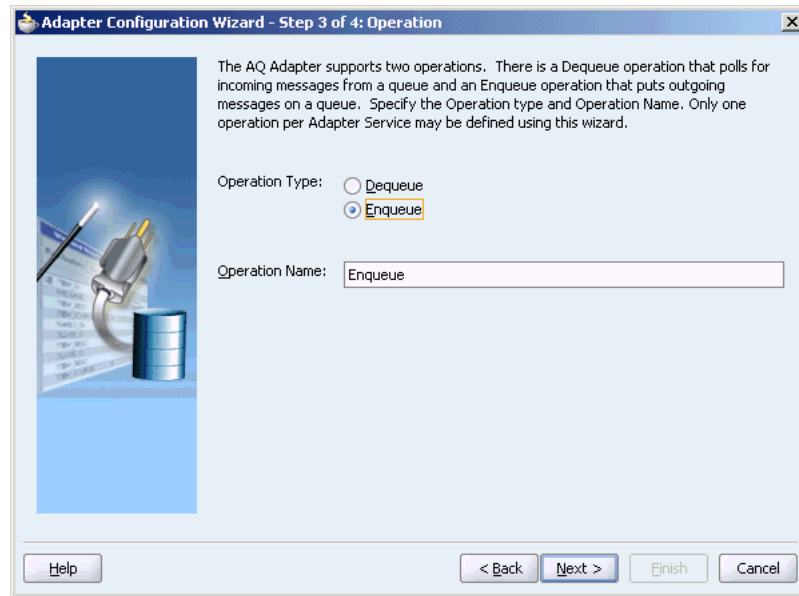
5. Retain the service name, and click **Next**.

The Service Connection dialog box is displayed.

6. For Connection, select **MyConnection**, and then click **Next**.

The Operation dialog box is displayed.

7. In the Operation dialog box, select **Equeue**, as shown in Figure 3–31.

Figure 3–31 The Operation Dialog Box

8. Click Next.

The Queue Name dialog box is displayed.

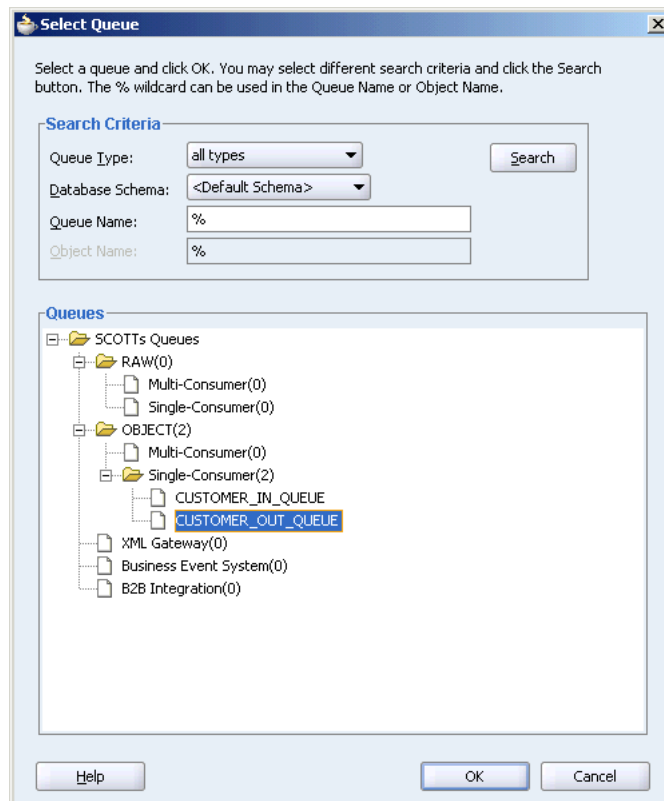
9. In the Queue Name dialog box, select a database schema from the drop-down list, or click **Browse to browse for the schema. In this example, click **Browse**.**

The Select Queue dialog box is displayed.

10. In the Select Queue dialog box, perform the following steps:

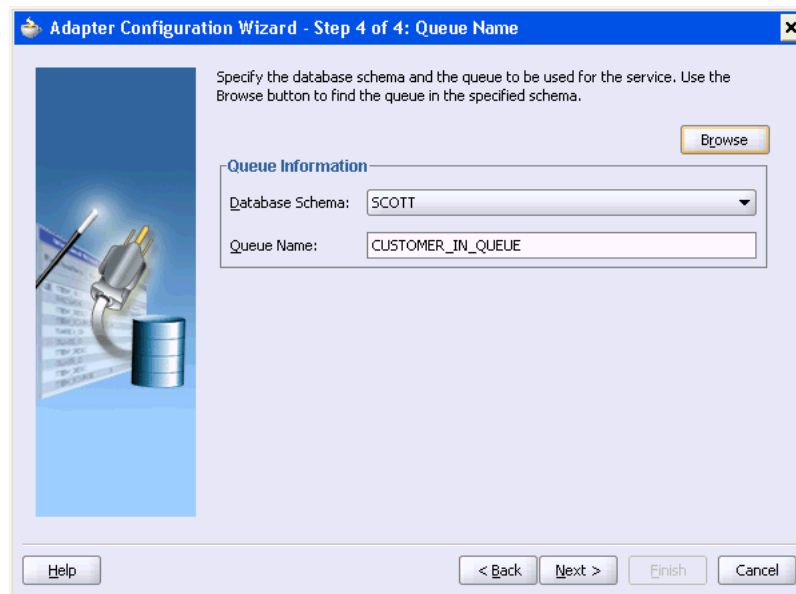
- a. For Queue Type, select **all types**.**
- b. For Database Schema, select **Scott**.**
- c. Retail the default values for the other fields.**
- d. Under Queues, select **CUSTOMER_OUT_QUEUES**.**

Figure 3–32 shows the Select Queue dialog box with all the fields, filled up.

Figure 3–32 Selecting a Queue for the Outbound Operation

11. Click **OK**.

The Queue Name dialog box with all the fields populated is displayed, as shown in [Figure 3–33](#).

Figure 3–33 The Queue Name Dialog Box

12. Click **Next**.

The Queue Parameters dialog box is displayed.

13. In the Queue Parameters dialog box, leave the fields empty, and then click **Next**.

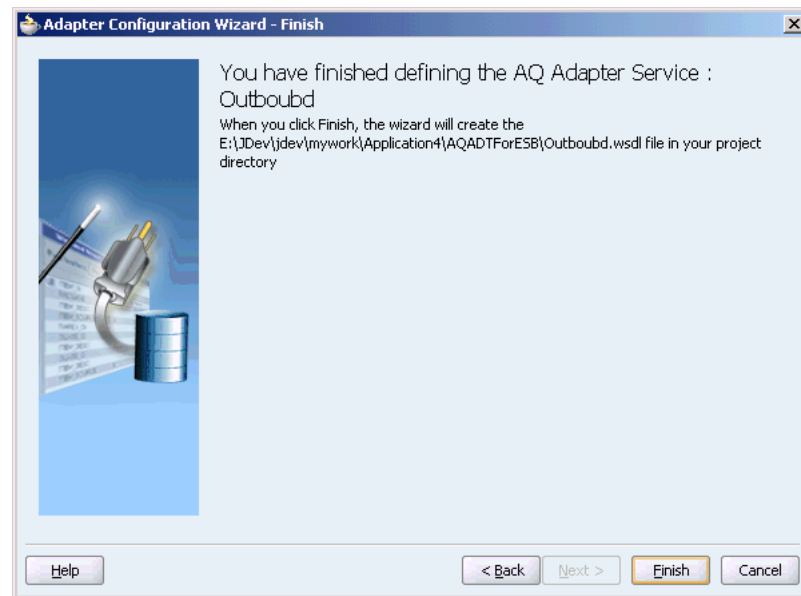
The Object Payload dialog box is displayed.

14. Select a business payload: either the entire object, or just one field within the object. In this example, select **Whole Object CUSTOMER_TYPE**.

15. Click **Next**.

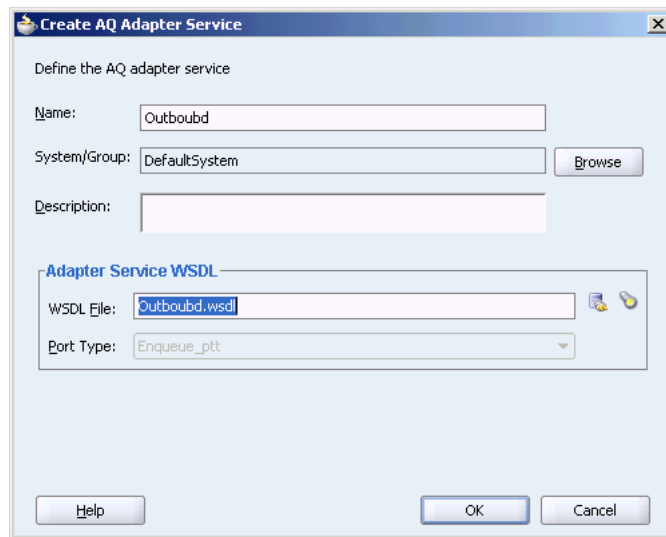
The Finish screen is displayed, as shown in [Figure 3–34](#). This dialog box shows the path and name of the adapter file that the wizard creates.

Figure 3–34 The Finish screen



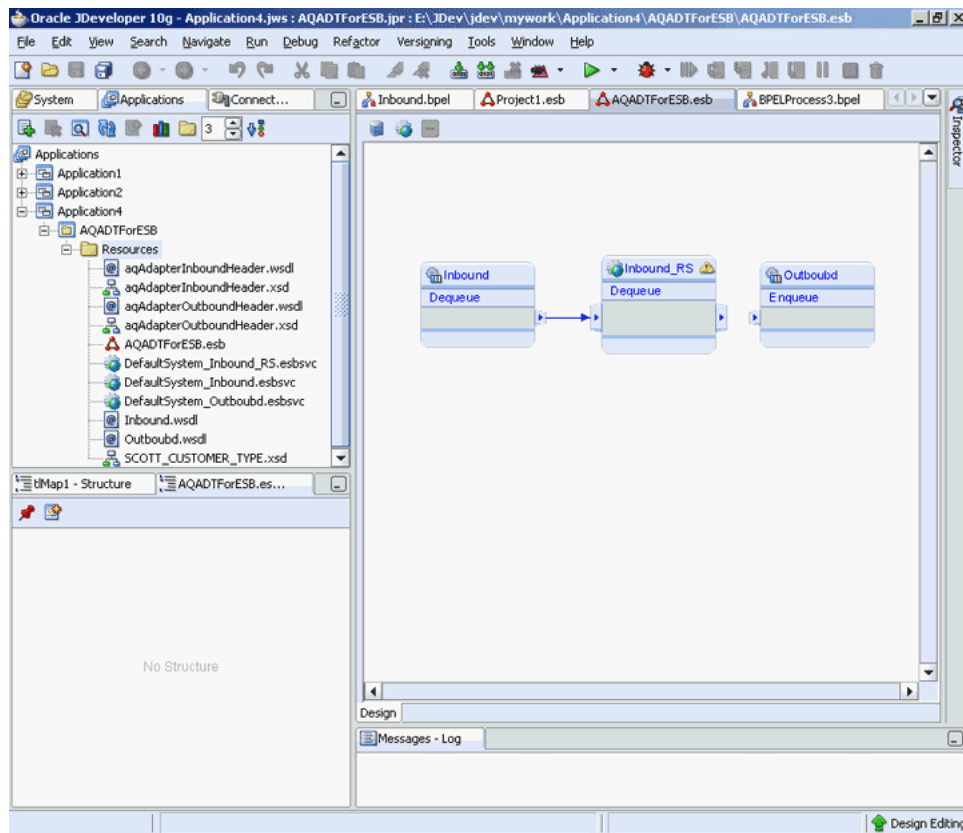
16. In the Finish window, click **Finish**.

The Create AQ Adapter Service dialog box is displayed, with the fields filled up, as shown in [Figure 3–35](#).

Figure 3–35 The Create AQ Adapter Service Dialog Box

17. Click **OK**.

The window will resemble [Figure 3–36](#).

Figure 3–36 Completing the Outbound AQ Adapter Service

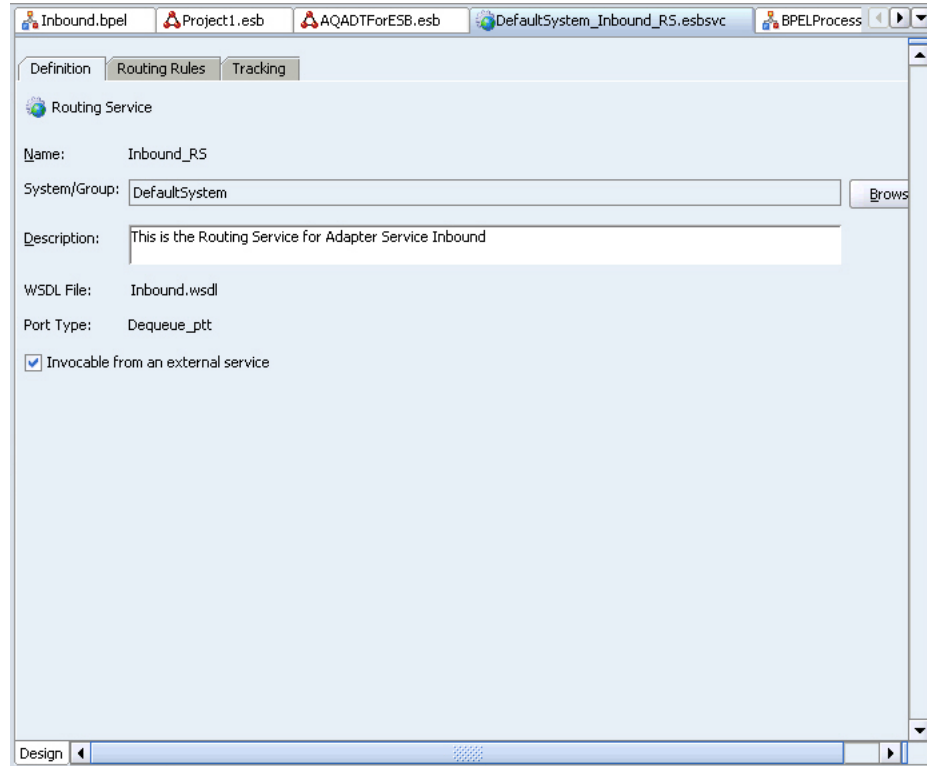
3.4.5 Configuring Routing Service

The following are the steps to configure the routing service:

1. Double-click the `DefaultSystem_Inbound_RS.esbsvc`.

The Routing Service window is displayed in the midpane of the Application window, as shown in [Figure 3–37](#).

Figure 3–37 The Routing Screen Window

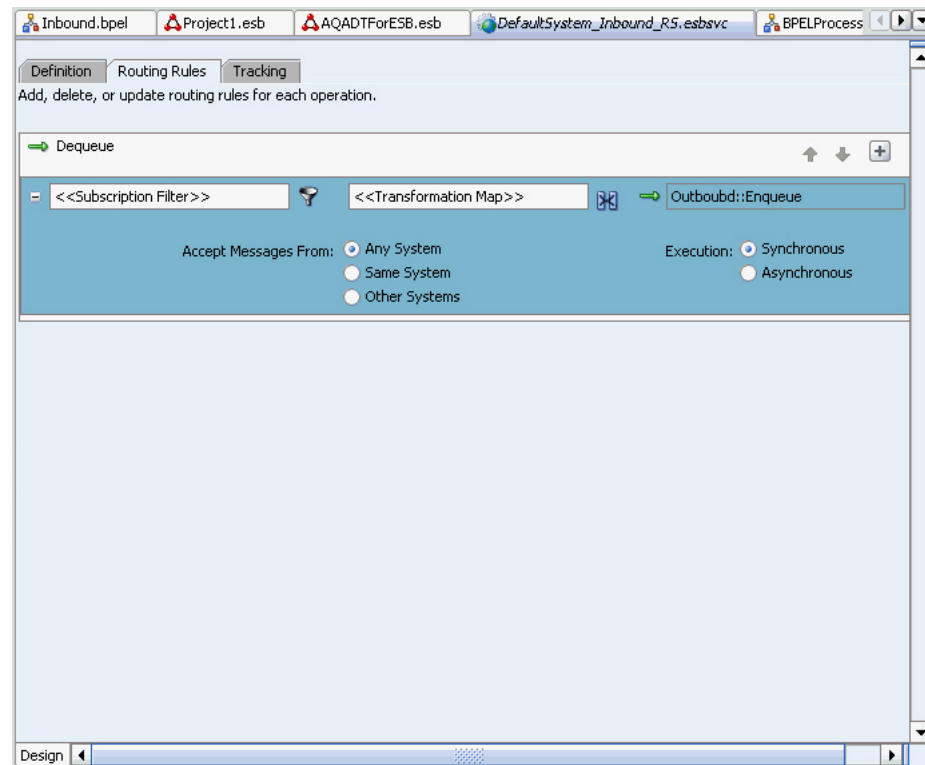


2. Select the **Routing Rules** tab, and then click on the + icon to add a rule.
The Browse Target Service Operation dialog box is displayed.
3. Select the **Enqueue** service, as shown in [Figure 3–38](#), and then click **OK**.

Figure 3–38 The Browse Target Service Operation Dialog Box

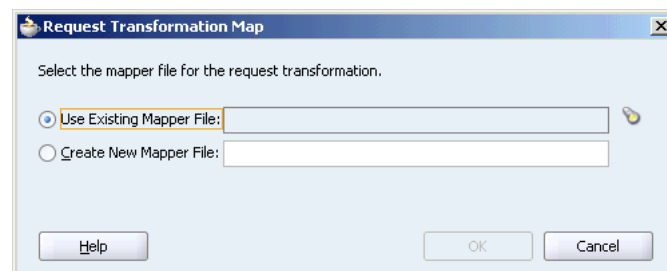
4. Click **Ok**.

The middle pane of the application window will resemble [Figure 3–39](#).

Figure 3–39 Selecting the Transformation Map

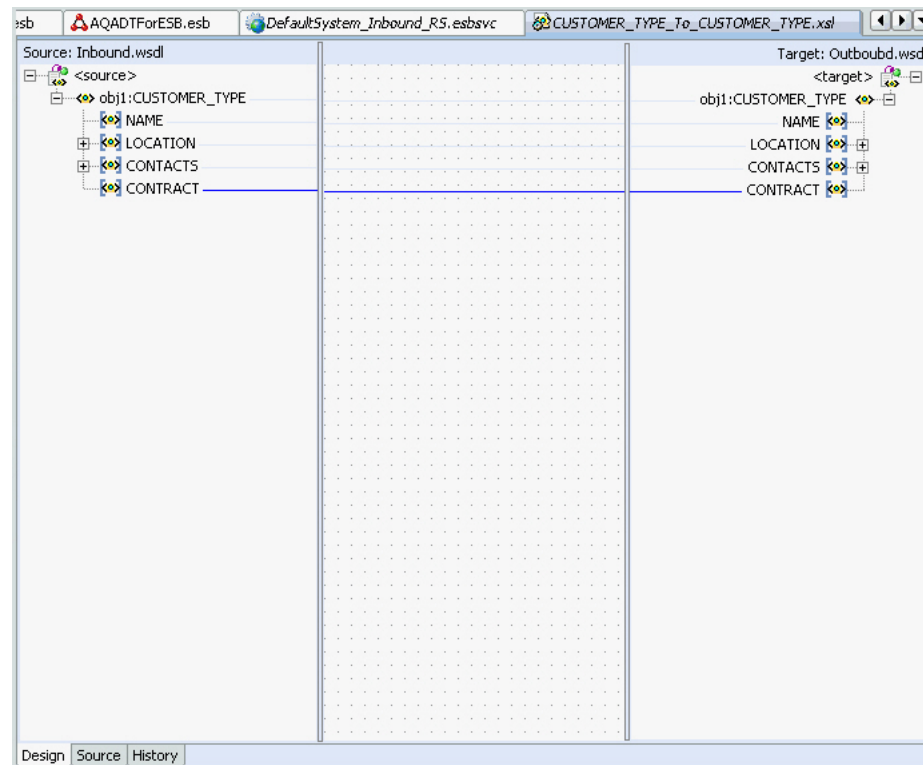
5. Double-click the **Transformation** icon, and then click the **Select Create New Mapper File** icon.

The Request Transformation Map dialog box is displayed, as shown in [Figure 3–40](#).

Figure 3–40 The Request Transformation Map Dialog Box

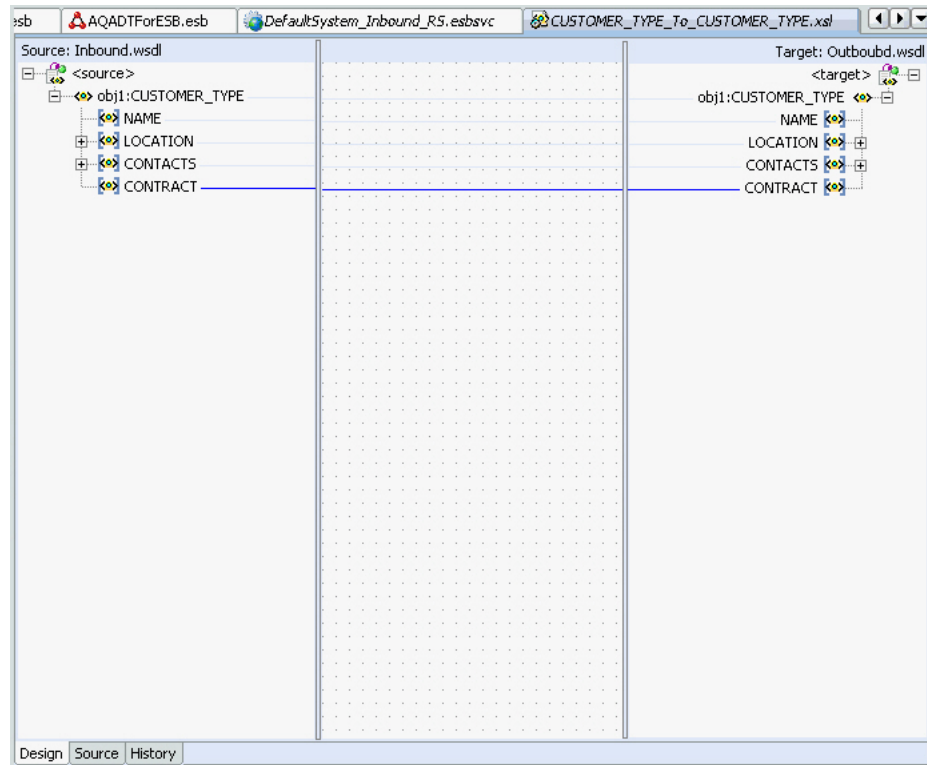
6. Select the **Create New Mapper File** option.
7. Accept the default values, and click **OK**.

The Transformation window appears, as shown in [Figure 3–41](#).

Figure 3–41 Transformation Definitions

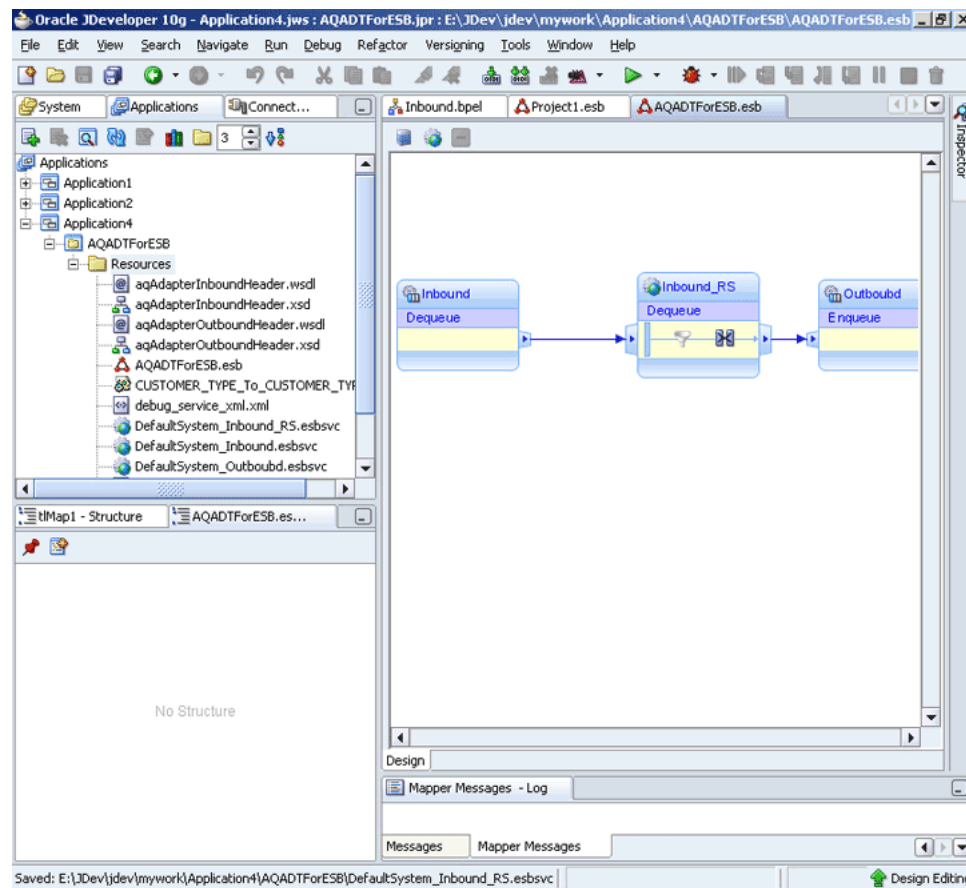
8. Select elements on the left-hand side of the mapper and drag it over to the elements on the right-hand side to set the map preferences.

The middle pane of the application window will resemble [Figure 3–42](#).

Figure 3–42 Setting Map Preferences

9. Save and close the tab for the mapper.
10. Save and close the tab for the routing service.

The AQADTForESB project will resemble [Figure 3–43](#).

Figure 3–43 The AQADTForESB Project After Setting Map Preferences

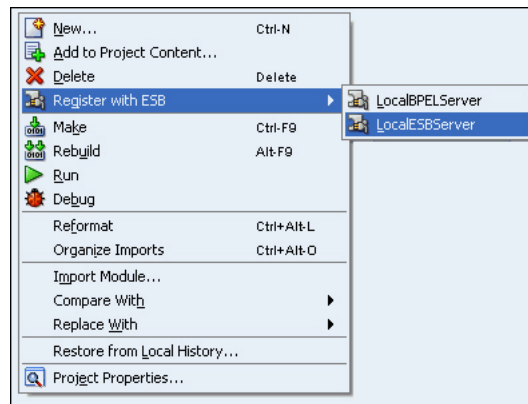
11. Edit `oc4j-ra.xml` to reflect your database connection. For example,

```
eis/DB/MyConnection
```

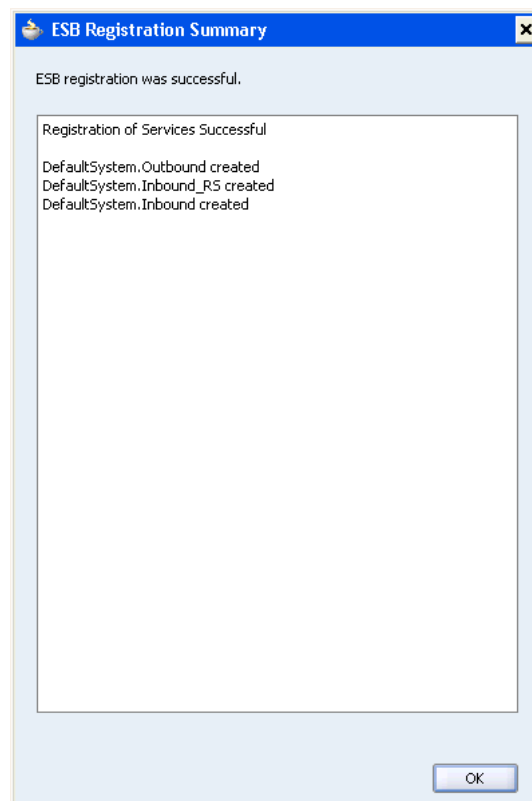
`oc4j-ra.xml` is available at the location where you have installed OracleAS.
For example:

```
C:\product\10.1.3.1\OracleAS_
1\j2ee\home\application-deployments\default\AQAdapter\oc4j-ra.xml
```

12. Right-click the project, select **Register with ESB**, and then click **LocalESBServer**, as shown in [Figure 3–44](#).

Figure 3–44 Deploying the Project

The Success page is displayed, as shown in [Figure 3–45](#).

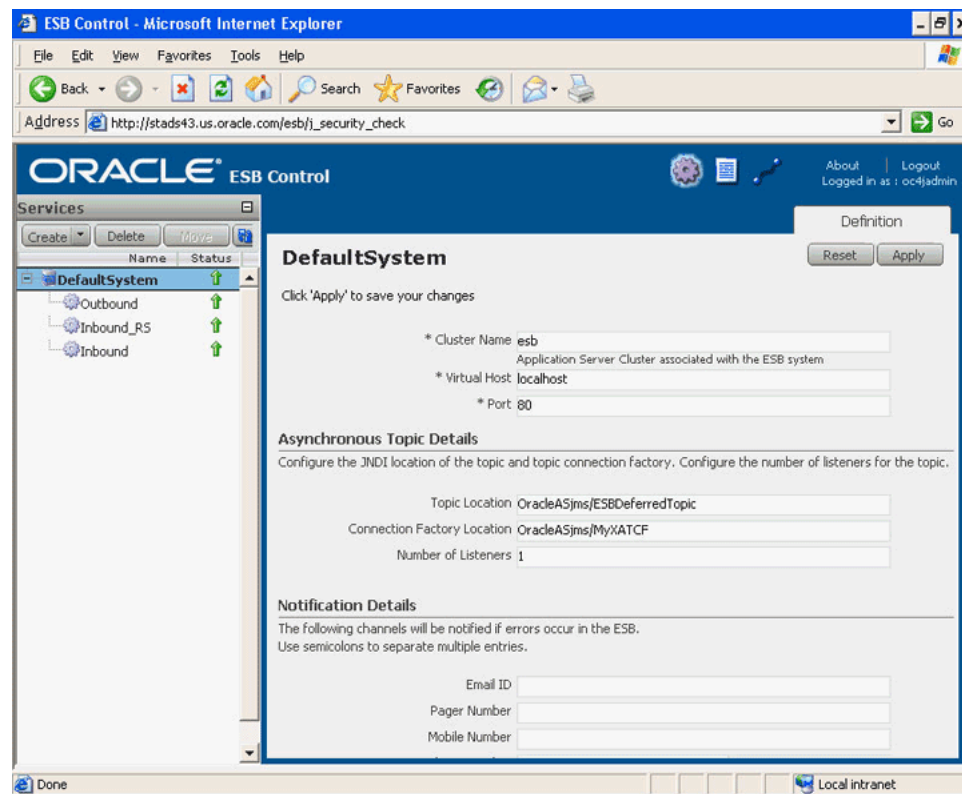
Figure 3–45 The ESB Registration Summary Page

3.4.6 Checking the ESB Console

To check the ESB control, open the ESB Console. For example:

<http://localhost:8888/esb/esb/EsbConsole.html>

Now, your service window will resemble [Figure 3–46](#):

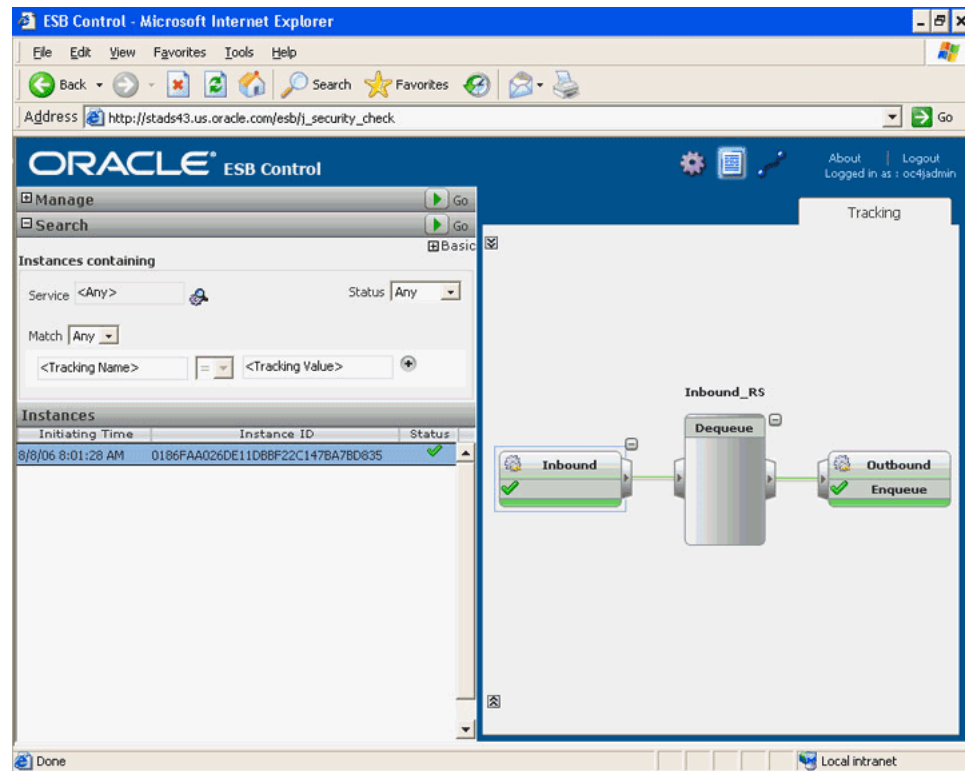
Figure 3–46 The ESB Console

3.4.7 Checking Execution in the ESB Control

Use the following steps to check execution in the ESB control:

1. Open the ESB Console.
2. Click **Instances** on top-right corner.
3. Click the green arrow next to **Search**.

An instance resembling [Figure 3–47](#) is displayed.

Figure 3–47 The ESB Control Instance

Oracle Application Server Adapter for Databases

This chapter describes the Oracle Application Server Adapter for Databases (database adapter), which works in conjunction with Oracle BPEL Process Manager and Oracle Enterprise Service Bus. Support for stored procedures and functions (for Oracle databases only) is also described. References to use cases for the database adapter and for stored procedures are provided.

This chapter contains the following topics:

- [Section 4.1, "Introduction to the Database Adapter"](#)
- [Section 4.2, "Database Adapter Concepts"](#)
- [Section 4.3, "Database Adapter Use Case for Oracle BPEL Process Manager"](#)
- [Section 4.4, "Database Adapter Use Cases for Oracle Enterprise Service Bus"](#)
- [Section 4.5, "Advanced Configuration"](#)
- [Section 4.7, "Stored Procedure and Function Support"](#)
- [Section 4.8, "Use Case for Creating and Configuring a Stored Procedure in JDeveloper BPEL Designer"](#)

4.1 Introduction to the Database Adapter

The database adapter enables a BPEL process to communicate with Oracle databases or third-party databases through JDBC. The database adapter service is defined within a BPEL process partner link using the Adapter Configuration Wizard of Oracle BPEL Process Manager.

This section contains the following topics:

- [Database Adapter Features](#)
- [Design Overview](#)
- [Database Adapter Integration with Oracle BPEL Process Manager](#)
- [Database Adapter Integration with Oracle Enterprise Service Bus](#)

4.1.1 Database Adapter Features

The database adapter connects to any relational database. For nonrelational databases and legacy systems, application and mainframe adapters are available. See *Oracle Application Server Adapter Concepts* for information about application and mainframe adapters.

To access an existing relational schema, you use the Adapter Configuration Wizard to do the following:

- Import a relational schema and map it as an XML schema (XSD)
See [Relational-to-XML Mapping](#) for more information.
- Abstract SQL operations such as `SELECT`, `INSERT`, and `UPDATE` as Web services
See [SQL Operations as Web Services](#) for more information.

While your BPEL process deals with XML and invokes Web services, database rows and values are queried, inserted, and updated. Unlike other solutions that give you a way to access data using a fixed schema, stored procedures, streams, or queues, with the database adapter, you access table data directly and transparently.

Features of the database adapter include:

- Compliance with open standards. The database adapter is an implementation of the JCA 1.5 connector. Like the other adapters that work with Oracle BPEL Process Manager, the database adapter is compatible with open standards such as BPEL, WSIF, and WSDL.
- Connectivity to any relational (SQL 92) database using JDBC, or ODBC using the Sun JdbcOdbcBridge
- Ability to map *any existing* relational schema to XML. The mapping is nonintrusive to the schema and no changes need to be made to it.
- Web services abstraction of SQL operations. The generated WSDL operations are `merge`, `insert`, `update`, `write`, `delete`, `select`, `queryByExample`, and `inbound polling`, which includes physical delete, logical delete, and sequencing-based polling strategies.
- Leveraging of TopLink technology, an advanced object-to-relational persistence framework. You can access the underlying TopLink project, and use the TopLink Workbench interface for advanced mapping and configuration, sequencing, batch and joined relationship reading, batch writing, parameter binding, statement caching, connection pooling, external transaction control (JTS and JTA), `UnitOfWork` for minimal updates, caching, optimistic locking, advanced query support, and query by example.
- Ability to execute arbitrary `sql`

See the following for more information:

- The Oracle BPEL Process Manager forum at
<http://forums.oracle.com/forums/forum.jsp?forum=212>
- The TopLink forum at
<http://forums.oracle.com/forums/forum.jsp?forum=48>

This site contains over 2,000 topics, such as implementing native sequencing, optimistic locking, and JTA-managed connection pools with TopLink

You can also access the forums from Oracle Technology Network at

<http://www.oracle.com/technology>

4.1.1.1 Querying over Multiple Tables

When executing a SQL `select` statement against multiple related tables there are the following three methods to build the SQL. These ways relate to how to pull in the detail records when the query is against the master record:

- [Using relationship Queries \(TopLink default\)](#)
- [Twisting the original `select` \(TopLink batch-attribute reading\)](#)
- [Returning a Single Result Set \(TopLink Joined-Attribute Reading\)](#)

The following sections contain an outline of these three methods and their comparison. However, note that when selecting rows from a single table there are no issues as against selecting from multiple tables.

Using relationship Queries (TopLink default)

Having selected a Master row, TopLink can always query separately to get all the details belonging to that Master table. These hidden queries (relationship queries) are cached in the TopLink metadata and need to be prepared only once.

Consider the SQL statement in following sample scenario:

```
SELECT DIRECTOR, ..., VIEWER_RATING
      FROM MOVIES
WHERE RATING = 'A';
```

For each master, this will be as follows:

```
SELECT CRITIC, ..., TITLE
      FROM MOVIE_REVIEWS
WHERE (TITLE = ?)
```

It enables you to bring in all the data with $1 + n$ query executions, where n is the number of master rows returned by the first query.

This approach is safe but slow, as a large number of round trips to the database are required to pull in all the data.

Twisting the original `select` (TopLink batch-attribute reading)

This feature allows TopLink to alter the original SQL `select` statement to read all the details in a second `select` statement as shown in the following example:

```
SELECT DIRECTOR, ..., VIEWER_RATING
      FROM MOVIES
WHERE RATING = 'A'
SELECT DISTINCT t0.CRITIC, ..., t0.TITLE
      FROM MOVIE_REVIEWS t0, MOVIES t1
WHERE ((t1.RATING = 'A') AND (t0.TITLE = t1.TITLE))
```

By considering the original `select` statement in pulling in the details, a total of two ($1 + 1 = 2$) query executions need to be performed.

Advantages

Batch attribute reading has the following advantages:

- All data read in two round trips to database
- This is a default feature in the 10.1.2.0.2 release

Disadvantages

Batch attribute reading has the following disadvantages:

- When using `maxTransactionSize` (on polling receive) or `maxRows` (on invoke select) to limit the number of rows loaded into memory at a time, these settings do not easily carry over to the batch attribute query. It is easier to work with a cursored result when there is only a single result set. (Multiple cursors can be used with difficulty, if the original query has an order by clause).
- `TopLink` can alter a SQL statement, only when it is in a format it can understand. If you use the hybrid SQL approach and set custom SQL for the root select, then `TopLink` will not be able to interpret that SQL to build the batch select.
- The `DISTINCT` clause is used on the batch query, to avoid returning the same detail twice if two master happen to both point to it. The `DISTINCT` clause cannot be used when returning LOBs in the resultset.

Configuration

Configuration is on a per 1-1 or 1-M mapping basis. By default, all such mappings in the 10.1.2.0.2 release have this property set.

Returning a Single Result Set (`TopLink` Joined-Attribute Reading)

The detail tables are outer-joined to the original SQL select statement, returning both master and detail in a single result set, as shown in the following example:

```
SELECT DISTINCT t1.DIRECTOR, ..., t1.VIEWER_RATING, t0.CRITIC, ..., t0.TITLE
FROM MOVIE_REVIEWS t0, MOVIES t1
WHERE ((t1.RATING = 'A') AND (t0.TITLE (+) = t1.TITLE))
```

This requires one query execution in total.

Advantages

The advantages include the following:

- In case of using `maxTransactionSize` while polling, the benefits of dealing with a single cursor can be great.
- When following the hybrid SQL route and entering custom SQL statements, you only have to deal with a single SQL statement, whereas `TopLink` normally uses a series of additional hidden SQL statements to bring in related rows.
- read consistency: Enables you to read all related rows at the same time, and not at different instances in time for the different tables.
- Performance can be ideal as only a single round trip to the database is required, whereas batch attribute reading requires one per table queried.

Disadvantages

There are some drawbacks however, namely the cost of returning duplicate data. For example, consider that you read the Master and Detail tables; Master has 100 columns per row, and Detail has 2 columns per row. Each row in the table, Master also, typically has 100 related Detail rows.

With one query per table, the result sets for the preceding example will look the following:

```
Master
Column1 column2 .... column100

Master1 ...

Detail
```

```

Detail
Column1 column2
Detail1 ...
Detail2
...
Detail100 ...

```

In this example, 300 column values are returned as shown:

$$\begin{aligned}
 &(\text{columns in master} + \text{columns in detail} \times \text{details per master}) = \\
 &(\quad 100 \quad + \quad 2 \quad \times \quad 100 \\
 &)= 300
 \end{aligned}$$

With one query for all tables, the result set will look like:

Master		Detail
Column1 Column2 ... Column100		Column1 Column2
Master1 ...		Detail1 ...
Master1 ...		Detail2 ...
Master1 ...		Detail100 ...

Note that, in the case of one query for all tables, 10,200 column values are returned in a single result set, versus 300 in two result sets, as shown here:

$$\begin{aligned}
 &((\text{columns in master} + \text{columns in detail}) \times \text{details per master}) = \\
 &((\quad 100 \quad + \quad 2 \quad) \times \quad 100 \quad) = 10,200
 \end{aligned}$$

This can have a serious drain on network traffic and computation, because 97 percent of the data returned is duplicate data. Also, if master had two related tables `detail1` and `detail2`, and there were 100 each per master, then the number of column values returned would be over 10 million per master row.

In general, you can use the following simple formula to estimate the relative cost of returning all rows in a single result set:

$$\begin{aligned}
 &(\text{Master columns} + \text{Detail1 columns} + \text{Detail2 columns} + \dots) \times \\
 &\quad \text{Detail1s per Master} \times \\
 &\quad \text{Detail2s per Master} \times \dots \\
 \text{bloat} = & \frac{\quad}{\quad} \\
 &(\text{Master columns} + \text{Detail1 columns} \times \text{Detail1s per Master} + \\
 &\quad \text{Detail2 columns} \times \text{Detail2s per Master} + \dots)
 \end{aligned}$$

Note that for 1-1 relationships this value is always 1, and if in the same example each master had two columns only and the details had 100 columns instead, and each master had only 3 or 4 details each, then the bloat would be

$$\text{bloat} = \frac{(2 + 100) \times 4}{(2 + 100 \times 4)} = \frac{408}{402} \approx 1$$

Another disadvantage is that this setting could distort the meaning of the `maxRows` setting on an outbound select.

Configuration

For example, assume that you have imported `Movies` and `MovieReviews`, with a 1-M attribute on `Movies` called `movieReviewsCollection`. The following are the steps to configure this:

1. In the BPEL project, choose **Application sources**, and then click **TopLink**.

2. In the Structure panel, click **Movies**, and then double-click **movieReviewsCollection**.

A view containing the check box, `Use Batch Reading` is displayed. By default, this box is enabled. You must deselect the check box to disable batch attribute reading (altering the SQL).

For a 1-1 relationship you may also see the option, **Use Joining** below the option, **Use Batch Reading**. This is similar to returning a single result set and the following is a list of a few key differences:

- The setting is on a per attribute basis, not a per query/wsdl operation basis.
- This setting is available only for 1-1 attributes.
- This setting does an inner join, that is, if there is a Master without a Detail, then that Master will be filtered out of the original select statement.

To configure returning a single result set, edit your wsdl and for each jca operation of type `DBActivationSpec` or `DBReadInteractionSpec`, add the property `ReturnSingleResultSet="true"`, as shown in the following example:

```
<operation name="SelectAllByTitleServiceSelect_title">
  <jca:operation
    InteractionSpec="oracle.tip.adapter.db.DBReadInteractionSpec"

    DescriptorName="SelectAllByTitle.Movies"
    QueryName="SelectAllByTitleServiceSelect"
    ReturnSingleResultSet="true"
    MappingsMetaDataURL="toplink_mappings.xml" />
  <input/>
</operation>
```

This setting will override altering the original select (batch attribute reading) statement.

Comparison of The Methods Used for Querying Over Multiple Tables

On the surface, returning a single result set looks best (1 query), followed by batch attribute reading (altering the select statement: 2 queries), and finally by default relationship reading (n + 1 queries). However, there are several pitfalls to both of the more advanced options, as explained below:

Altering User Defined SQL

If you specify custom/hybrid SQL, the TopLink cannot alter that SQL string to build the details select. For this reason, you should avoid using hybrid SQL and build selects using the wizards' visual expression builder as often as possible.

Show Me The SQL

The additional queries executed by TopLink in both, the default and batch attribute reading cases can be somewhat of a mystery to users. For this reason, the raw SQL shown to users in the DBAdapter wizard assumes returning a single result set, to make things clearer and also to improve manageability.

Returning Too Many Rows At Once

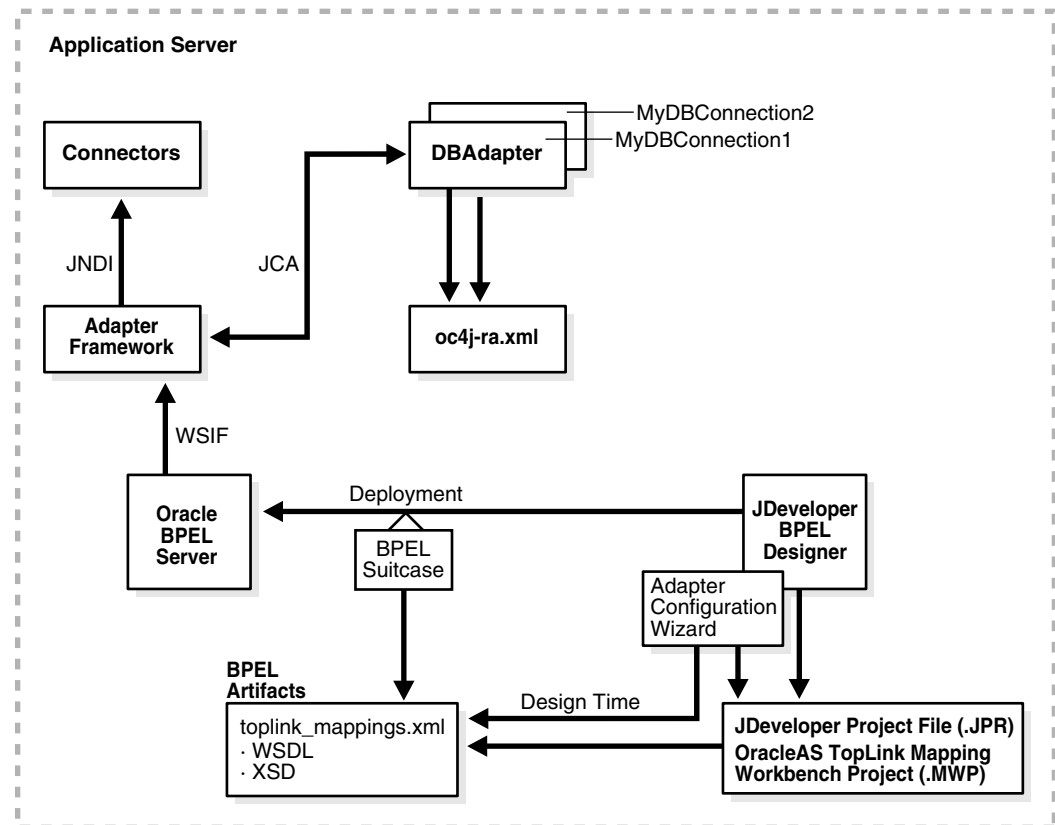
Databases can store vast quantities of information, and a common pitfall of select statements which return too much information at once. On a DBAdapter receive, a `maxTransactionSize` property can be set to limit the number of rows which are read from a cursored result set and processed in memory at a time. A similar

`max-rows` setting exists for one time invoke select statements. However, this setting is very risky.

4.1.2 Design Overview

Figure 4–1 shows how the database adapter interacts with the various design-time and deployment artifacts.

Figure 4–1 How the Database Adapter Works



The database adapter is a separate JCA 1.5 connector. It is deployed to the application server during installation, and is configured using `oc4j-ra.xml`. The `oc4j-ra.xml` file is the database adapter deployment descriptor file for Oracle Application Server.

Each entry in `oc4j-ra.xml` has a Java Naming and Directory Interface (JNDI) name (location) and session and login properties, and represents a single database and database adapter instance. The connector is tied to the application server; therefore, it can be used independently, but any change to `oc4j-ra.xml` requires restarting the application server. This file is created by the application server the first time Oracle BPEL Server comes up. Therefore, in a standalone installation, you do not see this file unless you start Oracle BPEL Server at least once.

When a business process is executed in Oracle BPEL Process Manager, a Web service (WSDL) may be invoked (using WSIF) against the database. The `jca:address` tag in the WSDL is used to look up an adapter instance, and the `jca:operation` tag in the WSDL is used to set up an interaction (outbound) or activation (inbound) with the database adapter using a JCA interface. The `jca:operation` tag contains a link to TopLink metadata for executing a query to push XML data to a relational schema, or vice versa.

The `toplink_mappings.xml` file and WSDL (with custom `jca:operation` and `jca:address` tags) are created during design time. In JDeveloper BPEL Designer, you create an endpoint, or partner link, for interacting with the database. Each partner link has its own WSDL. The WSDL defines all the operations (queries) that can be performed with that database.

To create the WSDL, you use the Adapter Configuration Wizard, where you import the relational schema, map it to an XML schema, and define one or more operations. This produces an XSD representing the schema and a `toplink_mappings.xml` file.

The Adapter Configuration Wizard creates an TopLink Workbench project (a `.mwp` file) as part of the JDeveloper BPEL Designer project. Like the JDeveloper BPEL Designer `.jpr` file, it enables a user to go back and visually change mappings or leverage TopLink to set advanced properties. Saving the MWP project does not regenerate `toplink_mappings.xml`; that is done by running through the wizard again in edit mode. (No changes are needed; you simply run through it.)

During deployment, a copy of `toplink_mappings.xml` is included in the BPEL suitcase. It is later read by the database adapter and the metadata is cached.

The database adapter is used for relational-to-XML mapping; therefore, no Java class files are needed. The database adapter generates byte codes for the classes in memory based on information in the descriptors. You do not compile class files or deal with class path issues when using the database adapter. The MWP project in the JDeveloper BPEL Designer project may create Java files as a by-product of using the wizard, but they are needed at design time only.

4.1.3 Database Adapter Integration with Oracle BPEL Process Manager

Adapter Framework is used for the bidirectional integration of the J2CA 1.5 resource adapters with BPEL Process Manager. Adapter Framework is based on standards and employs the Web service Invocation Framework (WSIF) technology for exposing the underlying J2CA interactions as Web services.

See *Oracle Application Server Adapter Concepts* for information on database adapter architecture, adapter integration with Oracle BPEL Process Manager, and adapter deployments.

4.1.4 Database Adapter Integration with Oracle Enterprise Service Bus

Oracle Enterprise Service Bus Server supports the Oracle adapters and enables you to define inbound and outbound adapter services for each. An inbound adapter service receives data from an external data source and transforms it into an XML message. An outbound adapter service sends data to a target application by transforming an XML message into the native format of the given adapter.

Using Oracle Enterprise Service Bus Server you can send or receive messages extracted from an Oracle Database table or created by executing a stored procedure.

BPEL pre-dates ESB and most of this guide and the samples implicitly assume use with BPEL. However the adapters work equally well with either BPEL or ESB. For any mention of BPEL here you may substitute ESB instead.

4.2 Database Adapter Concepts

This section contains the following topics:

- [Relational-to-XML Mapping](#)

■ SQL Operations as Web Services

For advanced topics in Relational-XML Mapping please see the section [Relational-to-XML Mappings \(toplink_mappings.xml\)](#) under [Advanced Configuration](#).

4.2.1 Relational-to-XML Mapping

For a flat table or schema, the relational-to-XML mapping is easy to see. Each row in the table becomes a complex XML element. The value for each column becomes a text node in the XML element. Both column values and text elements are primitive types.

[Table 4–1](#) shows the structure of the MOVIES table. This table is used in the use cases described in this chapter. See [Database Adapter Use Case for Oracle BPEL Process Manager](#) for more information.

Table 4–1 MOVIES Table Description

Name	Null?	Type
TITLE	NOT NULL	VARCHAR2 (50)
DIRECTOR	--	VARCHAR2 (20)
STARRING	--	VARCHAR2 (100)
SYNOPSIS	--	VARCHAR2 (255)
GENRE	--	VARCHAR2 (70)
RUN_TIME	--	NUMBER
RELEASE_DATE	--	DATE
RATED	--	VARCHAR2 (6)
RATING	--	VARCHAR2 (4)
VIEWER_RATING	--	VARCHAR2 (5)
STATUS	--	VARCHAR2 (11)
TOTAL_GROSS	--	NUMBER
DELETED	--	VARCHAR2 (5)
SEQUENCENO	--	NUMBER
LAST_UPDATED	--	DATE

The corresponding XML schema definition (XSD) is as follows:

```
<xs:complexType name="Movies">
  <xs:sequence>
    <xs:element name="director" type="xs:string" minOccurs="0" nillable="true"/>
    <xs:element name="genre" type="xs:string" minOccurs="0" nillable="true"/>
    <xs:element name="rated" type="xs:string" minOccurs="0" nillable="true"/>
    <xs:element name="rating" type="xs:string" minOccurs="0" nillable="true"/>
    <xs:element name="releaseDate" type="xs:dateTime" minOccurs="0"
nillable="true"/>
    <xs:element name="runTime" type="xs:double" minOccurs="0" nillable="true"/>
    <xs:element name="starring" type="xs:string" minOccurs="0" nillable="true"/>
    <xs:element name="status" type="xs:string" minOccurs="0" nillable="true"/>
    <xs:element name="synopsis" type="xs:string" minOccurs="0" nillable="true"/>
    <xs:element name="title" type="xs:string"/>
    <xs:element name="totalGross" type="xs:double" minOccurs="0" nillable="true"/>
    <xs:element name="viewerRating" type="xs:string" minOccurs="0"/>
```

```

nillable="true"/>
</xs:sequence>
</xs:complexType>

```

As the preceding code example shows, MOVIES is not just a single CLOB or XMLTYPE column containing the entire XML string. Instead, it is an XML `complexType` comprised of elements, each of which corresponds to a column in the MOVIES table. For flat tables, the relational-to-XML mapping is straightforward.

Table 4–2 and Table 4–3 show the structure of the EMP and DEPT tables, respectively. These tables are used in the MasterDetail use case. See [Database Adapter Use Case for Oracle BPEL Process Manager](#) for more information.

Table 4–2 EMP Table Description

Name	Null?	Type
EMPNO	NOT NULL	NUMBER (4)
ENAME	--	VARCHAR2 (10)
JOB	--	VARCHAR2 (9)
MGR	--	NUMBER (4)
HIREDATE	--	DATE
SAL	--	NUMBER (7, 2)
COMM	--	NUMBER (7, 2)
DEPTNO	--	NUMBER (2)

Table 4–3 DEPT Table Description

Name	Null?	Type
DEPTNO	NOT NULL	NUMBER (2)
DNAME	--	VARCHAR2 (14)
LOC	--	VARCHAR2 (13)

As the preceding table definitions show, and as is typical of a normalized relational schema, an employee's department number is not stored in the EMP table. Instead, one of the columns of EMP (DEPTNO) is a foreign key, which equals the primary key (DEPTNO) in DEPT.

However, the XML equivalent has no similar notion of primary keys and foreign keys. Consequently, in the resulting XML, the same data is represented in a hierarchy, thereby preserving the relationships by capturing the detail record inline (embedded) inside the master.

An XML element can contain elements that are either a primitive type (string, decimal), or a complex type, that is, another XML element. Therefore, an employee element can contain a department element.

The corresponding XML shows how the relationship is materialized, or shown inline. DEPTNO is removed from EMP, and instead you see the DEPT itself.

```

<EmpCollection>
  <Emp>
    <comm xsi:nil = "true" ></comm>
    <empno >7369.0</empno>
    <ename >SMITH</ename>

```

```

    <hiredate >1980-12-17T00:00:00.000-08:00</hiredate>
    <job >CLERK</job>
    <mgr >7902.0</mgr>
    <sal >800.0</sal>
    <dept>
      <deptno >20.0</deptno>
      <dname >RESEARCH</dname>
      <loc >DALLAS</loc>
    </dept>
  </Emp>
  ...
</EmpCollection>

```

Materializing the relationship makes XML human readable, and allows the data to be sent as one packet of information. No cycles are allowed in the XML; therefore, an element cannot contain itself. This is handled automatically by the database adapter. However, you may see duplication (that is, the same XML detail record appearing more than once under different master records). For example, if a query returned two employees, both of whom work in the same department, then, in the returned XML, you see the same DEPT record inline in both the EMP records.

Therefore, when you import tables and map them as XML, it is recommended that you avoid excessive duplication, although the database adapter does not print an element inside itself. The database adapter prints the following:

```

<Emp>
  <name>Bob</name>
  <spouse>
    <name>June</name>
  </spouse>
</Emp>

```

But not:

```

<Emp>
  <name>Bob</name>
  <spouse>
    <name>June</name>
    <spouse>
      <name>Bob</name>
    </spouse>
    ...
  </spouse>
</spouse>
</Emp>

```

To avoid duplication, you can do the following:

- Import fewer tables. If you import only EMP, then DEPT does not appear.
- Remove the relationship between EMP and DEPT in the wizard. This removes the relationship, but the foreign key column is put back.

In both these cases, the corresponding XML is as follows:

```

<EmpCollection>
  <Emp>
    <comm xsi:nil = "true" ></comm>
    <empno >7369.0</empno>
    <ename >SMITH</ename>
    <hiredate >1980-12-17T00:00:00.000-08:00</hiredate>
  </Emp>
</EmpCollection>

```

```

    <job >CLERK</job>
    <mgr >7902.0</mgr>
    <sal >800.0</sal>
    <deptno >20.0</deptno>
  </Emp>
  ...
</EmpCollection>

```

Note that one of the two preceding solutions is feasible only if getting back the foreign key suffices, as opposed to getting back the complete detail record in its entirety.

4.2.1.1 Relational Types to XML Schema Types

Table 4–4 shows how database data types are converted to XML primitive types when you import tables from a database.

Table 4–4 Mapping Database Data Types to XML Primitive Types

Database Type	XML Type (Prefixed with xs:)
VARCHAR, VARCHAR2, CHAR, NCHAR, NVARCHAR, NVARCHAR2, MEMO, TEXT, CHARACTER, CHARACTER VARYING, UNICHAR, UNIVARCHAR, SYSNAME, NATIONAL CHARACTER, NATIONAL CHAR, NATIONAL CHAR VARYING, NCHAR VARYING, LONG, CLOB, NCLOB, LONGTEXT, LONGVARCHAR, NTEXT	string
BLOB, BINARY, IMAGE, LONGVARBINARY, LONG RAW, VARBINARY, GRAPHIC, VARGRAPHIC, DBCLOB, BIT VARYING	base64Binary
BIT, NUMBER(1) DEFAULT 0, SMALLINT DEFAULT 0, SMALLINT DEFAULT 0	boolean
TINYINT, BYTE	byte
SHORT, SMALLINT	short
INT, SERIAL	int
INTEGER, BIGINT	integer
NUMBER, NUMERIC, DECIMAL, MONEY, SMALLMONEY, UNIQUEIDENTIFIER	decimal
FLOAT, FLOAT16, FLOAT(16), FLOAT32, FLOAT(32), DOUBLE, DOUBLE PRECIS, REAL	double
TIME, DATE, DATETIME, TIMESTAMP, TIMESTAMP(6), SMALLDATETIME, TIMESTAMP TZ, TIMESTAMPLTZ, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE	dateTime

Essentially, NUMBER goes to DECIMAL, the most versatile XML data type for numbers, VARCHAR2 and CLOB to string, BLOB to base64Binary (to meet the plain-text requirement), and date types to dateTime.

Any type not mentioned in this discussion defaults to `java.lang.String` and `xs:string`. Timestamp support is basic, because only the `xs:dateTime` format is supported. The BFILE, USER_DEFINED, OBJECT, STRUCT, VARRAY, and REF types are specifically not supported.

Because XML is plain text, BLOB and byte values are base 64/MIME encoded so that they can be passed as character data.

4.2.1.2 Mapping Any Relational Schema to Any XML Schema

The database adapter supports mapping any relational schema on any relational database to an XML schema, although not any XML schema of your choice, because the wizard generates the XML schema with no explicit user control over the layout of the elements. You can control how you map the schema in both the Adapter Configuration Wizard and later in TopLink Workbench. By pairing the database adapter with a transformation step, you can map any relational schema to any XML schema.

4.2.2 SQL Operations as Web Services

After mapping a relational schema as XML, you must also map basic SQL operations as Web services. Each operation discussed in the following sections has a corresponding tutorial and readme. It is recommended that you start with these and try to run one or more as you read this section. As the tutorials demonstrate, some operations translate directly to the SQL equivalent, while others are more complex.

See the following sections for details:

- [Use Cases for Outbound Invoke Operations](#)
- [Use Cases for Polling Strategies](#)

This section comprises the following topics:

- [DML Operations](#)
- [Polling Strategies](#)

4.2.2.1 DML Operations

Data manipulation language (DML) operations align with basic SQL `INSERT`, `UPDATE`, and `SELECT` operations. SQL `INSERT`, `UPDATE`, `DELETE`, and `SELECT` are all mapped to Web service operations of the same name. The `WRITE` is either an `INSERT` or `UPDATE`, based on the results of an existence check. A distinction is made between the data manipulation operations—called outbound writes—and the `SELECT` operations—called outbound reads. The connection between the Web service and the SQL for `merge` (the default for outbound write) and `queryByExample` are not as obvious as for basic SQL `INSERT`, `UPDATE`, and `SELECT`.

This section comprises the following topics:

- [Merge](#)
- [queryByExample](#)
- [Use Cases for Outbound Invoke Operations](#)

Merge

`Merge` first reads the corresponding records in the database, calculates any changes, and then performs a minimal update. `INSERT`, `UPDATE`, and `WRITE` make the most sense when you are thinking about a single row and a single table. However, your XML can contain complex types and map to multiple rows on multiple tables. Imagine a `DEPT` with many `EMPS`, each with an `ADDRESS`. In this case, you must calculate which of possibly many rows have changed and which to insert, update, or delete. If a particular row did not change or only one field changed, the DML calls will be minimal.

queryByExample

Unlike the `SELECT` operation, `queryByExample` does not require a selection criteria to be specified at design time. Instead, for each `invoke`, a selection criteria is inferred from an exemplar input XML record.

For instance, if the output `xmlRecord` is an employee record, and the input is a sample `xmlRecord` with `lastName = "Smith"`, then on execution, all employees with a last name of Smith are returned.

A subset of `queryByExample` is to query by primary key, which can be implemented by passing in sample XML records where only the primary key attributes are set.

Use `queryByExample` when you do not want to create a query using the visual query builder, and want the flexibility of allowing the input record to share the same XML schema as the output records.

The `queryByExample` operation is slightly less performant because a new `SELECT` needs to be prepared for each execution. This is because the attributes that are set in the example XML record can vary each time, and therefore the selection criteria varies.

Input `xmlRecord`:

```
<Employee>
  <id/>
  <lastName>Smith</lastName>
</Employee>
```

Output `xmlRecord`:

```
<EmployeeCollection>
  <Employee>
    <id>5</id>
    <lastName>Smith</lastName>
    ....
  </Employee>
  <Employee>
    <id>456</id>
    <lastName>Smith</lastName>
    ....
  </Employee>
</EmployeeCollection>
```

Use Cases for Outbound Invoke Operations

Outbound invoke operations are demonstrated in the following tutorial files:

- `Insert`
- `Update`
- `Delete`
- `Merge`
- `SelectAll`
- `SelectAllByTitle`
- `PureSQLSelect`
- `QueryByExample`

For these files, go to

`Oracle_Home\bpel\samples\tutorials\122.DBAdapter`

Note: When defining a DB outbound adapter service, ensure that the JNDI value you give to the service is declared in

`j2ee\home\application-deployments\default\DBAdapter\oc4j-ra.xml`

The outbound adapter service will use the `xDataSource` definition defined in `oc4j-ra.xml` to lookup the datasource. If the JNDI value is not defined in `oc4j-ra.xml`, then the db adapter would use unmanaged datasource which does not support XA transaction.

Use Cases For Pure SQL

A new option in 10.1.3.1 enables you to specify any arbitrary SQL string, and an xml representing the inputs and outputs to the SQL is generated. Pure SQL operations are demonstrated in the following tutorial files:

- `UpdateAll`
- `SelectCount`
- `SelectGroupBy`
- `SelectStar`

For these files, go to

`Oracle_Home\bpel\samples\tutorials\122.DBAdapter`

Advanced Use Cases for Outbound Invoke Operations

Advanced outbound invoke operations are demonstrated in the following tutorial files:

- `InsertWithClobs`
- `XAInsert`
- `NativeSequencingInsert`

For these files, go to

`Oracle_Home\bpel\samples\tutorials\122.DBAdapter\advanced\dmlInvoke`

4.2.2.2 Polling Strategies

The inbound receive enables you to listen to and detect events and changes in the database, which in turn can be the initiators of a business process. This is not a one-time action, but rather an activation. A polling thread is started, which polls a database table for new rows or events.

Whenever a new row is inserted into the `MOVIES` table, the polling operation raises it to Oracle BPEL Process Manager. The stratagem is to poll every record once. The initial `SELECT` has to be repeated over time, to receive the rows that exist at the start and all new rows as they are inserted over time. However, a new row once read is not likely to be deleted, and therefore can possibly be read repeatedly with each polling.

The various ways to poll for events—called polling strategies, also known as after-read strategies or publish strategies—range from simple and intrusive to sophisticated and nonintrusive. Each strategy employs a different solution for the problem of what to do after reading a row or event so as not to pick it up again in the next polling interval. The simplest (and most intrusive) solution is to delete the row so that you do not query it again.

Note: When you attempt to insert multiple records to a database, the `DBWriteInteractionSpec Execute Failed` Exception is thrown. The following is the solution to overcome this exception.

The number of retries can be tuned through the following `bpel/system/config/collaxa-config.xml` setting:

```
<property id="nonFatalConnectionMaxRetry">
  <name>Non fatal connection retry limit</name>
  <value>2</value>
  <comment>
    <![CDATA[The maximum number of times a non-fatal connection
    error can be retried before failing. Non-fatal connections errors
    may be thrown when the dehydration store is a RAC installation and
    the node the application server is pointing to is shutdown. The
    engine will resubmit messages that failed as a result of the
    connection error.
    <p/>
    The default value is 2.
    ]]>
  </comment>
</property>
```

Increase this value to a really high value to overcome this exception.

This section discusses the following polling strategies and factors to help you determine which strategy to employ for a particular situation:

- [Physical Delete](#)
- [Logical Delete](#)
- [Sequencing Table: Last-Read Id](#)
- [Sequencing Table: Last Updated](#)
- [Control Tables](#)

Physical Delete

The physical delete polling strategy polls the database table for records and deletes them after processing. This strategy can be used to capture events related to `INSERT` operations and cannot capture database events related to `DELETE` and `UPDATE` operations on the parent table. This strategy cannot be used to poll child table events. This strategy allows multiple adapter instances to go against the same source table. There is zero data replication.

Preconditions: You must have deletion privileges on the parent and associated child tables to use the delete polling strategy. [Table 4-5](#) describes the requirements for using the delete polling strategy.

Table 4-5 Delete Polling Strategy Preconditions

Requirements Met	Conflicts with
Poll for inserts	No delete on source
Shallow delete	No updates on source
Cascading delete	Poll for updates
Minimal SQL	Poll for deletes

Table 4–5 (Cont.) Delete Polling Strategy Preconditions

Requirements Met	Conflicts with
Zero data replication	Poll for child updates
Default	--
Allows raw SQL	--
Concurrent polling	--

Note: In *Shallow delete* and *Cascading delete*, delete can be configured to delete the top-level row, to cascade all, or to cascade on a case-by-case basis.

Concurrent polling can be configured for both delete and logical delete polling strategies.

Configuration: You can configure the delete polling strategy to delete the top-level row, to cascade all, or to cascade on a case-by-case basis. This enables deleting only the parent rows and not the child rows, cascaded deletes, and optional cascaded deletes, determined on a case-by-case basis. You can configure the polling interval for performing an event publish at design time.

Delete Cascade Policy: The optional advanced configuration is to specify the cascade policy of the DELETE. For instance, after polling for an employee with an address and many phone numbers, the phone numbers are deleted because they are privately owned (default for one-to-many), but not the address (default for one-to-one). This can be altered by configuring `toplink_mappings.xml`, as in the following example:

```
<database-mapping>
  <attribute-name>orders</attribute-name>
  <reference-class>taxonomy.Order</reference-class>
  <is-private-owned>true</is-private-owned>
```

You can also configure the activation itself to delete only the top level (master row), or to delete everything.

A receive operation appears in an inbound WSDL as:

```
<operation name="receive">
  <jca:operation
    ActivationSpec="oracle.tip.adapter.db.DBActivationSpec"
    ...
    PollingStrategyName="DeletePollingStrategy"
    DeleteDetailRows="true"
```

Logical Delete

The logical delete polling strategy involves updating a special field on each row processed, and updating the WHERE clause at run time to filter out processed rows. It mimics logical delete, wherein applications rows are rarely deleted but instead a status column `isDeleted` is set to true. The status column and the read value must be provided, but the modified WHERE clause and the post-read update are handled automatically by the database adapter.

Preconditions: You must have the logical delete privilege or a one-time alter schema (add column) privilege on the source table. [Table 4–6](#) describes the requirements for using the logical delete polling strategy.

Table 4–6 Logical Delete Polling Strategy Preconditions

Requirements Met	Conflicts With
Poll for inserts	No updates on source
No delete on source	Poll for deletes
Minimal SQL	--
Zero data replication	--
Minimal configuration	--
Allows raw SQL	--
Poll for updates	--
Poll for child updates	--
Concurrent polling	--

Note: The requirements of the following are met, as follows:

- **Poll for updates:** By adding a trigger
 - **Poll for child updates:** By adding a trigger
 - **Concurrent polling:** By specifying additional mark unread and reserved values.
-

Configuration: The logical delete polling strategy requires minimal configuration. You must specify the mark read column, and the value that indicates a processed record.

A receive operation appears in an inbound WSDL as:

```
<operation name="receive">
  <jca:operation
    ActivationSpec="oracle.tip.adapter.db.DBActivationSpec"
    ...
    PollingStrategyName="LogicalDeletePollingStrategy"
    MarkReadField="STATUS"
    MarkReadValue="PROCESSED"
```

Given the configuration for logical delete, the database adapter appends the following WHERE clause to every polling query:

```
AND (STATUS IS NULL) OR (STATUS <> 'PROCESSED')
```

Database Configuration: A status column on the table being polled must exist. If it does not exist already, you can add one to an existing table.

Support for Polling for Updates: Given that rows are not deleted with each read, it is possible to repetitively read a row multiple times. You should add a trigger to reset the mark read field whenever a record is changed, as follows:

```
create trigger Employee_modified
before update on Employee
for each row
begin
  :new.STATUS := 'MODIFIED';
end;
```

Support for Concurrent Access Polling: Just as a single instance should never process an event more than once, the same applies to a collection of instances. Therefore, before processing a record, an instance needs to reserve that record with a unique value. Again, the status column can be used:

```
<operation name="receive">
  <jca:operation
    ActivationSpec="oracle.tip.adapter.db.DBActivationSpec"
    ...
    PollingStrategyName="LogicalDeletePollingStrategy"
    MarkReadField="STATUS"
    MarkUnreadValue="UNPROCESSED"
    MarkReservedValue="RESERVED-1"
    MarkReadValue="PROCESSED"
```

The polling query instead looks like the following:

```
Update EMPLOYE set STATUS = 'RESERVED-1' where (CRITERIA) AND (STATUS =
'UNPROCESSED');
```

```
Select ... from EMPLOYEE where (CRITERIA) AND (STATUS = 'RESERVED-1');
```

The after-read UPDATE is faster because it can update all:

```
Update EMPLOYEE set STATUS = 'PROCESSED' where (CRITERIA) AND (STATUS =
'RESERVED-1');
```

Sequencing Table: Last-Read Id

This polling strategy involves using a helper table to remember a sequence value. The source table is not modified; instead, rows that have been read in a separate helper table are recorded. A sequence value of 1000, for example, means that every record with a sequence less than that value has already been processed. Because many tables have some counter field that is always increasing and maintained by triggers or the application, this strategy can often be used for noninvasive polling. No fields on the processed row ever need to be modified by the database adapter.

Native sequencing with a preallocation size of 1 can ensure that rows are inserted with primary keys that are always increasing over time.

This strategy is also called a nondestructive delete because no updates are made to the source rows, and a sequencing strategy such as the sequence field can be used to order the rows in a sequence for processing. When the rows are ordered in a line, the database adapter knows which rows are processed and which are not with a single unit of information.

Preconditions: You must have a sequencing table or create table privilege on the source schema. The source table has a column that is monotonically increasing with every INSERT (an Oracle native sequenced primary key) or UPDATE (the last-modified timestamp). [Table 4-7](#) describes the requirements for using the sequencing polling strategy.

Table 4-7 Sequencing Polling Strategy Preconditions

Requirements Met	Conflicts With
Poll for inserts	Poll for deletes
Poll for updates	Allows raw SQL
No delete on source	Concurrent polling
No updates on source	Poll for child updates

Table 4–7 (Cont.) Sequencing Polling Strategy Preconditions

Requirements Met	Conflicts With
One extra SQL select	--
Zero data replication	--
Moderate configuration	--

Configuration: A separate helper table must be defined. On the source table, you must specify which column is ever increasing.

```
<operation name="receive">
<jca:operation
ActivationSpec="oracle.tip.adapter.db.DBActivationSpec"
...
PollingStrategyName="SequencingPollingStrategy"
SequencingFieldName="MODIFIED_DATE"
SequencingFieldType="java.sql.Date"
SequencingTableNameFieldValue="EMPLOYEE"
SequencingTableName="SEQUENCING_HELPER"
SequencingTableNameFieldName="TABLE_NAME"
SequencingTableValueFieldName="LAST_READ_DATE"
```

The sequencing field type can be excluded if it is actually a number.

Database Configuration: A sequencing table must be configured once for a given database. Multiple processes can share the same table. Given the `ActivationSpec` specified in the preceding example, the `CREATE TABLE` command looks as follows:

```
CREATE TABLE SEQUENCING_HELPER
(
TABLE_NAME VARCHAR2(32) NOT NULL,
LAST_READ_DATE DATE
)
;
```

Polling for Updates: In the preceding example, the polling is for new objects or updates, because every time an object is changed, the modified time is updated.

A sample trigger to set the modified time on every insert or update is as follows:

```
create trigger Employee_modified
before insert or update on Employee
for each row
begin
:new.modified_date := sysdate;
end;
```

Using a Sequence Number: A sequence number can be used for either insert or update polling. Native sequencing returns monotonically increasing primary keys, as long as an increment by 1 is used. You can also use the sequence number of a materialized view log.

Sequencing Table: Last Updated

This polling strategy involves using a helper table to remember a `last_updated` value. A `last_updated` value of 2005-01-01 12:45:01 000, for example, means that every record last updated at that time or earlier has already been processed. Because many tables have rows with a `last_updated` or `creation_time` column maintained by triggers or the application, this strategy can often be used for

noninvasive polling. No fields on the processed row ever need to be modified by the database adapter.

This strategy is also called a nondestructive delete because no updates are made to the source rows, and a sequencing strategy such as the `last_updated` field can be used to order the rows in a sequence for processing. When the rows are ordered in a line, the database adapter knows which rows are processed and which are not with a single unit of information.

See [Sequencing Table: Last-Read Id](#) for information about preconditions and configuration.

Control Tables

The control table polling strategy involves using a control table to store the primary key of every row that has yet to be processed. With a natural join between the control table and the source table (by primary key), polling against the control table is practically the same as polling against the source table directly. However, an extra layer of indirection allows the following:

- Destructive polling strategies such as the delete polling strategy can be applied to rows in the control table alone, while shielding any rows in the source table.
- Only rows that are meant to be processed have their primary key appear in the control table. Information that is not in the rows themselves can be used to control which rows to process (a good `WHERE` clause may not be enough).
- The entire row is not copied to a control table, and any structure under the source table, such as detail rows, can also be raised without copying.

Streams and materialized view logs make good control tables.

Preconditions: You must have create/alter triggers privilege on the source table. [Table 4–8](#) describes the requirements for using the control table polling strategy.

Table 4–8 Control Table Polling Strategy Preconditions

Requirements Met	Conflicts With
Poll for inserts	Advanced configuration: the native XML from the database will have control header, and triggers are required.
Poll for updates	--
Poll for deletes	--
Poll for child updates	Minimal data replication (primary keys are stored in control table)
No delete on source	--
No updates on source	--
No extra SQL selects	--
Concurrent polling	--
Allows raw SQL	--
Auditing	--

Using triggers, whenever a row is modified, an entry is added to a control table, containing the name of the master table, and the primary keys. At design time, the control table is defined to be the root table, with a one-to-one mapping to the master table, based on the matching primary keys. The control table can contain extra control information, such as a timestamp, and operation type (`INSERT`, `UPDATE`, and so on).

The delete polling strategy is useful with this setup. It is important to keep the control table small, and if the option `shouldDeleteDetailRows="false"` is used, then only the control rows are deleted, giving you a nondestructive delete (the `DELETE` is not cascaded to the real tables).

It is possible to reuse the same control table for multiple master tables. In `TopLink`, you can map the same table to multiple descriptors by mapping the control table as one abstract class with multiple children. Each child has a unique one-to-one mapping to a different master table. The advantage of this approach is that you can specify for each child a class indicator field and value so that you do not need an explicit `WHERE` clause for each polling query.

Some sample triggers follow for polling for changes both to a department table and any of its child employee rows:

```
CREATE OR REPLACE TRIGGER EVENT_ON_DEPT
  AFTER INSERT OR UPDATE ON DEPARTMENT
  REFERENCING NEW AS newRow
  FOR EACH ROW
  DECLARE X NUMBER;
BEGIN
  SELECT COUNT(*) INTO X FROM DEPT_CONTROL WHERE (DEPTNO = :newRow.DEPTNO);
  IF X = 0 then
    insert into DEPT_CONTROL values (:newRow.DEPTNO);
  END IF;
END;

CREATE OR REPLACE TRIGGER EVENT_ON_EMPLOYEE
  AFTER INSERT OR UPDATE ON EMPLOYEE
  REFERENCING OLD AS oldRow NEW AS newRow
  FOR EACH ROW
  DECLARE X NUMBER;
BEGIN
  SELECT COUNT(*) INTO X FROM DEPT_CONTROL WHERE (DEPTNO = :newRow.DEPTNO);
  IF X = 0 then
    INSERT INTO DEPT_CONTROL VALUES (:newRow.DEPTNO);
  END IF;
  IF (:oldRow.DEPTNO <> :newRow.DEPTNO) THEN
    SELECT COUNT(*) INTO X FROM DEPT_CONTROL WHERE (DEPTNO = :oldRow.DEPTNO);
    IF (X = 0) THEN
      INSERT INTO DEPT_CONTROL VALUES (:oldRow.DEPTNO);
    END IF;
  END IF;
END;
```

Use Cases for Polling Strategies

Polling strategies are demonstrated in the following tutorials:

- `PollingLogicalDeleteStrategy`
- `PollingLastUpdatedStrategy`
- `PollingLastReadIdStrategy`
- `PollingControlTableStrategy`
- `MasterDetail` (for physical delete polling strategy)

For these files, go to

`Oracle_Home\bpel\samples\tutorials\122.DBAdapter`

Advanced Use Cases for Polling Strategies

Advanced polling strategies are demonstrated in the following tutorials:

- DistributedPolling
- PollingExternalSequencing
- PollingFileSequencingStrategy
- PollingForChildUpdates
- PollingNoAfterReadStrategy
- PollingOracleSCNStrategy
- PollingPureSQLOtherTableInsert
- PollingPureSQLSysdateLogicalDelete
- PollingWithParameters

For these files, go to

`Oracle_Home\bpel\samples\tutorials\122.DBAdapter\advanced\polling`

4.3 Database Adapter Use Case for Oracle BPEL Process Manager

To use the database adapter demonstrated in the 122.DBAdapter tutorial, go to

`Oracle_Home\bpel\samples\tutorials\122.DBAdapter`

Table 4–9 shows the database adapter samples that are provided with Oracle BPEL Process Manager.

Table 4–9 Database Adapter Use Cases

Tutorial Name	Description
Delete	Illustrates the outbound delete operation of the database adapter. An XML record is passed to the operation and the row in the database with the same primary key is deleted.
File2StoredProcedure	Describes a simple scenario in which the file adapter is used to provide instance XML to a stored procedure, ADDEMPLOYEES, which is then executed. The instance XML provides a value for the parameter of the stored procedure. The ADDEMPLOYEES procedure must be installed in an Oracle database (not Oracle Lite).
File2Table	Illustrates the use of an input a native (CSV) data file defined in a custom format. The input file is a purchase order, which the file adapter processes and publishes as an XML message to the File2Table BPEL process. The message is transformed to another purchase order format and routed to an invoke activity.
Insert	Illustrates the outbound insert operation of the database adapter. An XML record is passed to the operation and inserted into the database as relational data. (In JDeveloper BPEL Designer, Merge (Insert or Update) is provided.)
InsertWithCatch	Illustrates the extra steps (based on the Insert tutorial) needed to add fault handling to your BPEL process.

Table 4–9 (Cont.) Database Adapter Use Cases

Tutorial Name	Description
JPublisherWrapper	Illustrates a workaround for using PL/SQL RECORD types. JPublisher is used to create a corresponding OBJECT type whose attributes match the fields of the RECORD, and conversion APIs that convert from RECORD to OBJECT and vice versa. JPublisher also generates a wrapper procedure (or function) that accepts the OBJECT and invokes the underlying method using the conversion APIs in both directions. The invoked methods must be installed in an in an Oracle database (not Oracle Lite).
MasterDetail	Illustrates how to migrate data from one set of tables to another. The sample uses the database adapter to read data from one set of tables, process the data, and write it in to another set of database tables using the adapter.
Merge	Illustrates the outbound merge operation of the database adapter. An XML record is passed to the operation and a corresponding row in the database is either inserted or updated.
PollingControlTableStrategy	Illustrates an inbound polling operation to poll XML instances from the MOVIES table. When a new row is inserted into the MOVIES table, the polling operation raises it to Oracle BPEL Process Manager. This strategy uses a control table to store the primary key of every row that has not yet been processed. With a natural join between the control table and the source table (by primary key), polling against the control table is practically the same as polling against the source table directly.
PollingLastReadIdStrategy	Illustrates an inbound polling operation to poll XML instances from the MOVIES table. Whenever a new row is inserted into the MOVIES table, the polling operation raises it to Oracle BPEL Process Manager. This strategy uses a helper table to remember a sequence value.
PollingLastUpdatedStrategy	Illustrates an inbound polling operation to poll XML instances from the MOVIES table. Whenever a new row is inserted into the MOVIES table, the polling operation raises it to Oracle BPEL Process Manager. This strategy involves using a helper table to remember a last_updated value.
PollingLogicalDeleteStrategy	Illustrates an inbound polling operation to poll XML instances from the MOVIES table. Whenever a new row is inserted into the MOVIES table, the polling operation raises it to Oracle BPEL Process Manager. This strategy involves updating a special field on each row processed, and updating the WHERE clause at run time to filter out processed rows.
PureSQLPolling	Illustrates how to poll a table based on a date field.
PureSQLSelect	Illustrates how to bypass the JDeveloper BPEL Designer WHERE-clause builder to specify arbitrarily complex SQL strings for SELECT operations.
QueryByExample	Illustrates the outbound queryByExample operation of the database adapter. A SELECT SQL query is built dynamically based on fields set in an example XML record, and any matching records are returned.
ResultSetConverter	Illustrates a workaround for using REF CURSORS. The solution involves the use of a Java stored procedure to convert the corresponding java.sql.ResultSet into a collection (either VARRAY or NESTED TABLE) of OBJECTs.
SelectAll	Illustrates the outbound SelectAll operation of the database adapter. With no WHERE clause, all rows in the MOVIES table are returned as XML.
SelectAllByTitle	Illustrates the outbound SelectAllByTitle operation of the database adapter. The row in the MOVIES table with the selected title is returned as XML.
Update	Illustrates the outbound Update operation of the database adapter. An XML record is passed to the operation and the row in the database with the same primary key is updated. (In JDeveloper BPEL Designer, Merge (Insert or Update) is provided.)

See [Table 4–1](#) for the structure of the `MOVIES` table, which is used for many of the use cases. The `readme.txt` files that are included with most of the samples provide instructions.

This section comprises the following two use cases:

- [Use Case: One](#)
- [Use Case: Two](#)

4.3.1 Use Case: One

This use case describes how by using the Adapter Configuration Wizard, you can import tables from the database, specify relationships spanning multiple tables, generate corresponding XML schema definitions, and create services to expose the necessary SQL or database operations. These services are consumed to define partner links that are used in the BPEL process. You use the Adapter Configuration to both create and edit adapter services.

This section contains the following topics:

- [Starting the Adapter Configuration Wizard](#)
- [Connecting to a Database](#)
- [Selecting the Operation Type](#)
- [Selecting and Importing Tables](#)
- [Defining Primary Keys](#)
- [Creating Relationships](#)
- [Creating the Object Filter](#)
- [Defining a WHERE Clause](#)
- [Choosing an After-Read Strategy](#)
- [Internal Processes at Design Time](#)

4.3.2 Starting the Adapter Configuration Wizard

After you create a BPEL project in JDeveloper BPEL Designer, you can start defining a database adapter. If you lose focus on the window, use `alt+tab` to get it back.

To launch the Adapter Configuration Wizard:

1. Ensure that **All Process Activities** is selected in the drop-down list of the **Component Palette** section.
2. Drag and drop a **PartnerLink** activity onto the right side of the designer window.
3. Enter a name in the Create Partner Link window.
4. Click the **Define Adapter Service** icon to start the Adapter Configuration Wizard.



5. Click **Next** on the Welcome window.
6. Select **Database Adapter** for the **Adapter Type**, and then click **Next**.

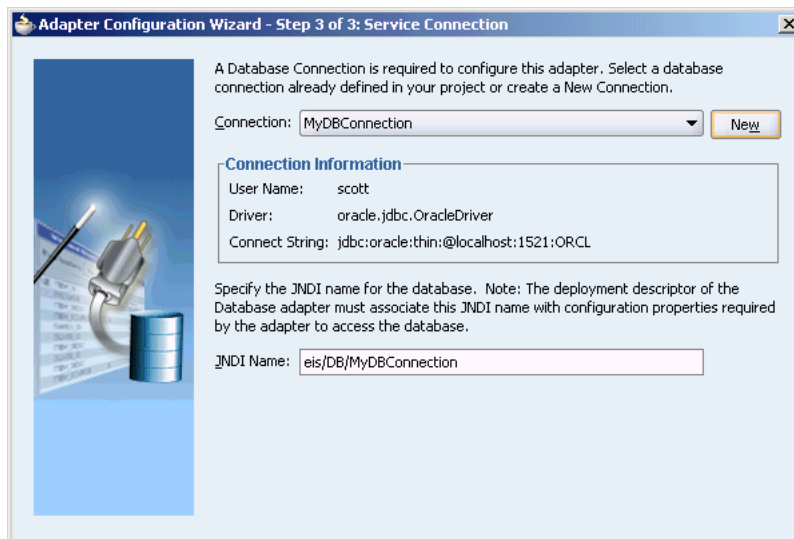
7. In the Service Name window, enter the service name, and a description for it.
The description is optional.
 8. Click **Next**. The Service Connection wizard is displayed.
- See [Connecting to a Database](#) to continue using the wizard.

4.3.3 Connecting to a Database

Figure 4–2 shows where you select the database connection that you are using with the service. This is the database from which you import tables to configure the service.

You can provide a Java Naming and Directory Interface (JNDI) name to identify the database connection, or use the default name that is provided. The JNDI name acts as a placeholder for the connection used when your service is deployed to Oracle BPEL Server. This enables you to use different databases for development and production. The Adapter Configuration Wizard captures the design-time connection in the generated WSDL as well, to serve as a fallback in case the run-time lookup fails.

Figure 4–2 Adapter Configuration Wizard: Service Connection



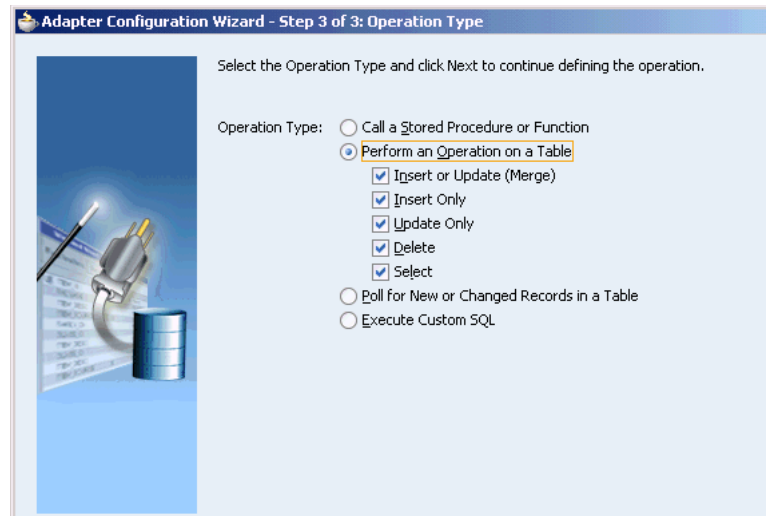
Note the following:

- In production environments, it is recommended that you add the JNDI entry to the adapter deployment descriptor (`oc4j-ra.xml`). This way, the database adapter is more performant by working in a managed mode. In a nonmanaged mode, the database adapter uses the design-time connection information.
- When you click **Next**, a connection to the database is attempted. If a connection cannot be made, you are not able to proceed to the next window, even if you are editing an existing partner link.

See [Selecting the Operation Type](#) to continue using the wizard.

4.3.4 Selecting the Operation Type

Figure 4–3 shows where you indicate the type of operation you want to configure for this service.

Figure 4–3 Adapter Configuration Wizard: Operation Type

The follow operation types are available:

- **Call a Stored Procedure or Function**

Select this option if you want the service to execute a stored procedure or function. See [Stored Procedure and Function Support](#) for more information.

- **Perform an Operation on a Table**

Select this option for outbound operations. You can select **Insert or Update**, **Insert Only**, **Update Only**, **Delete**, **Select**, or any combination of the five. These operations loosely translate to SQL INSERT, UPDATE, DELETE, and SELECT operations. See [DML Operations](#) for more information.

Note the following:

- The operations **merge**, **insert**, **update**, and **write** are created from selecting **Insert or Update**.
- The preceding Invoke window shows the MergeService service name as part of the **Select** operation, that is, **MergeServiceSelect**.
- The **queryByExample** operation appears in every WSDL.
- If the **Operation** list is initially blank, reselect the partner link and click the **Operation** list again.

Note: The operation `Update Only` sometimes performs inserts/deletes for child records. That is, an update to Master could involve a new or deleted detail. So if the input to update contains only one record, then the other records in the table will be deleted.

- **Poll for New or Changed Records in a Table**

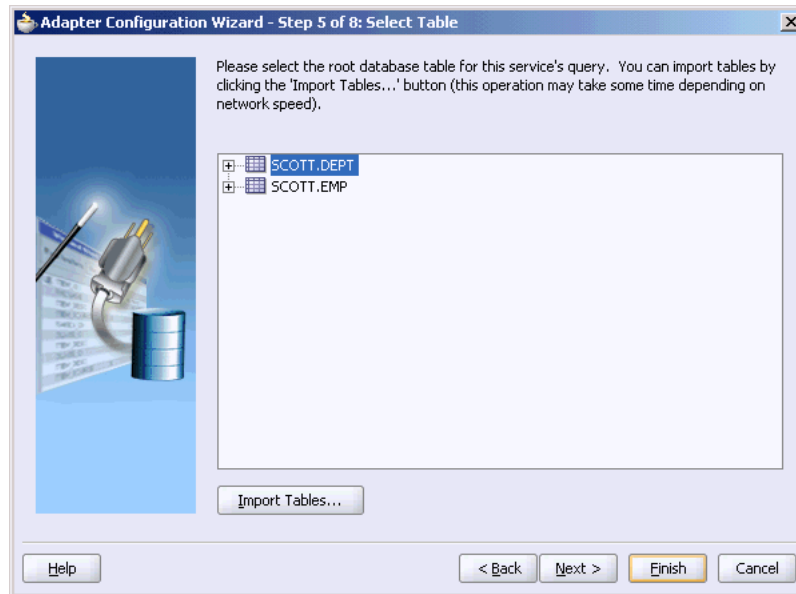
Select this option for an inbound operation (that is, an operation that is associated with a **Receive** activity). This operation type polls a specified table and returns for processing any new rows that are added. You can also specify the polling frequency. See [Polling Strategies](#) for more information.

See [Selecting and Importing Tables](#) to continue using the wizard.

4.3.5 Selecting and Importing Tables

Figure 4–4 shows where you select the root database table for your operation. If you are using multiple, related tables, then this is the highest-level table (or highest parent table) in the relationship tree.

Figure 4–4 Adapter Configuration Wizard: Select Table



This window shows all the tables that have been previously imported in the JDeveloper BPEL Designer project (including tables that were imported for other partner links). This enables you to reuse configured table definitions across multiple partner links in a given BPEL project. These are the generated TopLink descriptors.

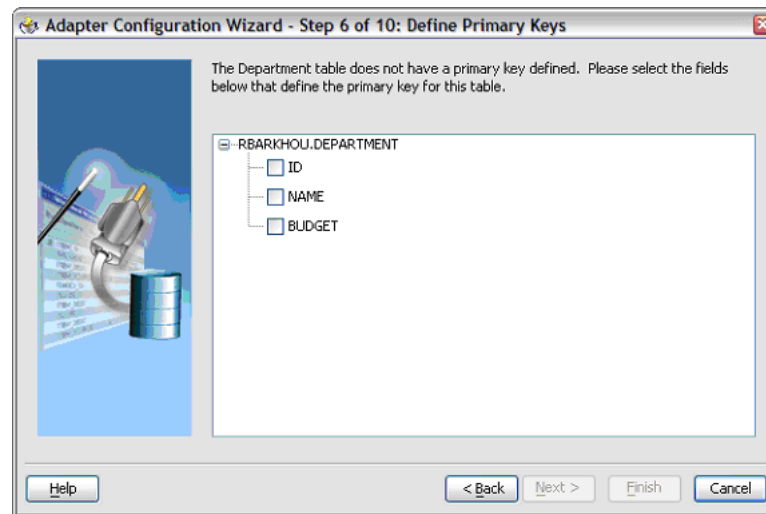
If the root database table you want to use for this operation has not been previously imported, you can click **Import Tables....** If you want to reimport a table (if the table structure has changed on the database, for example), import it again. You can then reimport the table and overwrite the previously configured table definition.

Note: If you reimport a table, you lose any custom relationships you may have defined on that table, as well as any custom WHERE clauses (if the table being imported was the root table).

See [Defining Primary Keys](#) to continue using the wizard.

4.3.6 Defining Primary Keys

If any of the tables you have imported do not have primary keys defined on the database, you are prompted to provide a primary key for each one, as shown in [Figure 4–5](#). You must specify a primary key for all imported tables. You can select multiple fields if you need to specify a multipart primary key.

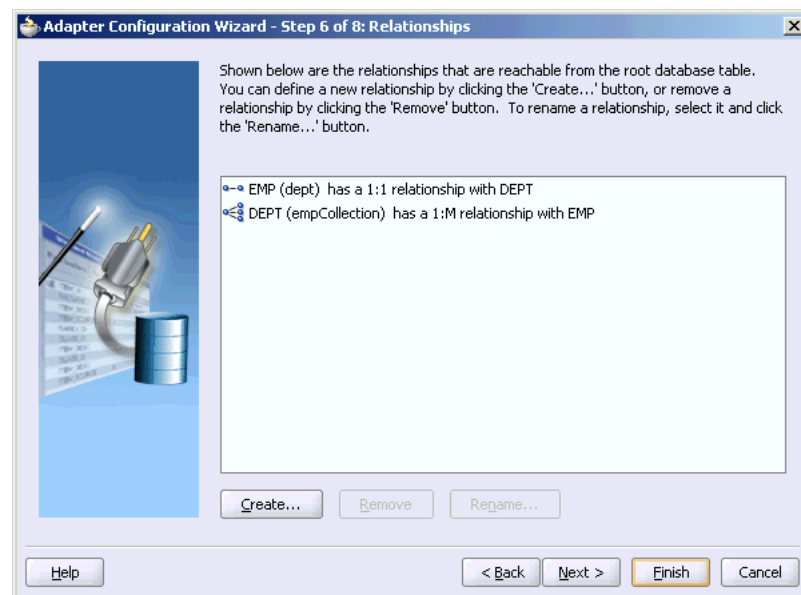
Figure 4–5 Adapter Configuration Wizard: Define Primary Keys

The primary key that you specify here is recorded on the offline database table and is not persisted back to the database schema; the database schema is left untouched.

See [Creating Relationships](#) to continue using the wizard.

4.3.7 Creating Relationships

[Figure 4–6](#) shows the relationships defined on the root database table and any other related tables. You can click **Create Relationships...** to create a new relationship between two tables, or **Remove Relationship** to remove it.

Figure 4–6 Adapter Configuration Wizard: Relationships

Note the following regarding creating relationships:

- If foreign key constraints between tables already exist on the database, then two relationships are created automatically when you import the tables: a one-to-one (1:1) from the source table (the table containing the foreign key constraints) to the

target table, as well as a one-to-many (1:M) from the target table to the source table.

- As Figure 4-6 shows, you see only the relationships that are reachable from the root database table. If, after removing a relationship, other relationships are no longer reachable from the root table, then they are not shown in the Relationships window. Consider the following set of relationships:

```
A --1:1--> B --1:1--> C --1:M--> D --1:1--> E --1:M--> F
      (1)      (2)      (3)      (4)      (5)
```

If you remove relationship 3, then you see only:

```
A --1:1--> B
```

```
B --1:1--> C
```

If you remove relationship 2, then you see only:

```
A --1:1--> B
```

If you remove relationship 1, you no longer see any relationships.

Figure 4-7 shows where you can create a new relationship.

Figure 4-7 Creating Relationships

Please specify the parent and child tables, relation type and relation name. You also must specify the foreign key / primary key associations in the table below.

Parent Table: SCOTT.DEPT

Child Table: SCOTT.EMP

☐ DEPT has a 1:1 Relationship with EMP
☐ DEPT has a 1:1 Relationship with EMP (Foreign Key on Child table)
☒ DEPT has a 1:M Relationship with EMP

DEPT	EMP
DEPTNO	DEPTNO

Relationship Name: managedemployee

Help OK Cancel

To create a new relationship:

1. Select the parent and child tables.
2. Select the mapping type (one-to-many, one-to-one, or one-to-one with the foreign key on the child table).
3. Associate the foreign key fields to the primary key fields.
4. Optionally name the relationship (a default name is generated).

Note: Only tables that are reachable from the root table can be selected as a parent.

4.3.7.1 What Happens When Relationships Are Created or Removed

When tables are initially imported into the wizard, a TopLink direct-to-field mapping corresponding to each field in the database is created. Consider the schemas shown in [Figure 4–8](#) and [Figure 4–9](#):

Figure 4–8 EMPLOYEE Schema

EMPLOYEE		
ID *	NAME	ADDR ID

Figure 4–9 ADDRESS Schema

ADDRESS		
ID *	ZIP	STREET

Immediately after importing these two tables, the following mappings in the Employee descriptor are created:

Employee:

- id (direct mapping to the ID field, for example, 151)
- name (direct mapping to the NAME field, for example, *Stephen King*)
- addrId (direct mapping to the ADDR_ID field, for example, 345)

When creating a relationship mapping, the direct-to-field mappings to the foreign key fields are removed and replaced with a single relationship (one-to-one, one-to-many) mapping. Therefore, after creating a one-to-one relationship between Employee and Address called *homeAddress*, the Employee descriptor looks like this:

Employee:

- id
- name
- homeAddress (one-to-one mapping to the ADDRESS table; this attribute now represents the entire Address object.)

When a relationship is removed, the direct mappings for the foreign keys are restored.

4.3.7.2 Different Types of One-to-One Mappings

The following ways of specifying one-to-one relationships are supported:

- The foreign keys exist on the parent table, as shown in [Figure 4–10](#) and [Figure 4–11](#).
- The foreign keys exist on the child table, as shown in [Figure 4–12](#) and [Figure 4–13](#).

Figure 4–10 Foreign Keys on the Parent Table EMPLOYEE

EMPLOYEE		
ID *	NAME	ADDR ID

Figure 4–11 Foreign Keys on the Parent Table ADDRESS

ADDRESS		
ID *	ZIP	STREET

Figure 4–12 Foreign Keys on the Child Table EMPLOYEE

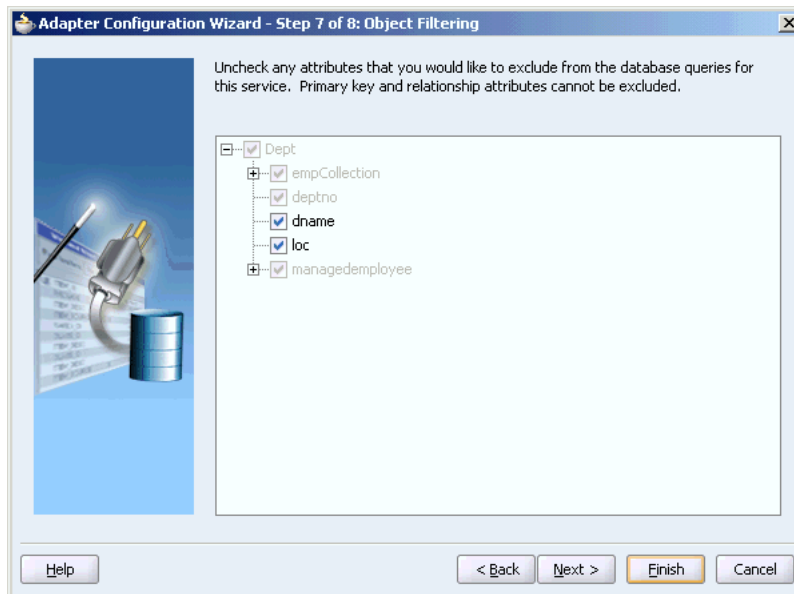
EMPLOYEE	
ID *	NAME

Figure 4–13 Foreign Keys on the Child Table ADDRESS

ADDRESS			
ID *	ZIP	STREET	EMP_ID

4.3.8 Creating the Object Filter

[Figure 4–14](#) shows the object filter that is created from the imported table definitions, including any relationships that you may have defined.

Figure 4–14 Adapter Configuration Wizard: Object Filtering

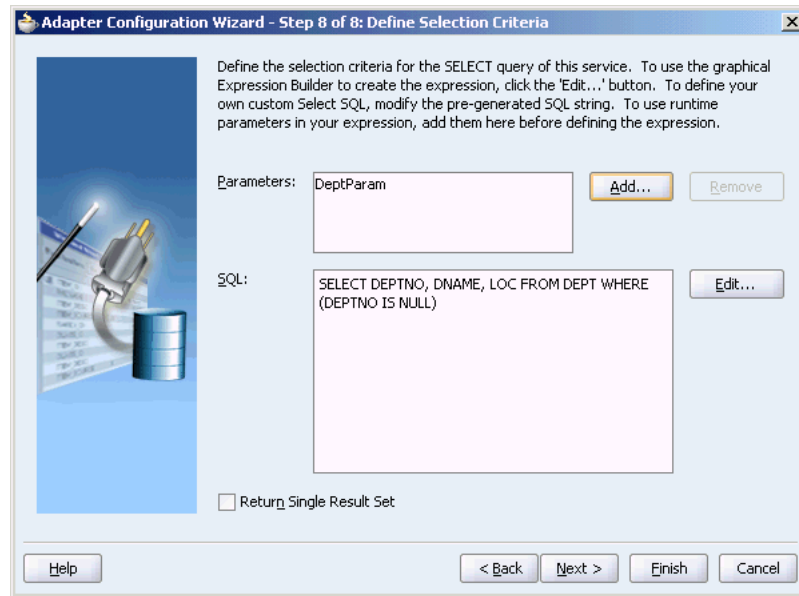
If your object filter contains self-relationships (for example, the employee-to-employee manager relationship), then you see these as loops in the tree. These loops are not present in the XSD. This is the descriptor object model, not the XSD.

See [Defining a WHERE Clause](#) to continue using the wizard.

4.3.9 Defining a WHERE Clause

If your service contains a `SELECT` query (that is, inbound polling services, or outbound services that contain a `SELECT`), then you can customize the `WHERE` clause of the `SELECT` statement.

[Figure 4–15](#) shows where you define a `WHERE` clause for an outbound service. For inbound services, you do not see the **Parameters** section.

Figure 4–15 Adapter Configuration Wizard: Define WHERE Clause

Note: The WHERE clause applies to SELECT operations only (that is, polling for new or changed records, or performing a SELECT operation on a table). It does not apply to INSERT, UPDATE, and DELETE operations.

The most basic expression in a WHERE clause can be one of the following three cases, depending on what the right-hand side (RHS) is:

1. EMP.ID = 123

In this case, the RHS is a literal value. This RHS is the **Literal** option shown in [Figure 4–16](#).

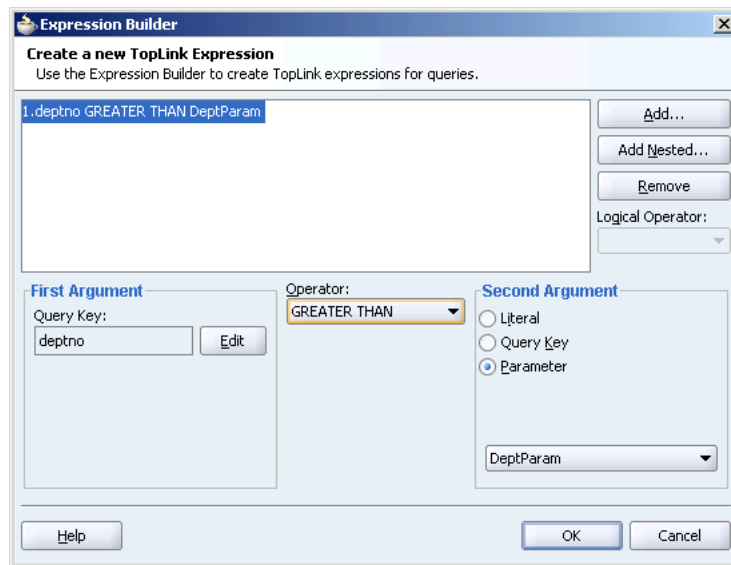
2. EMP.ADDR_ID = ADDR.ID

In this case, the RHS is another database field. This RHS is the **Query Key** option shown in [Figure 4–16](#).

3. EMP.ID = ?

In this case, the RHS value must be specified at run time. This is the **Parameter** option shown in [Figure 4–16](#).

You create the parameters that you need in the WHERE clause by clicking **Add** before you move on to build the WHERE clause. To build the WHERE clause, click **Edit...** to launch the Expression Builder, as shown in [Figure 4–16](#).

Figure 4–16 Expression Builder

See the following for more information:

- The TopLink page on OTN at
<http://www.oracle.com/technology/products/ias/toplink/index.html>
- TopLink documentation at
http://download.oracle.com/docs/cd/B14099_04/web.htm#toplink

This site contains documentation on configuring expressions using the XPath Expression Builder.

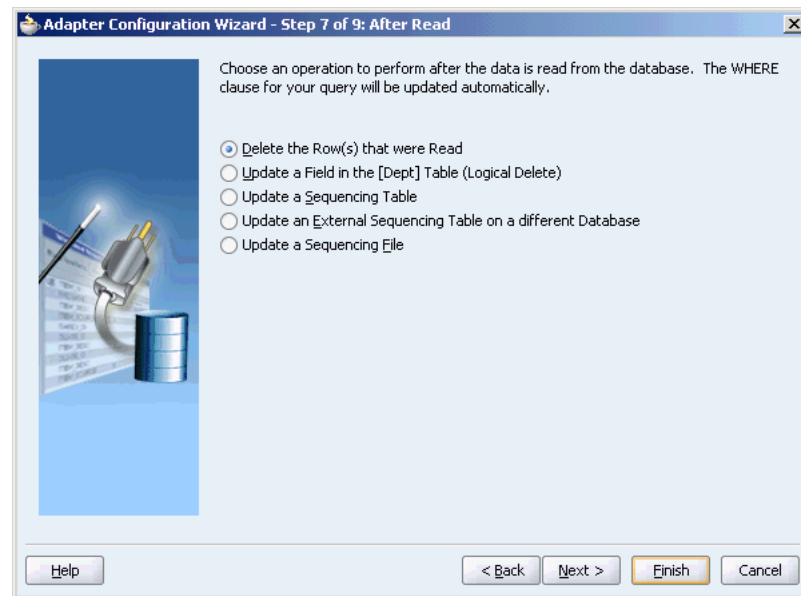
See [Choosing an After-Read Strategy](#) to continue using the wizard.

4.3.10 Choosing an After-Read Strategy

When configuring an inbound operation, you have the following options about what to do after a row or rows have been read:

- [Delete the Rows that Were Read](#)
- [Update a Field in the Table \(Logical Delete\)](#)
- [Update a Sequencing Table](#)

Figure 4–17 shows these options.

Figure 4–17 Adapter Configuration Wizard: After-Read Strategies

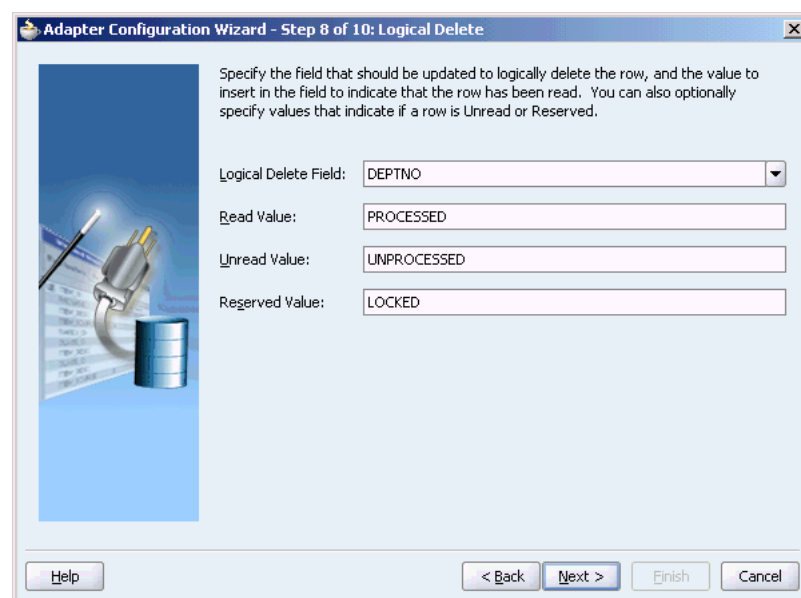
See [Polling Strategies](#) for more information.

4.3.10.1 Delete the Rows that Were Read

With this option, the rows are deleted from the database after they have been read and processed by the adapter service.

4.3.10.2 Update a Field in the Table (Logical Delete)

With this option, you update a field in the root database table to indicate that the rows have been read. The `WHERE` clause of the query is updated automatically after you complete the configuration, as shown in [Figure 4–18](#).

Figure 4–18 Adapter Configuration Wizard: Logical Delete

Using this approach, your database table looks something like [Figure 4–19](#).

Figure 4–19 Updating Fields in a Table

EMPLOYEE				
ID*	NAME	SALARY	...	STATUS
150	Dean Koontz	55000	...	PROCESSED
151	Stephen King	75000	...	
152	Patricia Cornwell	58000	...	UNPROCESSED
153	John Grisham	67500	...	PROCESSED
154	Michael Crichton	61250	...	LOCKED

Note the following:

- Rows 150 and 153 have been previously read and processed.
- At the next polling event, row 152 will be read and processed because it contains UNPROCESSED in the Status column. Because an explicit Unread Value was provided, row 151 will not be read.
- Row 154 has been flagged as LOCKED and will not be read. You can use this reserved value if your table is in use by other processes.

4.3.10.3 Update a Sequencing Table

With this option, you are keeping track of the last-read rows in a separate sequence table. [Figure 4–20](#) shows the information you provide. The WHERE clause of your query is updated automatically after you complete the configuration.

Figure 4–20 Adapter Configuration Wizard: Last Read IDs Table

Adapter Configuration Wizard - Step 8 of 10: Sequencing Table

Specify the table, name field, and value field to be updated when a row is read from the database. You must also specify the field in the Dept table that contains the sequential ID.

Sequencing Table: MY_SEQUENCE

Sequence Name Field: SEQ_NAME

Sequence Value Field: SEQ_VALUE

Sequenced ID Field: DEPTNO

Buttons: Help, < Back, Next >, Finish, Cancel

Using these settings, your sequence table looks something like [Figure 4–21](#).

Figure 4–21 Updating a Sequence Table

MY_SEQUENCE	
SEQ NAME	SEQ VALUE
EMPLOYEE	154

Whenever a row is read, this table is updated with the ID that was just read. Then when the next polling event occurs, it will search for rows that have an ID greater than the last-read ID (154).

Typical columns used are `event_id`, `transaction_id`, `scn` (system change number), `id`, or `last_updated`. These columns typically have (monotonically) increasing values, populated from a sequence number or `sysdate`.

4.3.11 Internal Processes at Design Time

This section describes happens internally at design time when you use the Adapter Configuration Wizard to configure the database adapter.

4.3.11.1 Importing Tables

When you import a table, the offline table support of JDeveloper BPEL Designer creates an offline snapshot of the database table. You can modify this offline version of the table (for example, you can add a foreign key constraint) without affecting the real database table. This creates a TopLink descriptor and associated Java source file for the table, and all the attributes in the descriptor are automapped to their corresponding database columns. The TopLink descriptor maps the Java class to the offline database table.

Most typical data columns are mapped as direct-to-field mappings, meaning that the value in the database column is directly mapped to the attribute. For example, a `SALARY` column in the database is mapped to a `salary` attribute in the object model, and that attribute contains the value of that column.

If foreign key constraints are already present in the imported tables, then relationship mappings are autogenerated between the tables. To cover as many scenarios as possible, two mappings are generated for every foreign key constraint encountered: a one-to-one mapping from the source table to the target table, and a one-to-many mapping in the opposite direction. After this is done, you are left with an TopLink Workbench project in your BPEL project.

Note: The Java classes that are created as part of the descriptor generation process are never actually deployed with your process or used at run time. They are present in the design time because TopLink Workbench is expecting each descriptor to be associated with a Java class. When your process is deployed, the mapping metadata is stored in `toplink_mappings.xml`.

When you have finished importing tables, you must select a root database table. In doing so, you are actually selecting which TopLink descriptor stores the autogenerated query.

4.3.11.2 Creating Relationships

When you create or remove a relationship, you are actually modifying the TopLink descriptor mappings. Creating a new relationship does the following:

- Creates a foreign key constraint in the offline database table
- Adds a one-to-one or one-to-many mapping to the descriptor
- Removes the direct-to-field mappings to the foreign key fields

Removing a relationship mapping does the following:

- Removes the one-to-one or one-to-many mapping from the descriptor
- Removes the foreign key constraint from the offline database table
- Adds direct-to-field mappings for each foreign key field involved in the relationship

4.3.11.3 Generating Design-Time Artifacts

The following files are generated:

- *service_name.wsdl*—contains the database adapter service definition
- *RootTable.xsd*—the XML type definition of the root object
- *toplink_mappings.xml*—contains the TopLink mapping metadata for your BPEL project. It is the only Toplink artifact that is deployed to the server.

4.3.12 Use Case: Two

This section explains an advanced use case, titled Polling File Sequencing Strategy. This use case is available at:

Oracle_
Home\bpel\samples\tutorials\122.DBAdapter\advanced\polling\PollingFileSequencingStrategy

This scenario aims at performing a completely non-intrusive polling. In this example, the sequencing strategies need to only remember a single piece of information, the last read id/sequence value. That one piece of information can either be stored on a table, on the same database or a different one, or that single piece of information can be stored simply in a file.

This section comprises the following sections:

- [Prerequisites](#)
- [Creating a New BPEL Project](#)
- [Starting the Adapter Configuration Wizard](#)

4.3.12.1 Prerequisites

Ensure that you run the sql scripts, *setup.sql* and *reset.sql* available at:

Oracle_Home\bpel\samples\tutorials\122.DBAdapter\sql

4.3.12.2 Creating a New BPEL Project

The following are the steps to create a new BPEL project:

1. Open Oracle JDeveloper.
2. From the **File** menu, select **New**.

The New Gallery dialog box is displayed.

3. Select **All Technologies** from the **Filter By** box. This displays a list of available categories.
4. Expand the **General** node, and then select **Projects**.
5. Select **BPEL Project** from the Items group.
6. Click **OK**.

The Create BPEL Project dialog box is displayed.

7. Click **OK**.

You have created a new BPEL project.

4.3.12.3 Starting the Adapter Configuration Wizard

After you create a BPEL project in Oracle JDeveloper, you can start defining a database adapter. To define a database adapter:

1. Drag and drop **Database Adapter** from the component Palette onto the designer window.

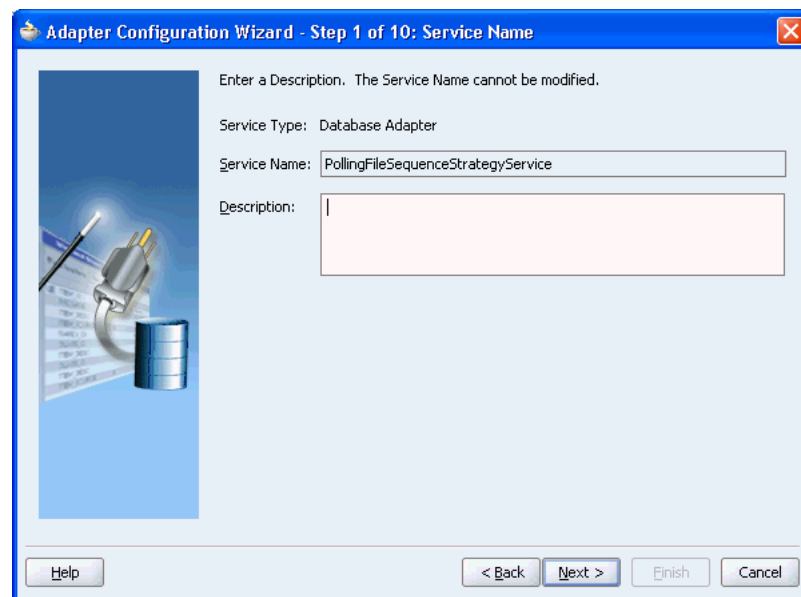
The Adapter Configuration Wizards-Welcome screen is displayed.

2. Click **Next**.

The Service Name dialog box is displayed.

3. Enter a service name as shown in [Figure 4-22](#). The Description field is optional.

Figure 4-22 *Entering the Service Name*



4. Click **Next**.

The Service Connection dialog box is displayed.

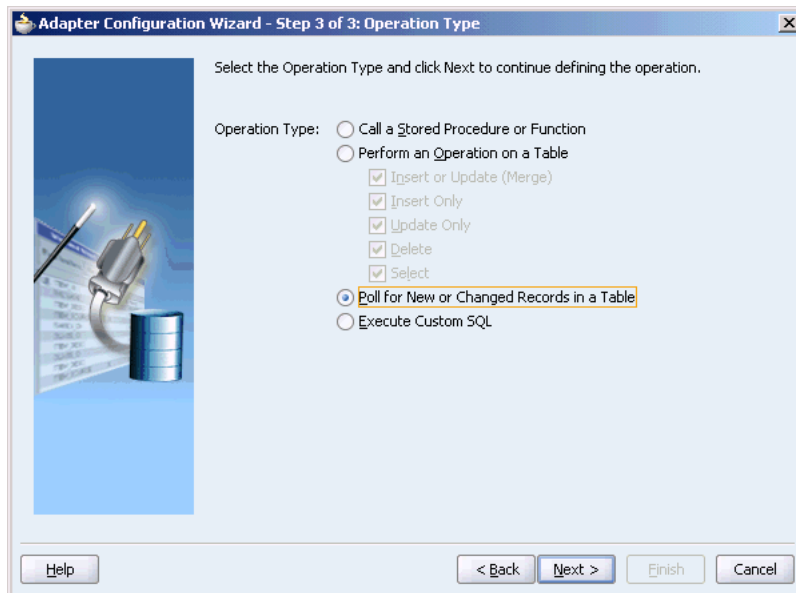
5. Select a database connection already defined in the project or create a new connection.

6. Click **Next**.

The Operation Type dialog box is displayed.

7. Select **Poll for New or Changed Records in a table**, as shown in [Figure 4-23](#).

Figure 4-23 Selecting an Operation Type



8. Click **Next**.

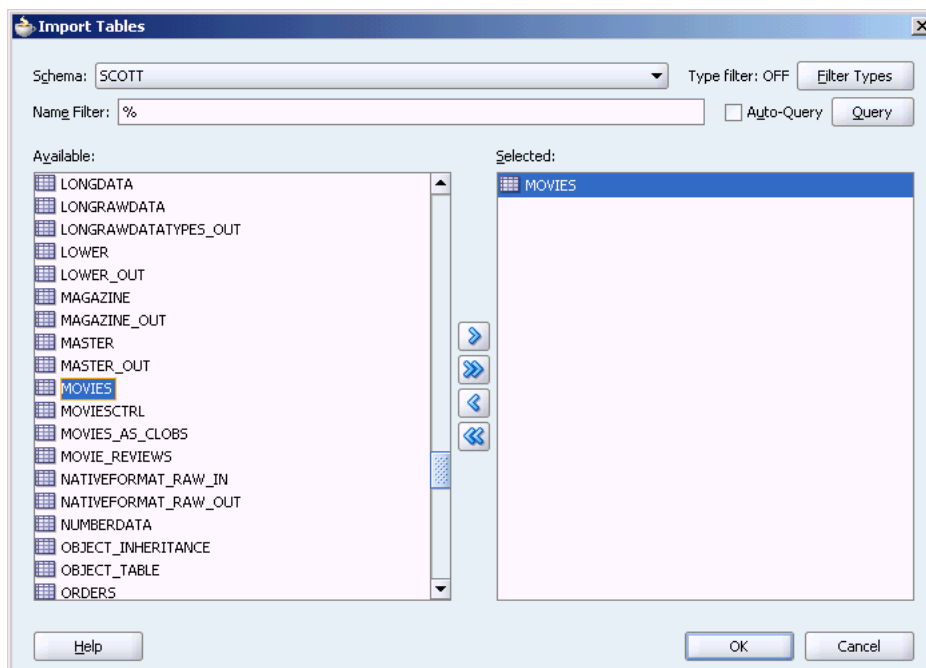
The Select Table dialog box is displayed.

9. Click **Import Tables**.

The Import Tables dialog box is displayed.

10. Double-click the **Movies** table to select it, as shown in [Figure 4-24](#).

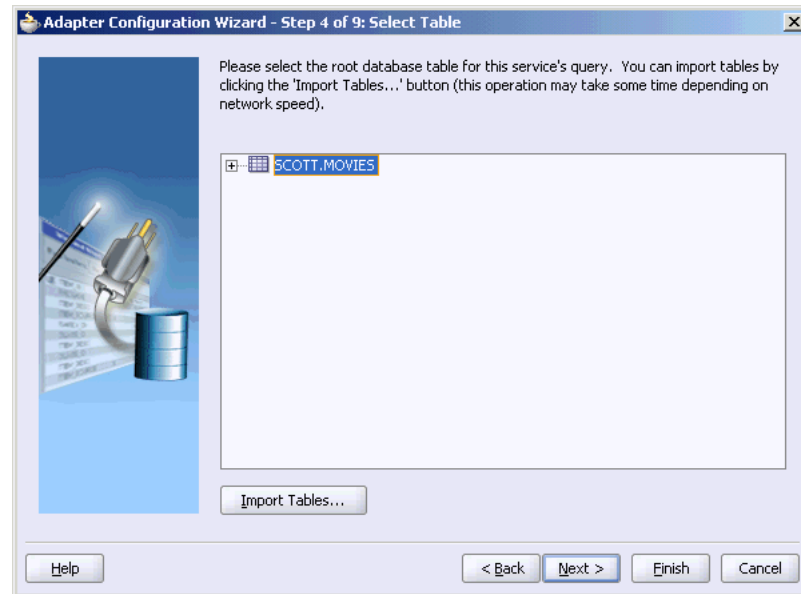
Figure 4-24 The Import Tables Dialog Box



11. Click **OK**.

The Select Tables dialog box is displayed with the Movies table selected, as shown in [Figure 4-25](#).

Figure 4-25 The Select Table Dialog Box



12. Click **Next**.

The Relationships dialog box is displayed.

13. In the Relationships dialog box you can define relationships that are reachable from the root database. However, in the example, you will not define any relationship. Click **Next**.

14. In the Object Filtering dialog box is displayed, click **Next**.

The After Read dialog box is displayed.

15. Select **Update a Sequencing File**, and then click **Next**.

The Sequencing File dialog box is displayed.

16. Enter the following information:

- **Sequencing File:** Specify the location of the of the sequencing file. In this example, specify,
`Oracle_`
`Home\bpel\samples\tutorials\122.DBAdapter\advanced\polling\PollingFileSeque`
`ncingStrategy\lastReadId.txt`
- **Sequenced ID Field:** Select **LATST_UPDATED**.

[Figure 4-26](#) shows the Sequencing File dialog box, wish the value populated.

Figure 4–26 The Sequencing File Dialog Box

Adapter Configuration Wizard - Step 8 of 10: Sequencing File

Specify the location of the sequencing file (which contains a single value, either a number or an ISO-format dateTime). You must also specify the field in the Movies table that contains the sequential ID.

Sequencing File:

Sequenced ID Field:

17. Click **Next**.

The Polling Options dialog box is displayed, as shown in [Figure 4–27](#).

Figure 4–27 The Polling Options Dialog Box

Adapter Configuration Wizard - Step 9 of 10: Polling Options

Specify any additional polling options. To use the recommended, default values, click Next.

Polling Frequency:

Database Rows per XML Document:

Database Rows per Transaction: ☒ Unlimited

Order By:

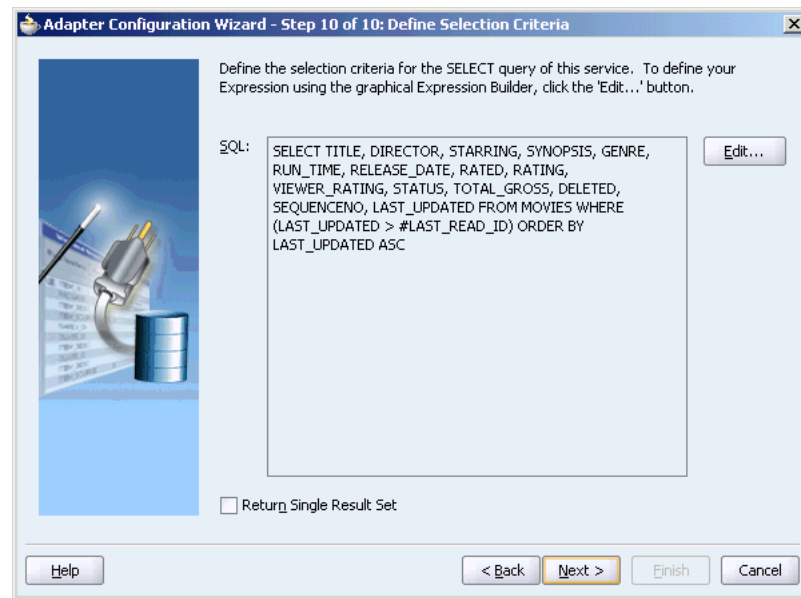
☐ Delay Commit ☐ Distributed Polling

☐ Poll for Child Updates ☐ Use Batch Destroy

SQL:

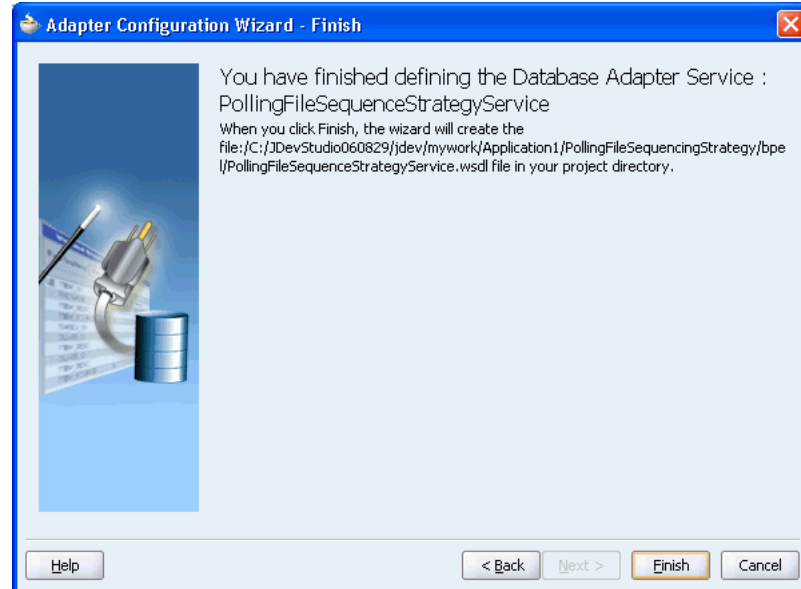
18. In the Polling Options dialog box, retain default values, and then click **Next**.

The Define Selection Criteria dialog box is displayed, as shown in [Figure 4–28](#).

Figure 4–28 The Define Selection Criteria Dialog Box

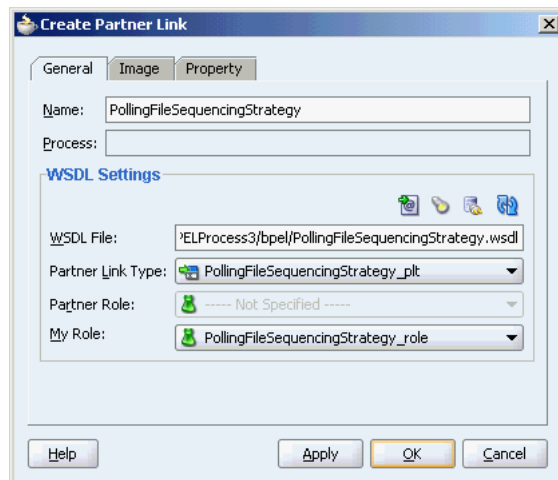
19. Retain default values, and then click **Next**.

The Adapter Configuration Wizard-Finish screen is displayed, as shown in [Figure 4–29](#).

Figure 4–29 The Adapter Configuration Wizard - Finish Screen

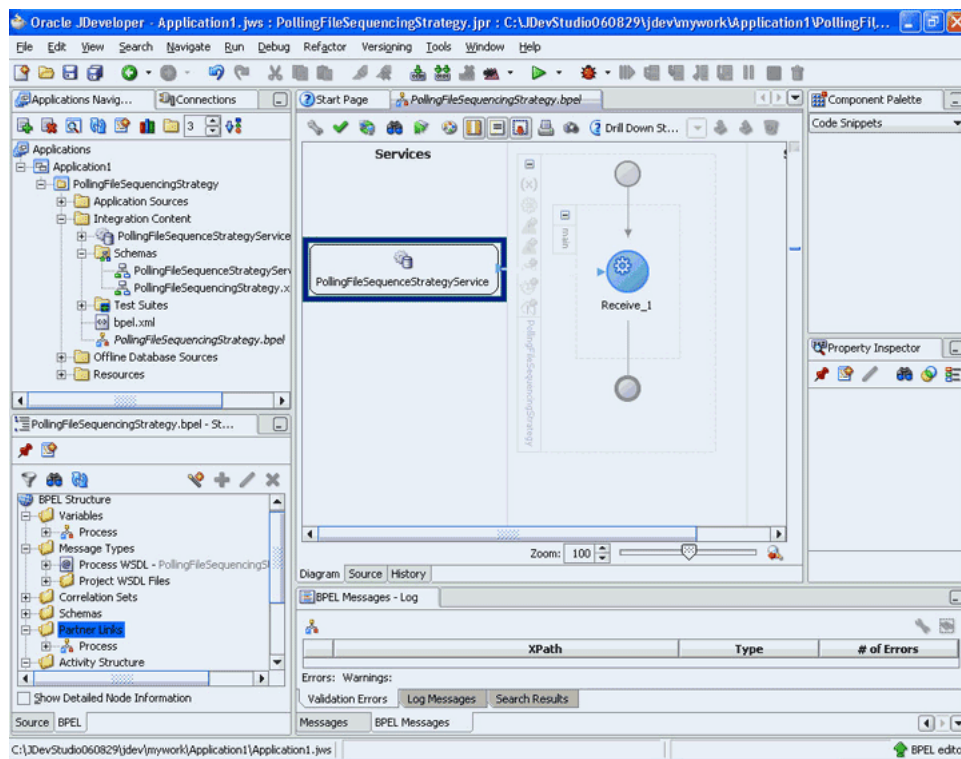
20. Click **Finish**.

The Create Partner Link dialog box is displayed with all the fields populated, as shown in [Figure 4–30](#).

Figure 4–30 The Create Partner Link Dialog Box

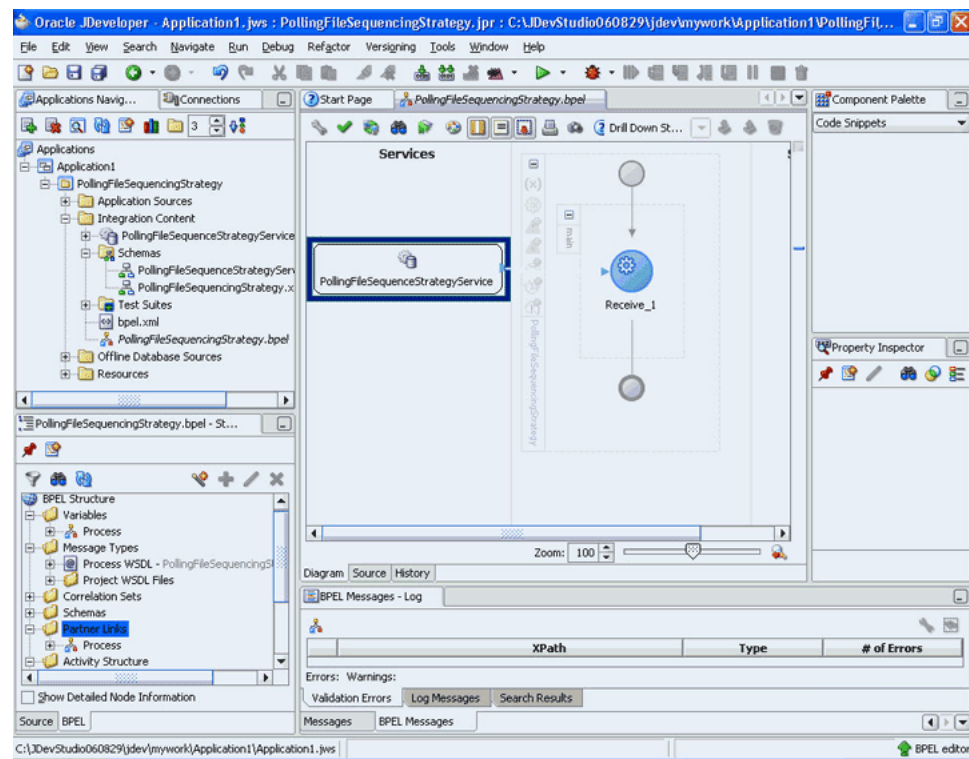
21. Click **Ok**.

Your screen will resemble [Figure 4–31](#).

Figure 4–31 The Application Screen

22. Right-click the project, select **Deploy**, point to **ServerConnection1**, and then click **Deploy to Default Domain**.

23. The Oracle BPEL Control will resemble [Figure 4–32](#).

Figure 4–32 The Oracle BPEL Control

4.4 Database Adapter Use Cases for Oracle Enterprise Service Bus

This use case shows a simple scenario for replicating data in one set of tables on one database to tables on another database. This scenario involves an inbound polling read on the source tables, and an outbound write/merge to destination tables.

In this use case, there are two sets of department and employee tables: SENDER_EMP and SENDER_DEPT, which are used for inbound data, and RECEIVER_EMP and RECEIVER_DEPT for outbound data.

Note that the operations against each set of tables are defined in separate WSDLs that can have their own Database Adapter JNDI (connection) names defined. This means that each set of tables could potentially reside in separate database schemas or instances.

This section comprises the following steps:

- [Prerequisites](#)
- [Creating a New ESB Project](#)
- [Creating Inbound Database Adapter](#)
- [Creating an Outbound Database Adapter Service](#)
- [Configuring Routing Service](#)
- [Checking the ESB Console](#)
- [Checking Execution in the ESB Control](#)

4.4.1 Prerequisites

Before you create the application, you must run the stored procedure, `create_schemas.sql` available at:

OracleAS_1\bpel\samples\tutorials\122.DBAdapter\MasterDetail\sql\oracle

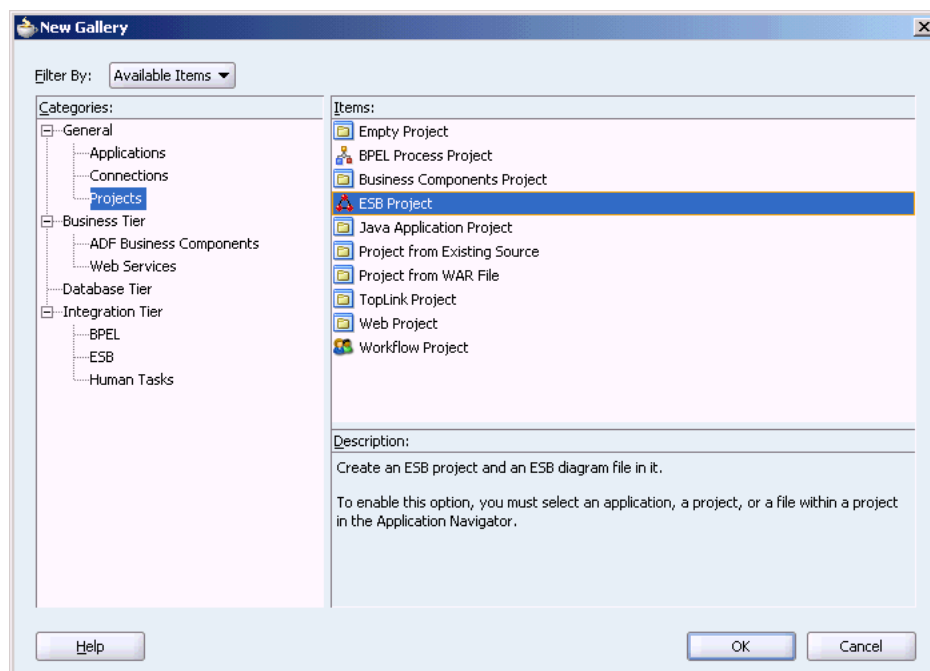
This stored procedure creates four tables: `SENDER_EMP` and `SENDER_DEPT`, from which data is read by the inbound database adapter, and `RECEIVER_EMP` and `RECEIVER_DEPT`, from which new data is stored by the outbound database adapter.

4.4.2 Creating a New ESB Project

The following are the steps to create a new ESB project:

1. Open Oracle JDeveloper.
2. From the **File** menu, select **New**.
The New Gallery dialog box is displayed.
3. Select **All Technologies** from the **Filter By** box. This displays a list of available categories.
4. Expand the **General** node, and then select **Projects**.
5. Select **ESB Project** from the Items group, as shown in [Figure 4-33](#)

Figure 4-33 Creating a New ESB Project



6. Click **OK**.
The Create ESB Project dialog box is displayed.
7. In the **Project Name** field, enter a descriptive name. For example, `MasterDetails`.
8. Click **OK**.

You have created a new ESB project, titled `MasterDetails`.

4.4.3 Creating Inbound Database Adapter

The following are the steps to create an inbound Database adapter:

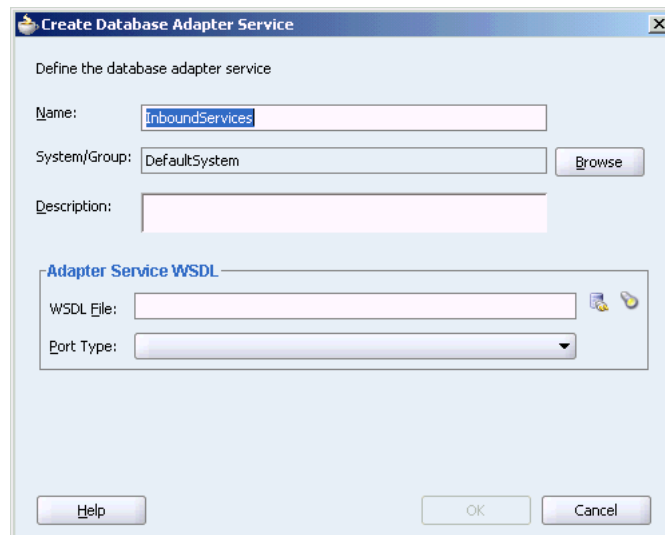
1. Select **Adapter Services** from the Component Palette, and then drag and drop **Database Adapter** into the `MasterDetails.esb` project.

The Create Database Adapter Service dialog box is displayed.

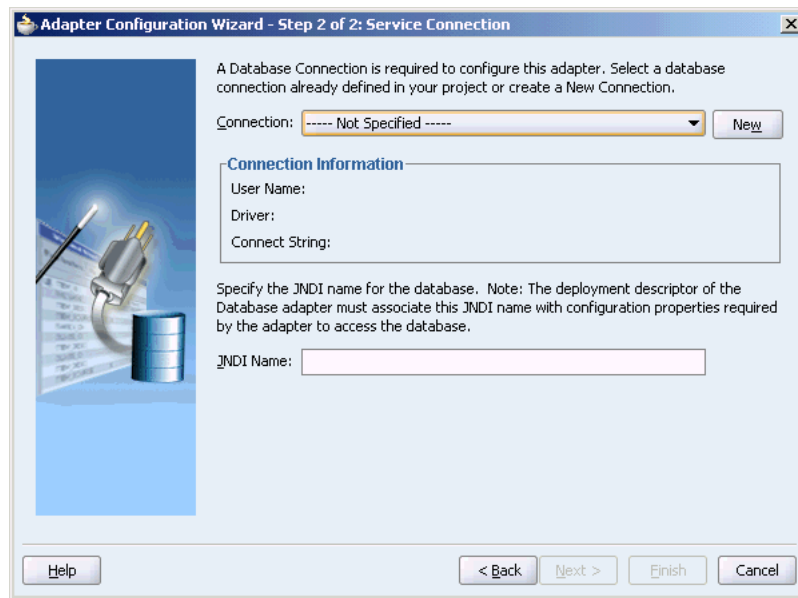
2. Specify the following information in the Create Database Adapter Service dialog box:
 - **Name:** Type a name for the service. In this example, type `InboundServices`.
 - **System/Group:** Retain the default value.

Figure 4–34 shows the Database Adapter Service dialog box with the **Name** and **System/Group** fields, filled up.

Figure 4–34 Defining the Database Adapter Service



3. Under Adapter Service WSDL, click the **Configure adapter service wsdl** icon.
The Adapter Configuration wizard Welcome page is displayed.
4. Click **Next**.
The Service Name dialog box is displayed with the Service Name field, filled up.
5. Retain the service name, and click **Next**.
The Service Connection dialog box is displayed, as shown in Figure 4–35

Figure 4–35 The Service Connection Dialog Box

6. Click **New** to define a database connection.

The Create Database Connection Wizard Welcome page is displayed.

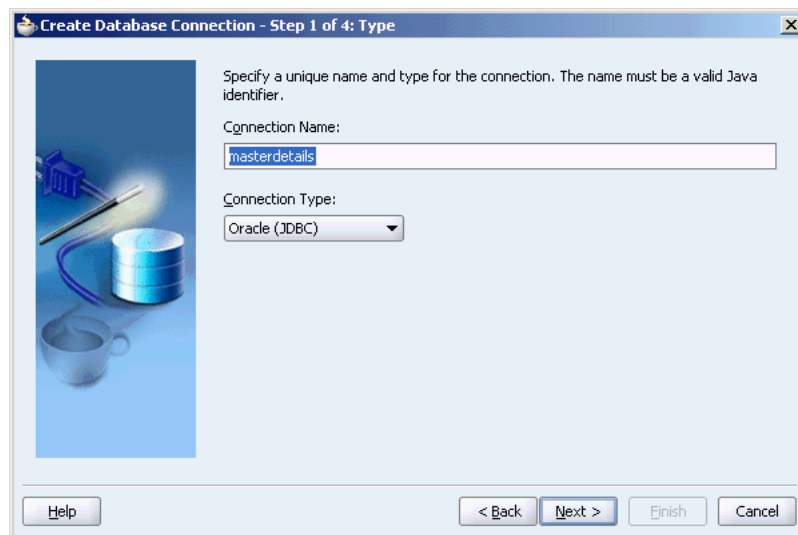
7. Click **Next**.

The Type dialog box is displayed.

8. Enter the following information in the Type dialog box:

- a. In the **Connection Name** field, specify a unique name for the database connection. In this example, type **masterdetails**.
- b. From the **Connection Type** box, select **Oracle (JDBC)**.

Figure 4–36 shows the Type dialog box.

Figure 4–36 Specifying the Connection Name and Type of Connection

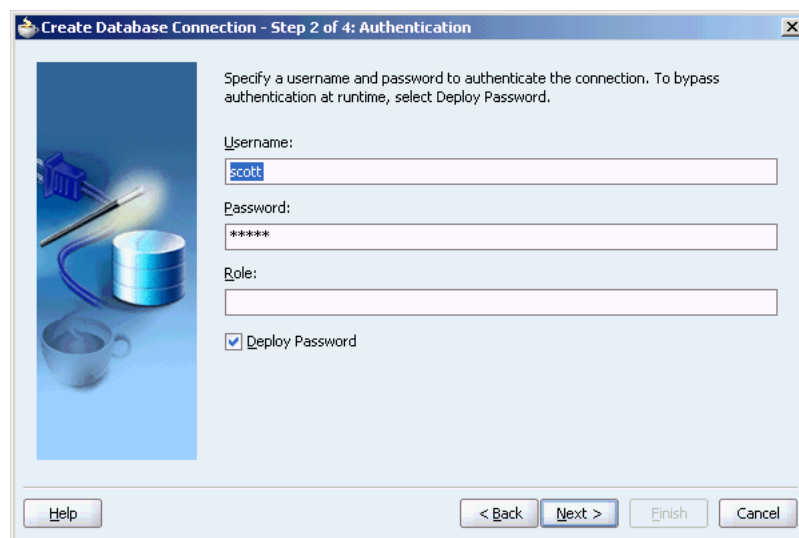
9. Click **Next**.

The Authentication dialog box is displayed.

10. Enter the authentication credentials in the following fields:
 - a. In the **UserName** field, specify a unique name for the database connection. In this example, type `scott`.
 - b. In the **Password** field, specify a password for the database connection. In this example, type `tiger`.
 - c. Leave the **Role** field blank.
 - d. Select **Deploy Password**.

Figure 4–37 shows the Authentication dialog box with the credentials fields, populated.

Figure 4–37 Specifying the Authentication Credentials

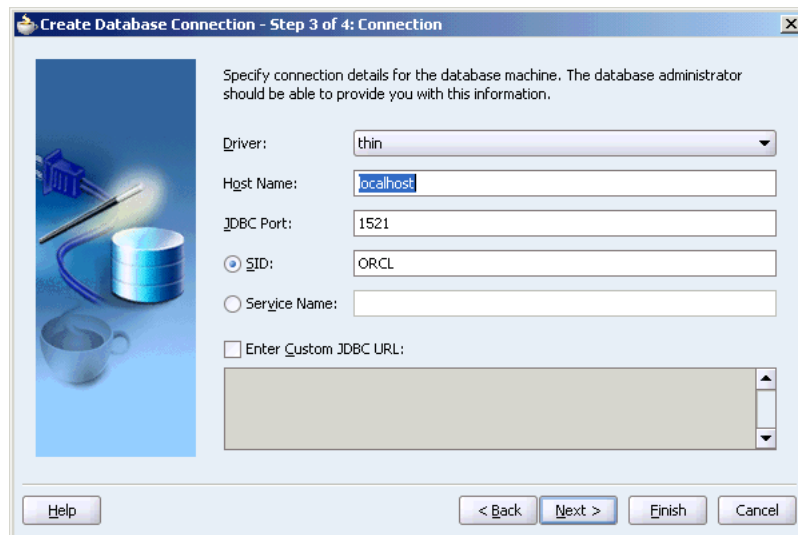


11. Click **Next**.

The Connection dialog box is displayed.

12. Enter information in the following fields:
 - a. In the **Driver** list, retain the default value, **Thin**.
 - b. In the **Host Name** field, retain the default value, **localhost**.
 - c. In the **JDBC Port** field, specify the port number for the database connection. In this example, type `1521`.
 - d. In the **SID** field, specify a unique SID value for the database connection. In this example, type `ORCL`.

Figure 4–38 shows the Connection dialog box.

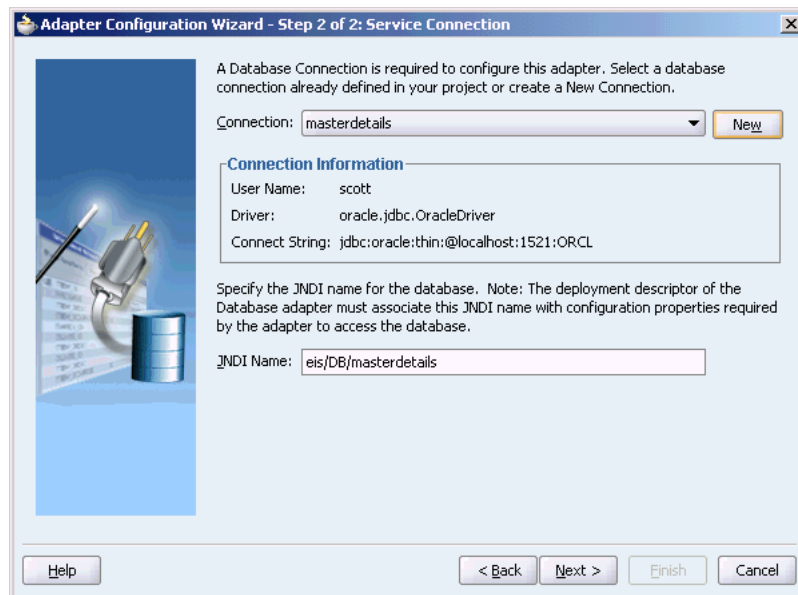
Figure 4–38 Specifying the New Database Connection Information

13. Click **Next**.

The Test dialog box is displayed.

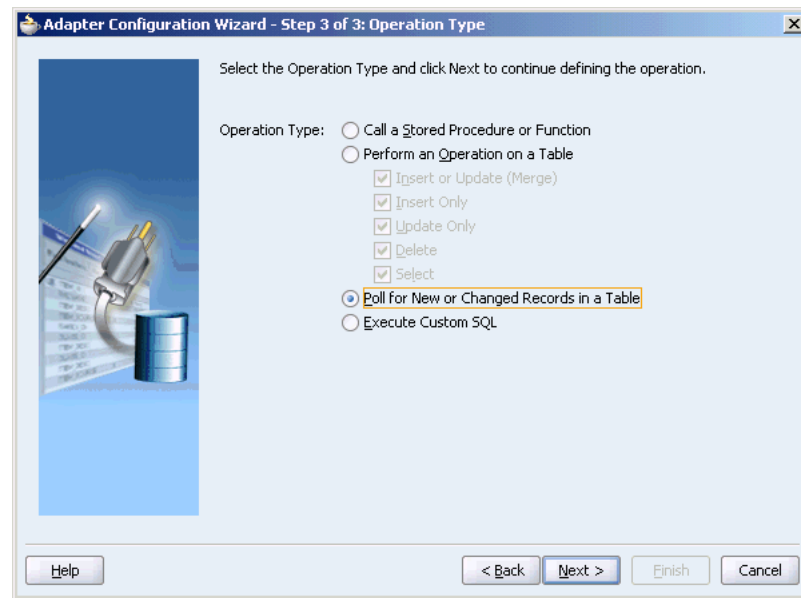
14. Click **Test Connection** to determine whether the specified information establishes a connection with the database.
15. Click **Finish** to complete the process of creating a new database connection.

The Service Connection dialog box is displayed, providing a summary of the database connection, as shown in [Figure 4–39](#).

Figure 4–39 The Service Connection Dialog Box

16. Click **Next**.

The Operation dialog box is displayed, as shown in [Figure 4–40](#).

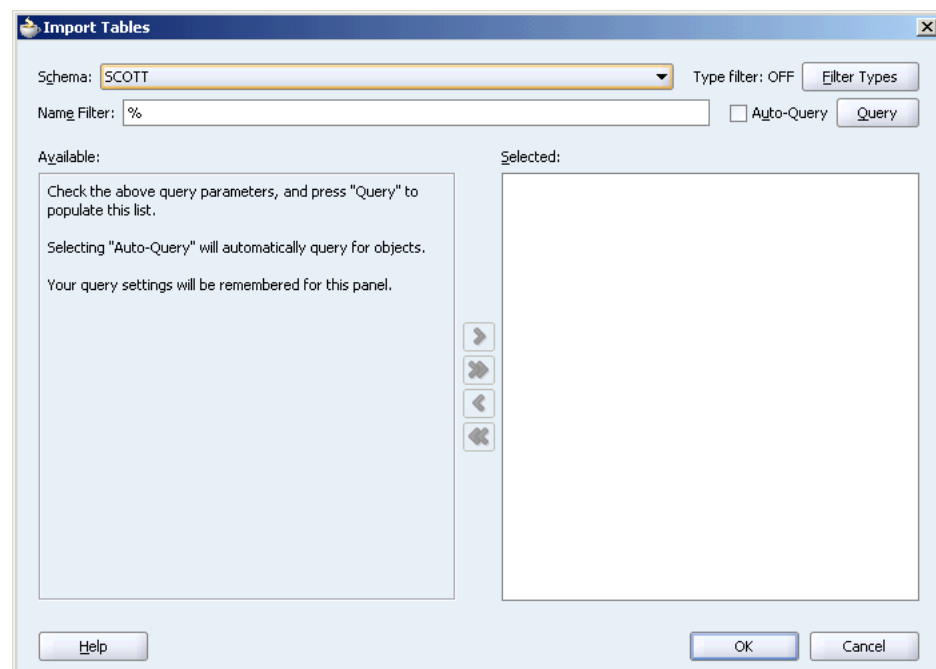
Figure 4–40 Specifying the Operation Type

17. Select **Poll for New or Changed Records in a Table**, and then click **Next**.

The Select Table dialog box is displayed.

18. Click **Import Tables** to import tables.

The Import Tables dialog box is displayed, as shown in [Figure 4–41](#).

Figure 4–41 The Import Tables Dialog Box

19. Select the following query parameters in the Import Tables dialog box:

- **Schema:** Select **Scott**
- **Name Filter:** Click **Query** to retrieve objects from database.

A list of available tables is displayed.

20. Select RECEIVER_DEPT and RECEIVER_EMP from the Available column, and move it to the Selected column by clicking the **Add** button.

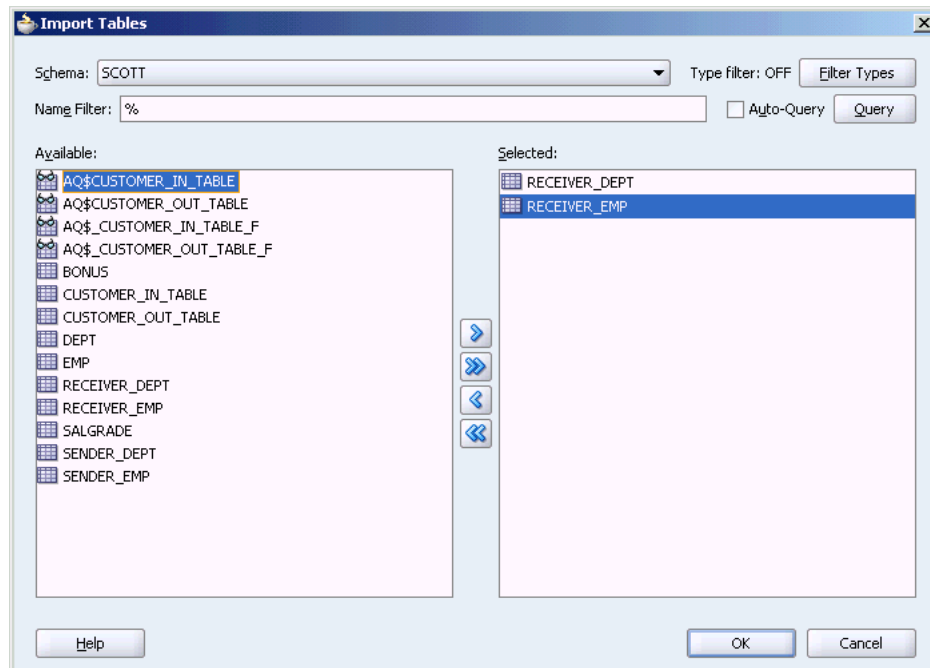
Figure 4–42 shows the Add button.

Figure 4–42 The Add Button



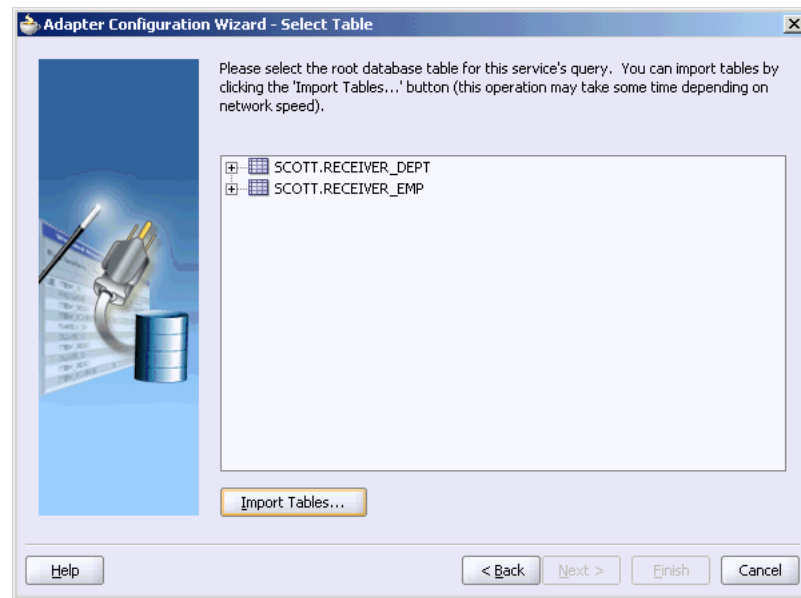
The selected tables now appear in the Selected column, as shown in Figure 4–43.

Figure 4–43 Adding the Required Tables from the Schema



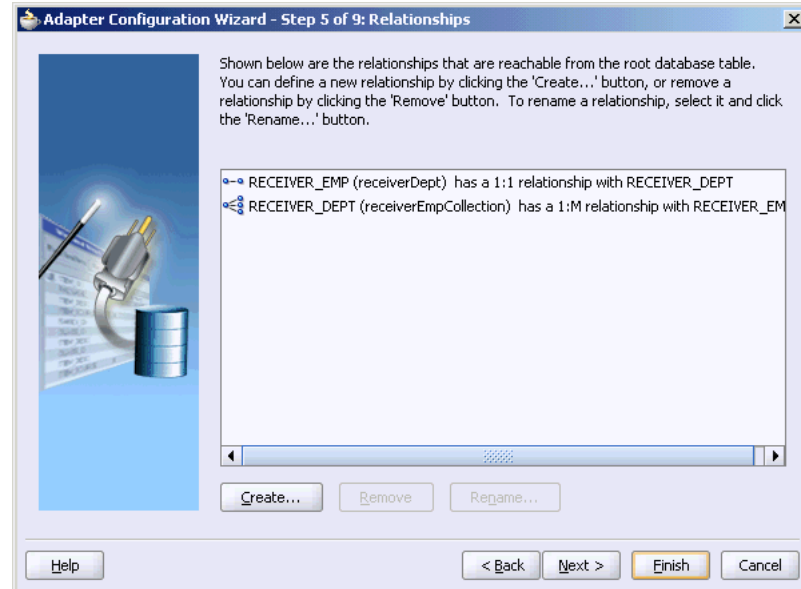
21. Click **Ok**.

The Select Table dialog box is displayed with the tables you imported, as shown in Figure 4–44.

Figure 4–44 The Select Tables Dialog Box with the Imported Tables

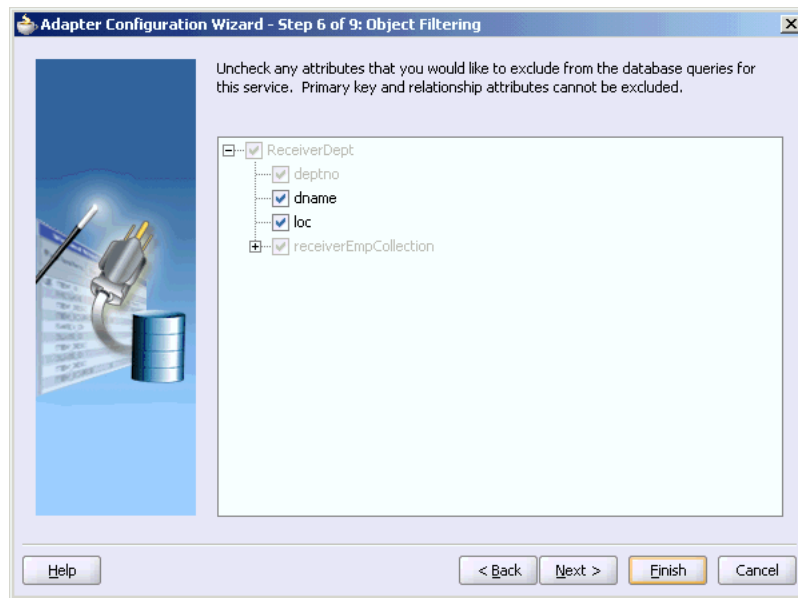
22. Select a root table, and then click **Next**. In this example, select `SCOTT.RECEIVER_DEPT`.

The Relationships dialog box is displayed, as shown in [Figure 4–45](#).

Figure 4–45 The Relationships Dialog Box

23. Click **Next**.

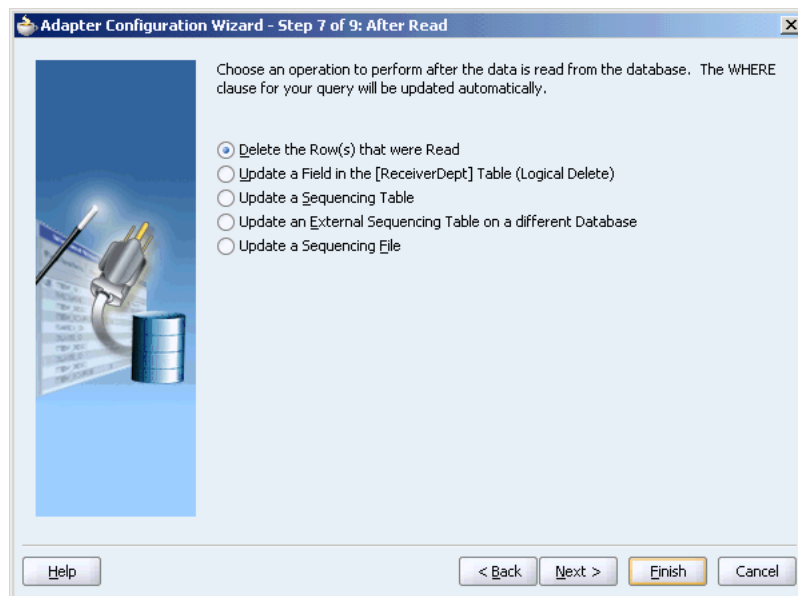
The Object Filtering dialog box is displayed, as shown in [Figure 4–46](#).

Figure 4–46 The Object Filtering Dialog Box

24. Click **Next**.

The After Read dialog box is displayed.

25. In the After Read dialog box, select **Delete the Row(s) that were Read**, as shown in [Figure 4–47](#), and then click **Next**.

Figure 4–47 The After Read Dialog Box

The Polling Options dialog box is displayed, as shown in [Figure 4–48](#).

Figure 4–48 The Polling Options Dialog Box

Adapter Configuration Wizard - Step 8 of 9: Polling Options

Specify any additional polling options. To use the recommended, default values, click Next.

Polling Frequency: 5 seconds

Database Rows per XML Document: 1

Database Rows per Transaction: ☒ Unlimited

Order By: DEPTNO

☐ Delay Commit ☐ Distributed Polling

☐ Poll for Child Updates ☐ Use Batch Destroy

SQL:

```
SELECT DEPTNO, DNAME, LOC FROM RECEIVER_DEPT ORDER BY DEPTNO ASC
```

Help < Back Next > Finish Cancel

26. Retain the default value in the Polling Options dialog box and then click **Next**.

The Define Selection Criteria dialog box is displayed, as shown in [Figure 4–49](#).

Figure 4–49 The Define Selection Criteria Dialog Box

Adapter Configuration Wizard - Step 9 of 9: Define Selection Criteria

Define the selection criteria for the SELECT query of this service. To define your Expression using the graphical Expression Builder, click the 'Edit...' button.

SQL:

```
SELECT DEPTNO, DNAME, LOC FROM RECEIVER_DEPT ORDER BY DEPTNO ASC
```

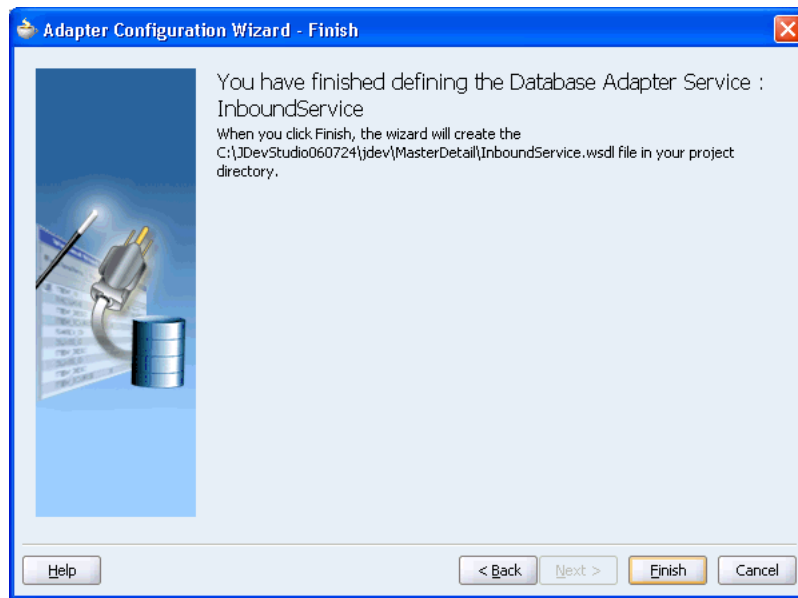
Edit...

☐ Return Single Result Set

Help < Back Next > Finish Cancel

27. Retain the default query, and click **Next**. However, to define your own expression, click **Edit**

The Adapter Configuration Wizard - Finish screen is displayed, as shown in [Figure 4–50](#).

Figure 4–50 The Adapter Configuration Wizard- Finish

4.4.4 Creating an Outbound Database Adapter Service

The following are the steps to create an inbound Database adapter:

1. Select **Adapter Services** from the Component Palette, and then drag and drop **Database Adapter** into the `MasterDetails.esb` project.

The Create Database Adapter Service dialog box is displayed.

2. Specify the following information in Database Adapter Service dialog box:

- **Name:** Type a name for the service. In this example, type `OutboundServices`.
- **System/Group:** Retain the default value.

3. Under Adapter Service WSDL, click the **Configure adapter service wsdl** icon.

The Adapter Configuration wizard Welcome page is displayed.

4. Click **Next**.

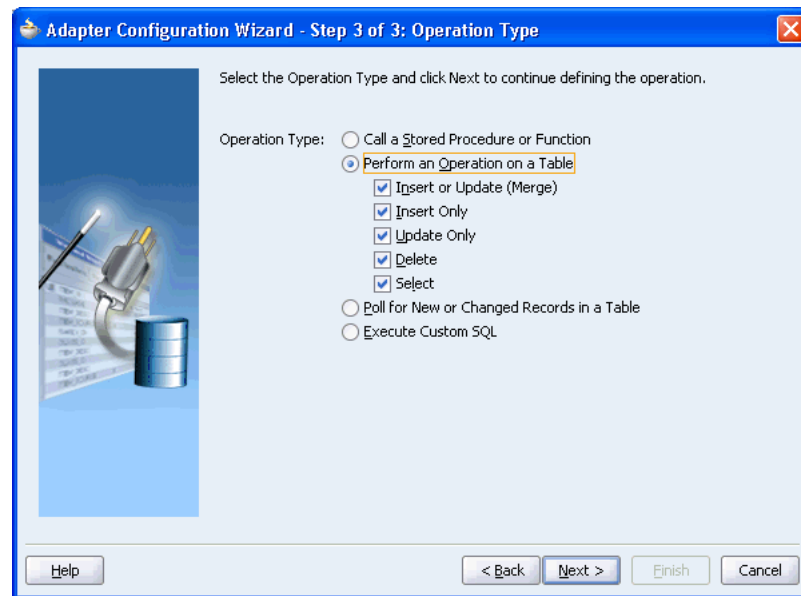
The Service Name dialog box is displayed with the Service Name field, filled up.

5. Retain the service name, and click **Next**.

The Service Connection dialog box is displayed.

6. Select the connection that you already created, and the click **Next**.

The Operation dialog box is displayed, as shown in [Figure 4–51](#).

Figure 4–51 Specifying the Operation Type

7. Select **Perform an Operation on a Table**, and then click **Next**.

The Select Table dialog box is displayed.

8. Click **Import Tables** to import tables.

The Import Tables dialog box is displayed.

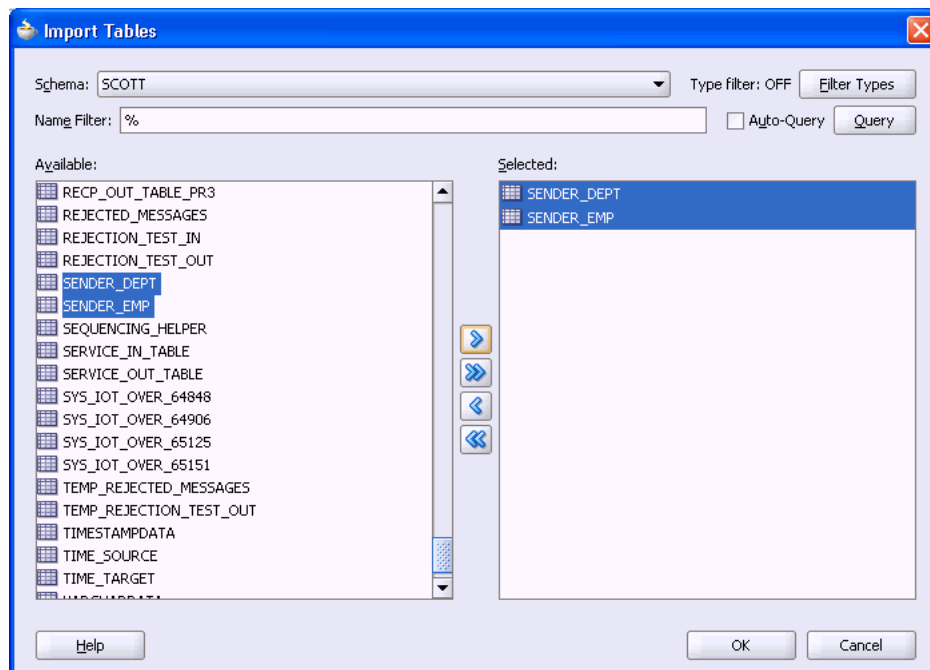
9. Select the following query parameters in the Import Tables dialog box:

- **Schema:** Select **Scott**
- **Name Filter:** Click **Query** to retrieve objects from database.

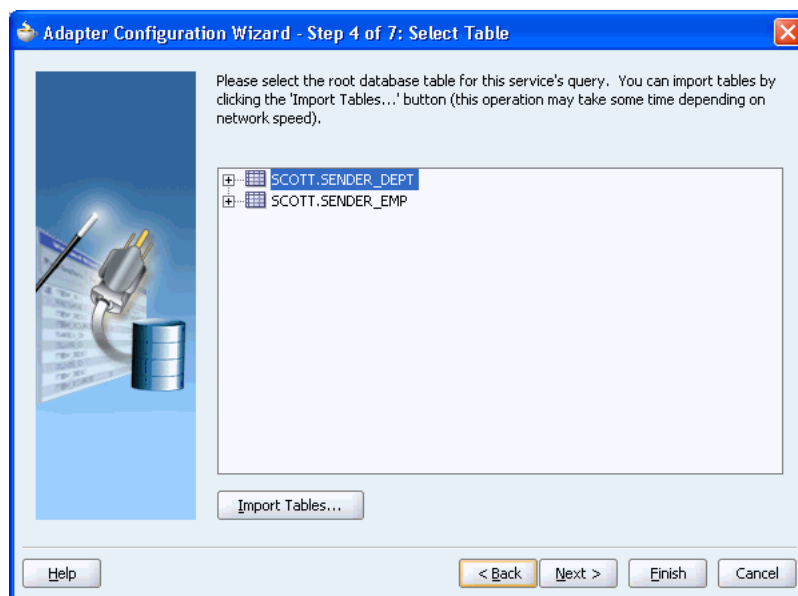
A list of available tables is displayed.

10. Select **SENDER_DEPT** and **SENDER_EMP** from the Available column, and move it to the Selected column by clicking the **Add** button.

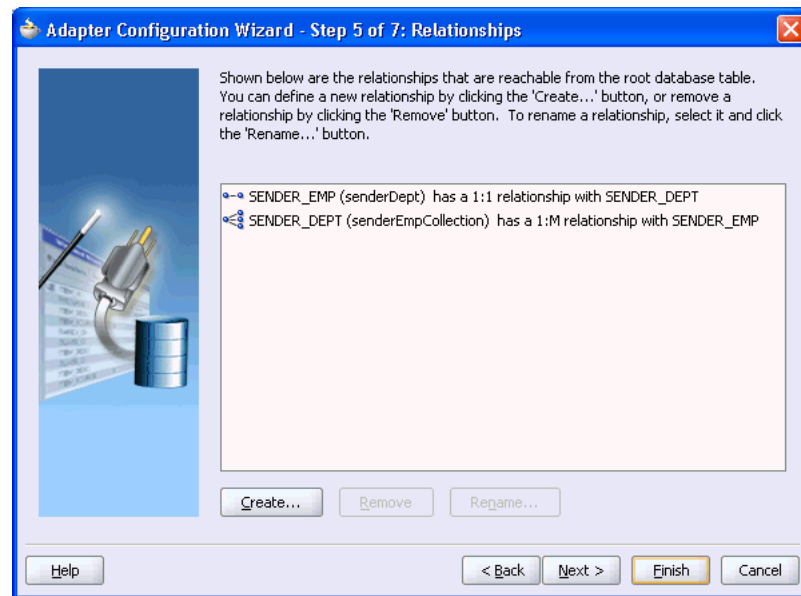
The selected tables now appears in the Selected column, as shown in [Figure 4–52](#).

Figure 4–52 Adding the Required Tables from the Schema**11. Click Ok.**

The Select Table dialog box is displayed with the tables you imported, as shown in [Figure 4–53](#).

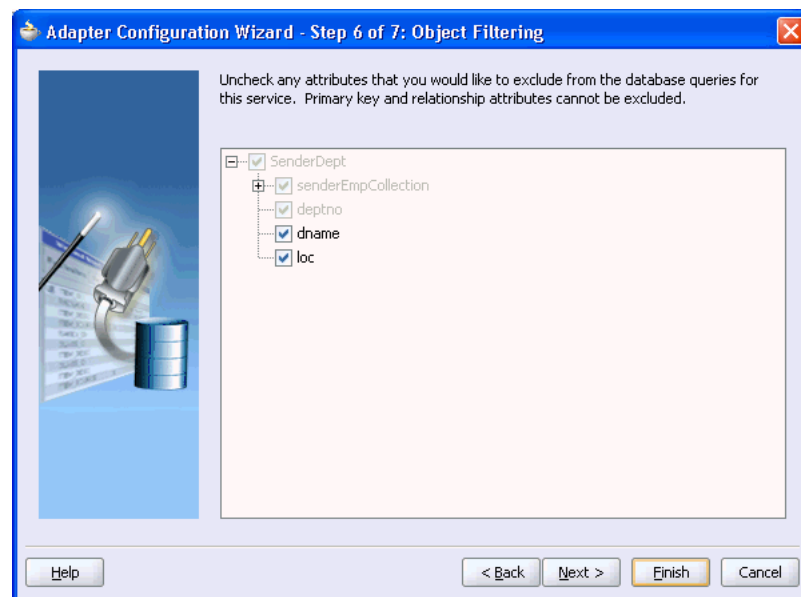
Figure 4–53 The Select Tables Dialog Box with the Imported Tables**12. Select a root table, and then click Next. In this example, select SCOTT . SENDER_ DEPT.**

The Relationships dialog box is displayed, as shown in [Figure 4–55](#).

Figure 4–54 The Relationships Dialog Box

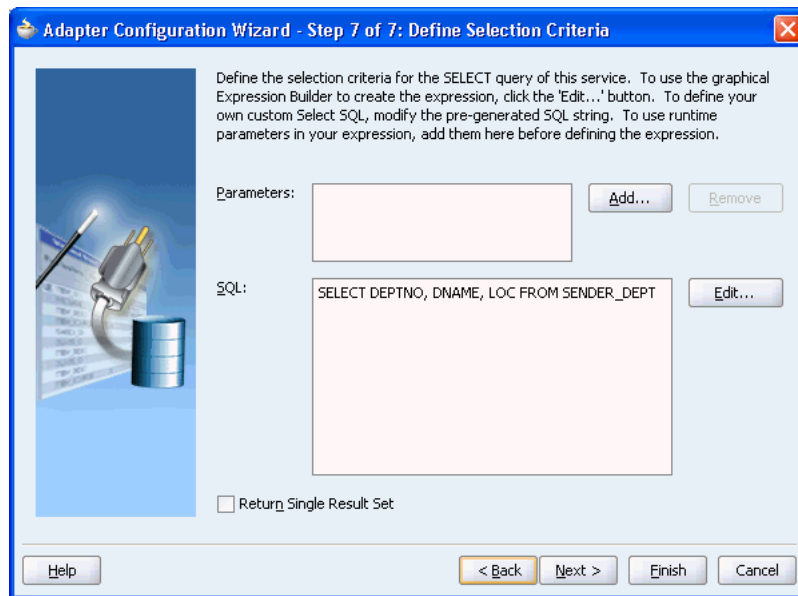
13. Click Next.

The Object Filtering dialog box is displayed, as shown in [Figure 4–55](#).

Figure 4–55 The Object Filtering Dialog Box

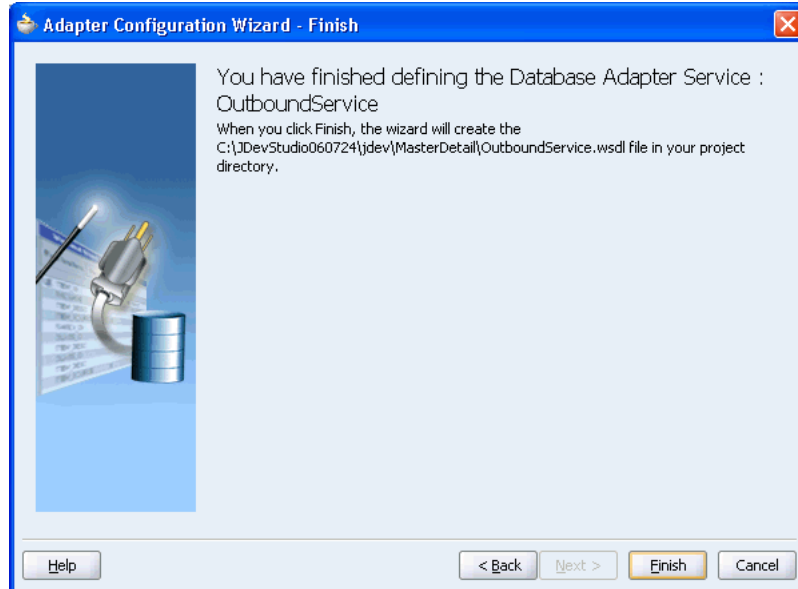
14. Click Next.

The Define Selection Criteria dialog box is displayed, as shown in [Figure 4–56](#).

Figure 4–56 The Define Selection Criteria Dialog Box

15. Click **Next**.

The Finish screen confirming the creation of the outbound service is displayed, as shown in [Figure 4–57](#).

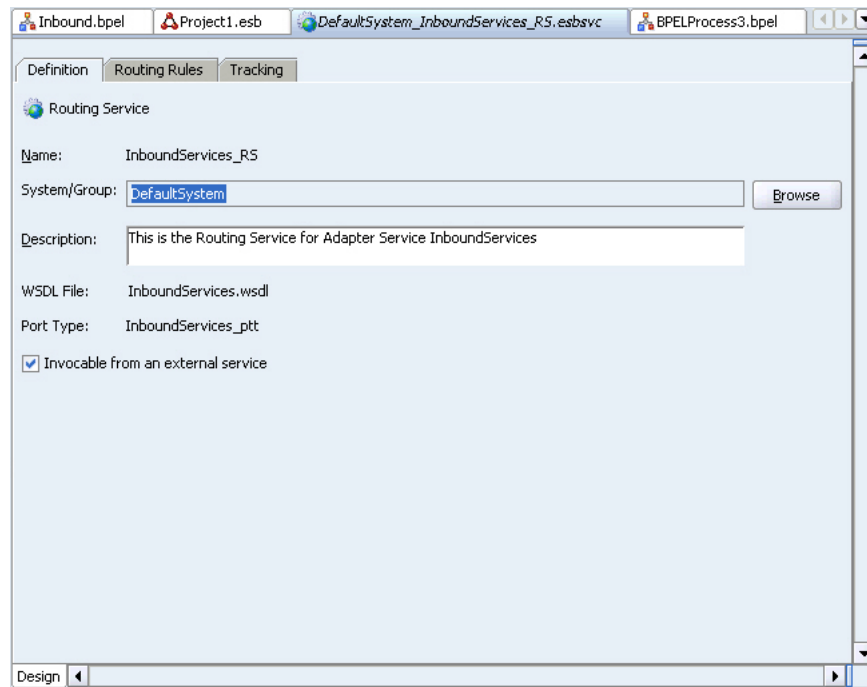
Figure 4–57 The Adapter Configuration Wizard Screen - Finish Screen

4.4.5 Configuring Routing Service

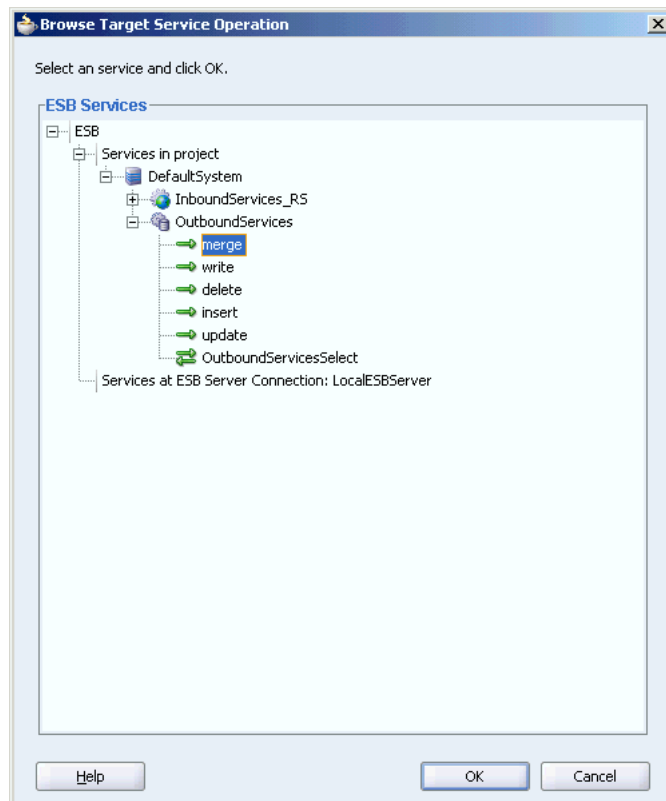
The following are the steps to configure the routing service:

1. Double-click the `InboundServices` routing service.

The Routing Service window is displayed in the midpane of the Application window, as shown in [Figure 4–58](#).

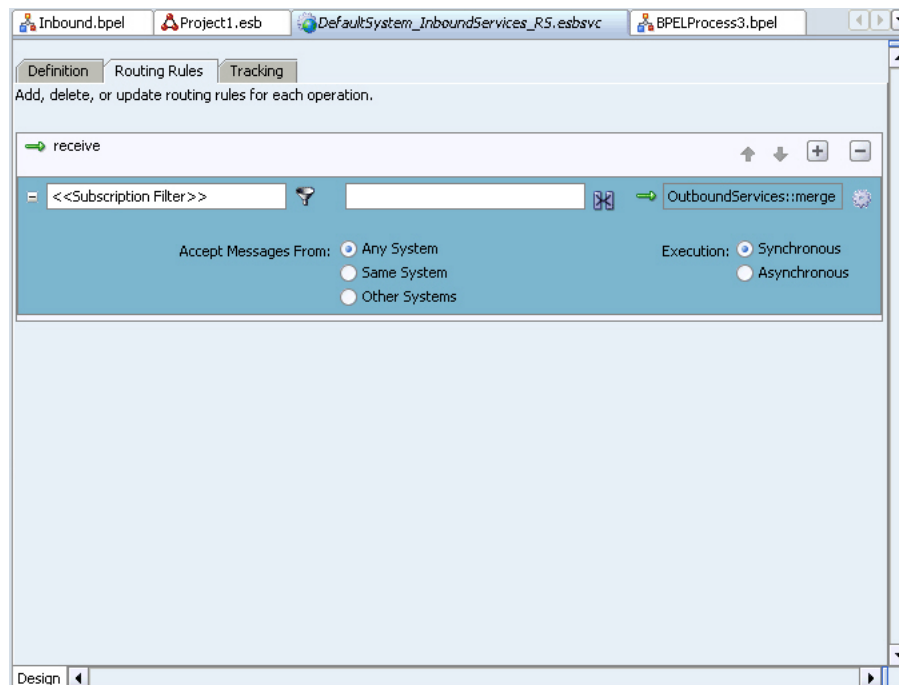
Figure 4–58 The Routing Screen Window

2. Select the **Routing Rules** tab, and then click + icon to add a rule.
The Browse Target Service Operation dialog box is displayed.
3. Select the **Merge** service, as shown in [Figure 4–59](#), and then click **OK**.

Figure 4–59 The Browse Target Service Operation Dialog Box

4. Click **Ok**.

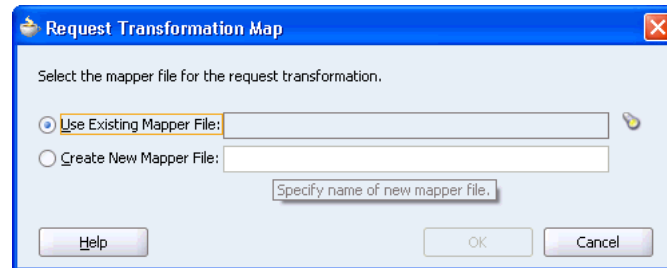
The middle pane of the application window will resemble [Figure 4–60](#).

Figure 4–60 Selecting the Transformation Map

5. Double-click the **Transformation** icon, and then click the **Select Create New Mapper File** icon.

The Request Transformation Map dialog box is displayed, as shown in [Figure 4–61](#).

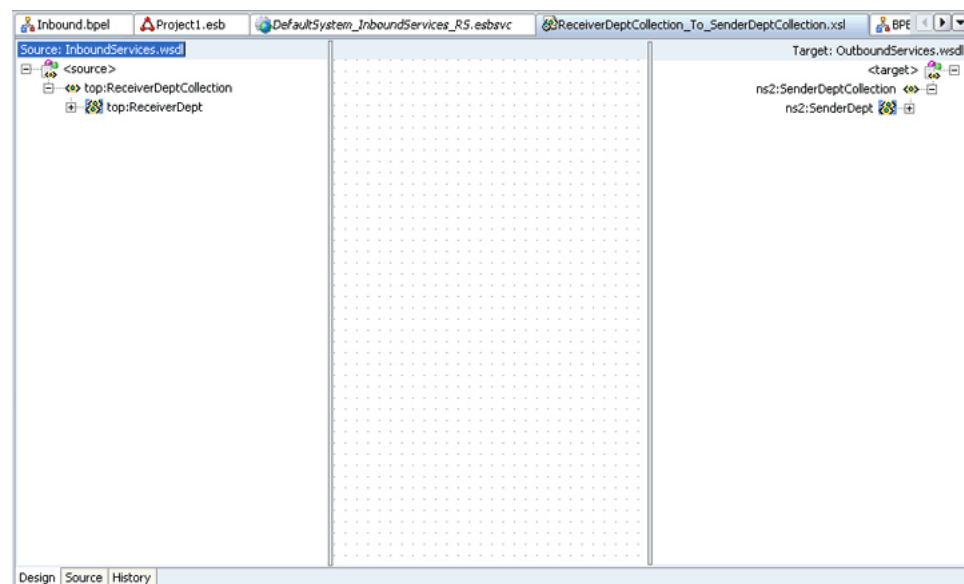
Figure 4–61 The Request Transformation Map Dialog Box



6. Select the **Create New Mapper File** option.
7. Accept the default values, and click **OK**.

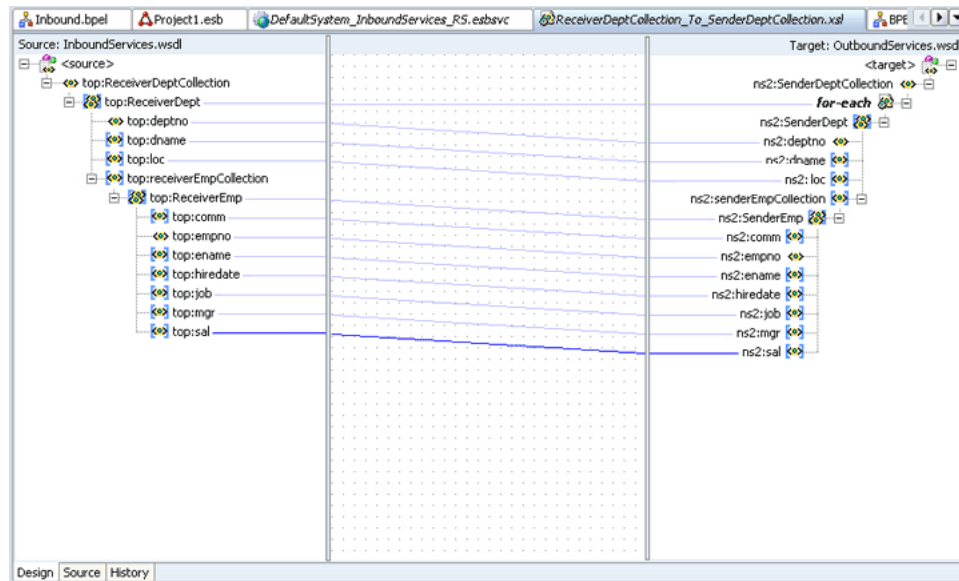
The Transformation window appears, as shown in [Figure 4–62](#).

Figure 4–62 Transformation Definitions



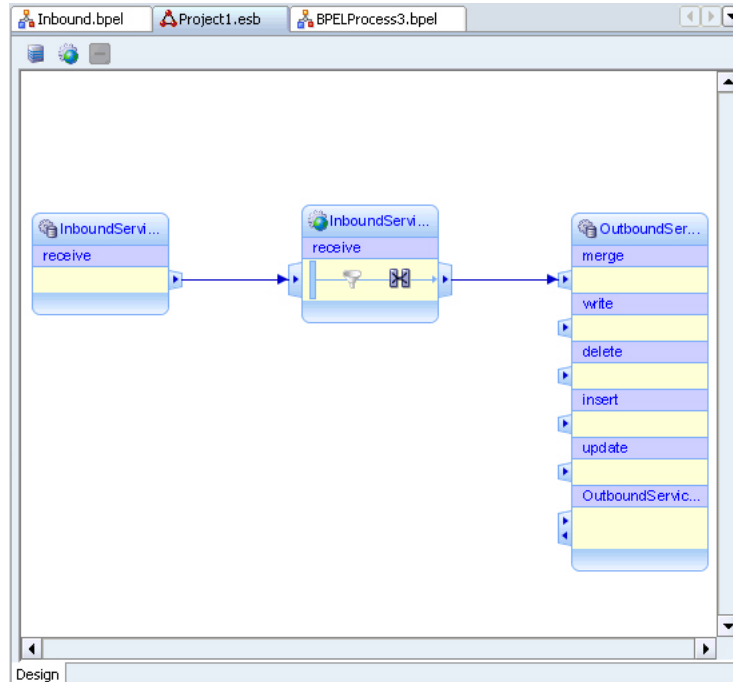
8. Select elements on the left-hand side of the mapper and drag it over to the elements on the right-hand side to set the map preferences.

The middle pane of the application window will resemble [Figure 4–63](#).

Figure 4–63 Setting Map Preferences

9. Save and close the tab for the mapper.
10. Save and close the tab for the routing service.

The midpane of the MasterDetails project will resemble [Figure 4–64](#).

Figure 4–64 The MasterDetails Project After Setting Map Preferences

11. Edit oc4j-ra.xml to reflect your database connection. For example,

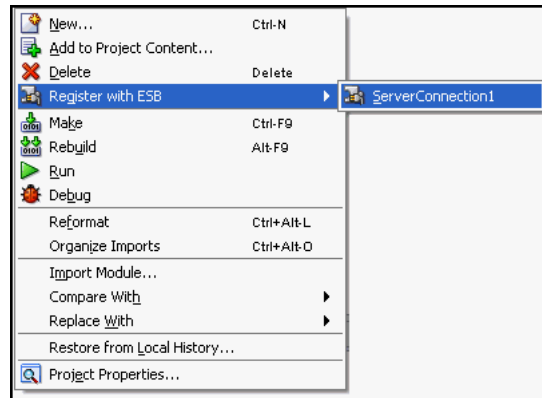

```
eis/DB/masterdetails
```

 oc4j-ra.xml is available at the location where you have installed OracleAS. For example:

```
C:\product\10.1.3.1\OracleAS_
1\j2ee\home\application-deployments\default\DbAdapter\oc4j-ra.xml
```

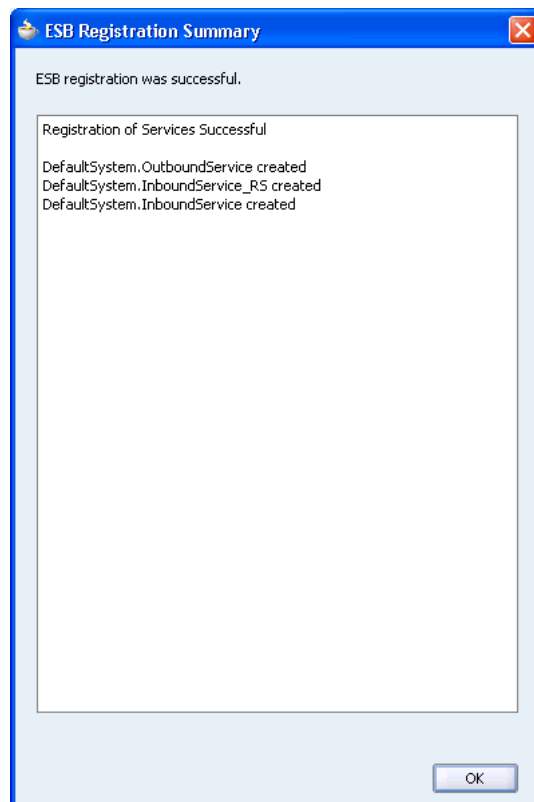
12. Right-click the project, select **Register with ESB**, and then click **LocalESBServer**, as shown in [Figure 4-65](#).

Figure 4-65 Deploying the Project



The Success page is displayed, as shown in [Figure 4-66](#).

Figure 4-66 The ESB Registration Summary Page

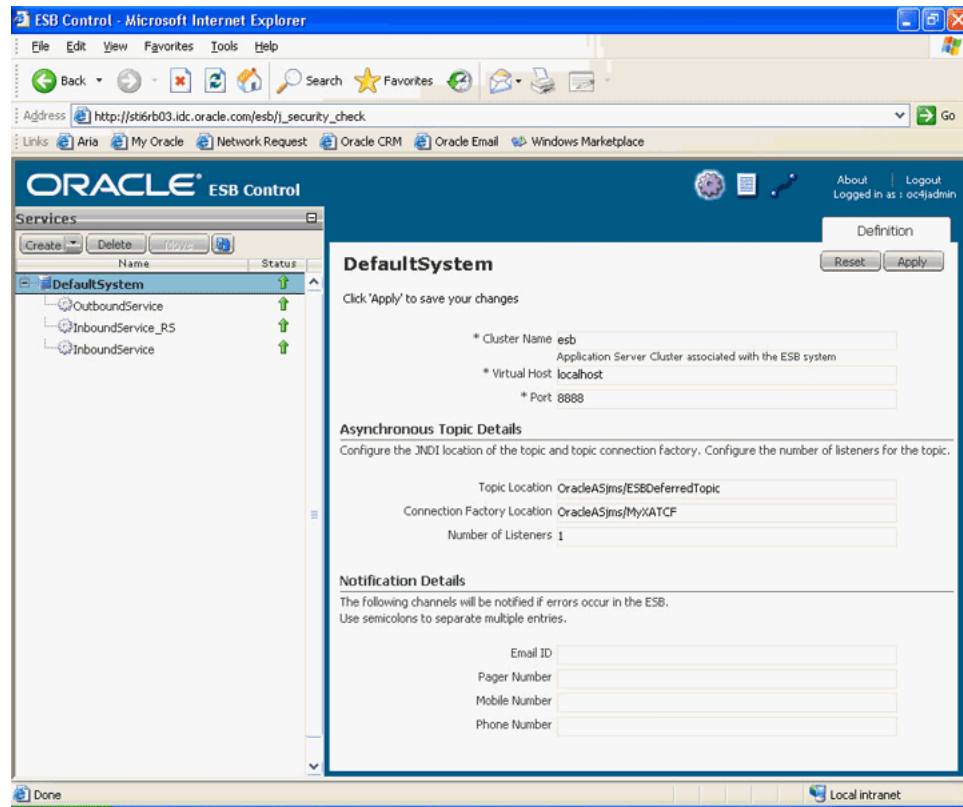


4.4.6 Checking the ESB Console

To check the ESB control, open the ESB Console. For example:
<http://localhost:8888/esb/esb/EsbConsole.html>

Now, your service window will resemble [Figure 4-67](#):

Figure 4-67 The ESB Console

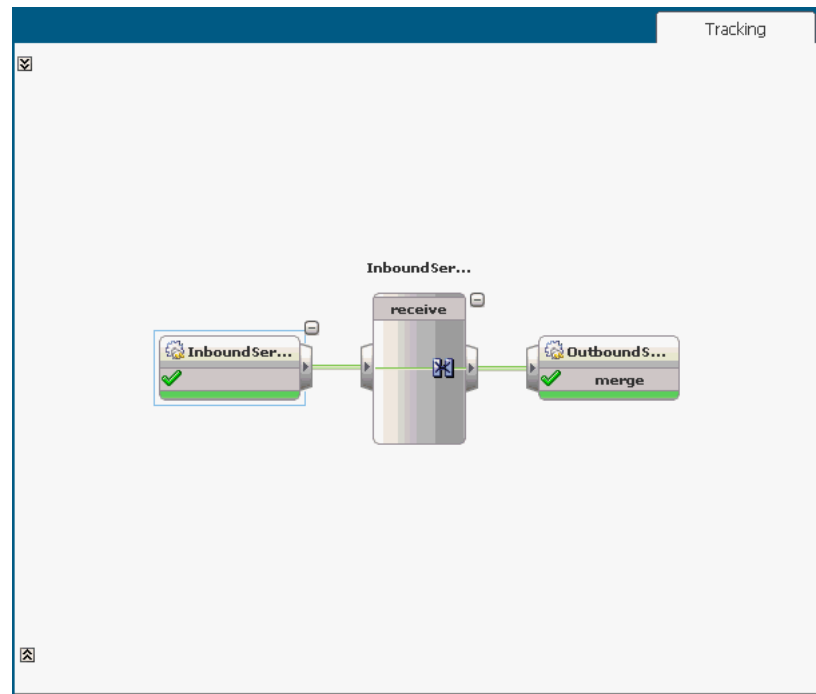


4.4.7 Checking Execution in the ESB Control

Use the following steps to check execution in the ESB control:

1. Open the ESB Console.
2. Click **Instances** on top-right corner.
3. Click the green arrow next to **Search**.

An instance resembling [Figure 4-68](#) is displayed.

Figure 4–68 The ESB Control Instance

4.5 Advanced Configuration

The Adapter Configuration Wizard generates everything you need to use the database adapter as part of a BPEL process. The following sections describe what happens in the background when you use the wizard, as well as performance considerations.

This section contains the following topics:

- [The TopLink Workbench Project](#)
- [Relational-to-XML Mappings \(toplink_mappings.xml\)](#)
- [The Service Definition \(WSDL\)](#)
- [XML Schema Definition \(XSD\)](#)
- [Deployment](#)
- [Performance](#)
- [detectOmissions Feature](#)
- [OutputCompletedXml Feature](#)

4.5.1 The TopLink Workbench Project

The wizard works by creating an TopLink Workbench project as part of your BPEL process project. This TopLink project contains metadata for mapping a database schema to objects/XML.

The TopLink mappings are stored in two formats. The `toplink_mappings.mwp` file is your design time project, which you can edit visually in JDeveloper BPEL Designer. In contrast, the `toplink_mappings.xml` file is an XML representation of your project for use at run time. It is not as easy as editing the `.bpel` file, where there is only one file, but you can toggle between **Diagram View** and **Source**.

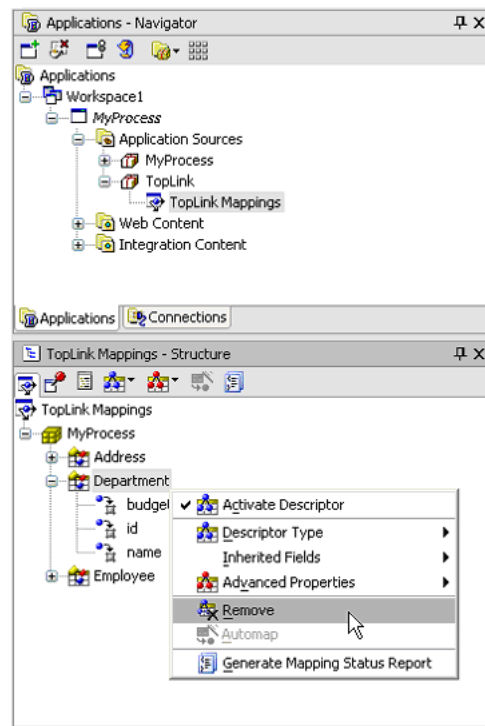
Note the following:

- Rather than edit the `toplink_mappings.xml` file directly, it is recommended that you edit the `toplink_mappings.mwp` visually, and regenerate all the BPEL artifacts to reflect the changes. You can do this by double-clicking the partner link to open the Adapter Configuration Wizard in edit mode, and then clicking through the wizard until you can click **Finish**. Changing the MWP version does not update the XML version until you click through the wizard in edit mode.
- When running the wizard, any changes that affect the TopLink project (importing tables, creating or removing mappings, specifying an expression, and so on) are applied immediately, and are *not undone* if you cancel the wizard.

4.5.1.1 Deleting a Descriptor

You cannot remove TopLink descriptors from your project from within the wizard because removing descriptors can potentially affect other partner links that are sharing that descriptor. To explicitly remove a descriptor, do the following:

- Click the **TopLink Mappings** node under **Application Sources** under your project in the **Applications - Navigator**.
- Select the descriptor from the tree in the **TopLink Mappings - Structure** pane.
- Right-click and select **Remove**.

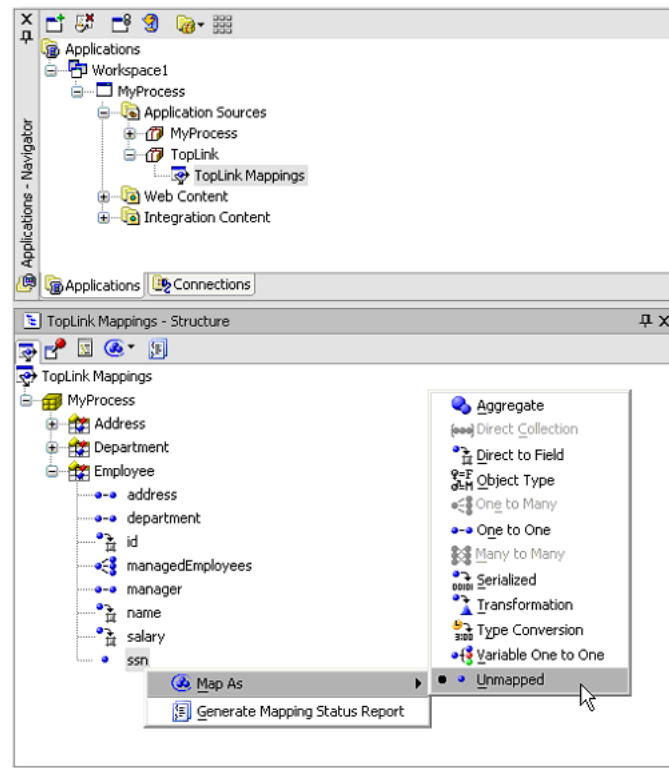


4.5.1.2 Returning Partial Objects When Querying

Currently, the Adapter Configuration Wizard does not have built-in support for partial object reading, that is, returning only specific fields from a table. To achieve this functionality, you can manually unmap any attributes that you do not want to include in the result set. Relationship mappings can be unmapped by removing them in the **Relationships** window, but direct mappings must be explicitly unmapped on the TopLink descriptor.

To unmap attributes:

1. Click the **TopLink Mappings** node under **Application Sources** under your project in the **Applications - Navigator**.
2. Select the descriptor containing the attribute you want to unmap from the tree in the **TopLink Mappings - Structure** pane.
3. Right-click the attribute you want to unmap and select **Map As > Unmapped**.

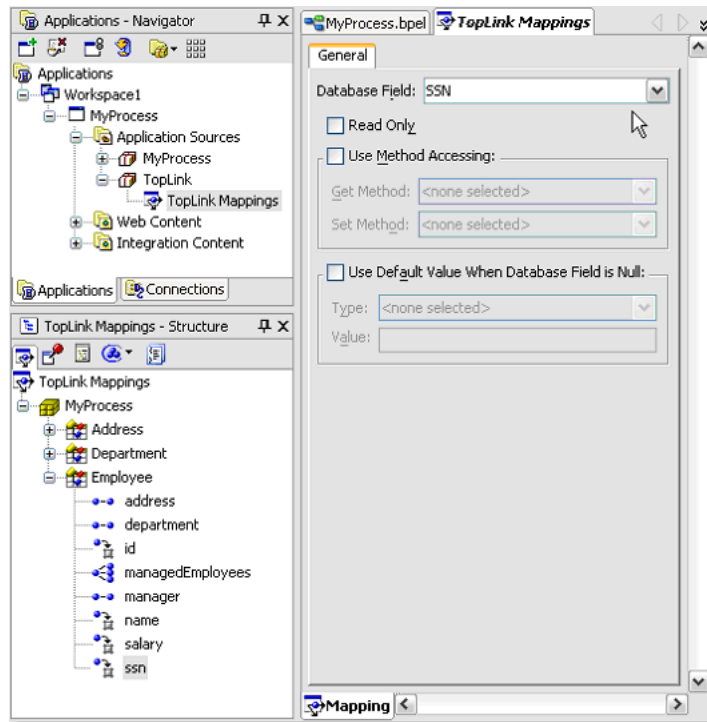


To remap the attribute, you can do the following:

1. Click the **TopLink Mappings** node under **Application Sources** under your project in the **Applications - Navigator**.
2. Select the descriptor containing the attribute you want to remap from the tree in the **TopLink Mappings - Structure** pane.
3. Right-click the attribute you want to remap and select **Map As > Direct to Field**.

The TopLink Mappings Editor automatically opens in the JDeveloper BPEL Designer window.

4. From **Database Field**, select the column to which the attribute should be mapped.



This image shows how to delete a descriptor. The top half of the image shows the Applications -Navigator window from the JDeveloper. The lower half is the TopLink Mappings - Structure window. The lower half contains a logical tree of elements. To the right of these windows, is a window showing the TopLink Mappings. The General tab is active (the only tab). It contains a field named Database Field containing SSN, an unselected check box labeled Read Only, an unselected check box labeled Use Method Accessing with two grayed out fields, Get Method and Set Method. Below this is an unselected check box labeled Use Default Value When Database Field is Null; and two grayed out fields, Type and Value.

4.5.1.3 Renaming a Mapping

Open the corresponding Java source file and change the name. Then go to the structure/Mappings pane, and the newly named attribute will appear unmapped. Right-click it and select **Map As** to remap it. Then save and regenerate BPEL artifacts.

Keep in mind there are four views, the project view, the table/descriptor view, and the individual attribute/column view you can access from the TopLink Mappings structure window. The Java source view is not exactly a TopLink view, but can be treated as such (when renaming a mapping).

4.5.1.4 Configuring Offline Database Tables

Offline database tables are internal to the TopLink Workbench project. When you run the wizard, a TopLink project is created. When you import tables, they are saved as offline table definitions.

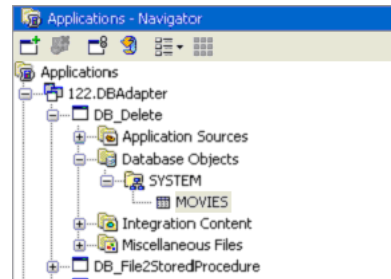
You can use the offline database tables to control the micromapping from database data type to XML data type. If you are using a third-party database, you may need to edit these objects as a workaround. For instance, a *serial* field on a third-party database may need to be mapped as *Integer* so that it is recognized by the wizard

and mapped to `xs:integer`. See ["Problems Importing Third-Party Database Tables"](#) on page A-15 for more information.

Run the wizard once. Then add the following to your JDeveloper BPEL Designer project: `database/schemaName/schemaName.schema`

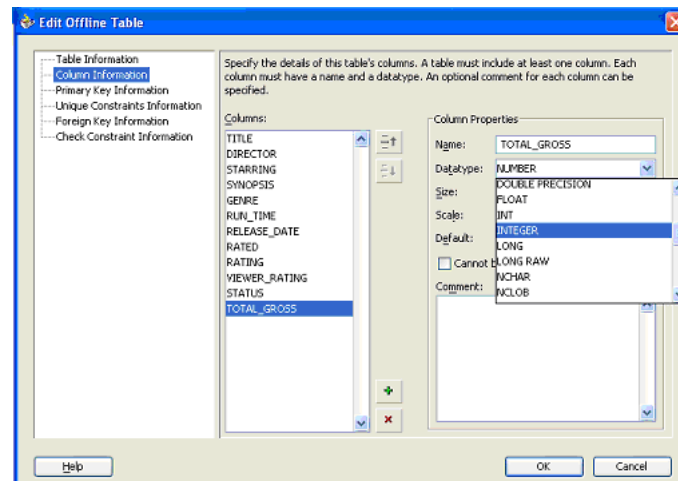
Click the table name (see [Figure 4-69](#)) after it is added to your project and change the types of any of the columns. When you run the wizard again (in edit mode) and click **Finish**, the `toplink_mappings.xml` and XSD file are remapped based on the new database data types.

Figure 4-69 *Configuring Offline Tables*



Edit the offline table in your JDeveloper BPEL Designer project (see [Figure 4-70](#)), not the table reachable from the `ConnectionManager`. If you try the latter, the column types will not be editable, because you are editing the table itself, not an offline representation of it.

Figure 4-70 *Editing Offline Tables*



4.5.2 Relational-to-XML Mappings (`toplink_mappings.xml`)

The database adapter is implemented using TopLink. For every business process, there is an underlying TopLink project, which contains metadata from mapping a database schema to objects/XML.

In TopLink terminology, `toplink_mappings.xml` is an XML deployment file. It is generated from a `.mwp` project file for use at run time. It is recommended that you edit the project in TopLink Workbench and periodically refresh `toplink_mappings.xml`.

The `toplink_mappings.xml` file is the run-time version of the TopLink Workbench project. If you edit this file directly, keep in mind that changes are not reflected in the design-time `toplink_mappings.mwp`. Therefore, any changes are lost when you edit a partner link.

The `toplink_mappings.xml` file consists of a set of descriptors and mappings. Descriptors roughly represent a single table in the database schema, and mappings represent either a single column in the table (direct to field), or a one-to-one or one-to-many relationship to another table (foreign reference).

When modifying the `toplink_mappings.xml` file, the recommended approach is to use TopLink Workbench. The following is an example of a mapping and a descriptor from a `toplink_mappings.xml` file.

```
<mappings>
  <database-mapping>
    <attribute-name>fname</attribute-name>
    <read-only>>false</read-only>
    <field-name>ACTOR.FNAME</field-name>
    <attribute-classification>java.lang.String</attribute-classification>
    <type>oracle.toplink.mappings.DirectToFieldMapping</type>
  </database-mapping>
```

and:

```
<descriptor>
  <java-class>BusinessProcess.Actor</java-class>
  <tables>
    <table>ACTOR</table>
  </tables>
  <primary-key-fields>
    <field>ACTOR.ID</field>
    <field>ACTOR.PROGRAM_ID</field>
    <field>ACTOR.PROGRAM_TYPE</field>
  </primary-key-fields>
```

However, the recommended approach is to work from the TopLink Workbench.

Useful attributes on foreign reference mappings (one-to-one, one-to-many) include:

- `<privately-owned>>false/true`

If a relationship is privately owned, that means that any target rows are deleted whenever any source rows are deleted.

This is important for one-to-many relationships because, if you remove Dept without first deleting its Emp rows, you get a 'child records found' constraint exception.

If you set `privately-owned` to true, the database adapter automatically deletes child records before deleting source rows. In XML everything is assumed to be privately owned; therefore, this tag is set to true by default.

- `<uses-batch-reading>>false/true` and `<uses-joining>>false/true`

There are two key optimizations in relation to reading rows with detail rows from the database.

The following shows the series of selects that TopLink uses to read two department objects (1 and 2), and their employees:

Unoptimized:

```
SELECT DEPT_COLUMNS FROM DEPT WHERE (subQuery)
```

```
SELECT EMP_COLUMNS FROM EMP WHERE (DEPTID = 1)
SELECT EMP_COLUMNS FROM EMP WHERE (DEPTID = 2)
```

Batch Reading:

```
SELECT DEPT_COLUMNS FROM DEPT WHERE (subQuery)
SELECT EMP_COLUMNS FROM EMP e, DEPT d WHERE ((subQuery) AND (e.DEPTID =
d.DEPTID))
```

Joined Reading:

```
SELECT DEPT_COLUMNS, EMP_COLUMNS FROM DEPT d, EMP e WHERE ((subQuery) AND
e.DEPTID = d.DEPTID))
```

Joined reading appears to be the more advanced, but only works for one-to-one mappings currently, and the detail record cannot be null because the join is not an outer join.

Therefore, by default, batch reading is enabled, but not joined reading. This can easily be reversed to improve performance.

If you specify raw SQL for a query, that query cannot be a batched or joined read. To use batched or joined reading, you must not use raw SQL.

You can set other properties in `toplink_mappings.xml`.

4.5.3 The Service Definition (WSDL)

The WSDL generated by the Adapter Configuration Wizard defines the adapter service. This WSDL specifies the various operations exposed by the service. [Table 4–10](#) specifies the operations that are generated based on your selection in the wizard.

Table 4–10 WSDL Operations Generated by the Adapter Configuration Wizard

Adapter Configuration Wizard Selection	Generated WSDL Operation
Insert or Update	insert, update, merge, write, queryByExample
Delete	delete, queryByExample
Select	serviceNameSelect, queryByExample
Poll for New or Changed Records in a Table	receive

Of the preceding operations, `receive` is associated with a BPEL `receive` activity, whereas the rest of the preceding operations are associated with a BPEL `invoke` activity.

See ["SQL Operations as Web Services"](#) on page 4-13 for more information on the preceding operations.

This section discusses the database adapter-specific parameters in the generated WSDL. This is intended for advanced users who want information about all the parameters in the generated WSDL.

A given database adapter service is meant for either continuous polling of a data source (translates to a JCA Activation) or for performing a one-time DML operation (translates to a JCA Interaction). In the continuous polling case, the WSDL contains only one receive operation with a corresponding activation spec defined in the binding section. In the one-time DML operation case, the WSDL contains multiple operations, all of which have a corresponding interaction spec defined in the binding section.

Table 4–11 specifies the JCA Activation/Interaction spec associated with each of the preceding operations:

Table 4–11 Operation and JCA Activation/Interaction Spec

WSDL Operation	JCA Activation/Interaction Spec
insert, update, merge, write, delete	oracle.tip.adapter.db.DBWriteInteractionSpec
select, queryByExample	oracle.tip.adapter.db.DBReadInteractionSpec
receive	oracle.tip.adapter.db.DBActivationSpec

4.5.3.1 DBWriteInteractionSpec

The following code example shows the binding section corresponding to the movie service to write to the Movies table:

```
<binding name="movie_binding" type="tns:movie_ptt">
  <jca:binding />
  <operation name="merge">
    <jca:operation
      InteractionSpec="oracle.tip.adapter.db.DBWriteInteractionSpec"
      DescriptorName="BPELProcess1.Movies"
      DmlType="merge"
      MappingsMetaDataURL="toplink_mappings.xml" />
    <input/>
  </operation>
  <operation name="insert">
    <jca:operation
      InteractionSpec="oracle.tip.adapter.db.DBWriteInteractionSpec"
      DescriptorName="BPELProcess1.Movies"
      DmlType="insert"
      MappingsMetaDataURL="toplink_mappings.xml" />
    <input/>
  </operation>
  <operation name="update">
    <jca:operation
      InteractionSpec="oracle.tip.adapter.db.DBWriteInteractionSpec"
      DescriptorName="BPELProcess1.Movies"
      DmlType="update"
      MappingsMetaDataURL="toplink_mappings.xml" />
    <input/>
  </operation>
  <operation name="write">
    <jca:operation
      InteractionSpec="oracle.tip.adapter.db.DBWriteInteractionSpec"
      DescriptorName="BPELProcess1.Movies"
      DmlType="write"
      MappingsMetaDataURL="toplink_mappings.xml" />
    <input/>
  </operation>
  <operation name="delete">
    <jca:operation
      InteractionSpec="oracle.tip.adapter.db.DBWriteInteractionSpec"
      DescriptorName="BPELProcess1.Movies"
      DmlType="delete"
      MappingsMetaDataURL="toplink_mappings.xml" />
    <input/>
  </operation>
</binding>
```

```
</binding>
```

Table 4–12 describes the DBWriteInteractionSpec parameters:

Table 4–12 DBWriteInteractionSpec Parameters

Parameter	Description	Mechanism to Update
DescriptorName	Indirect reference to the root database table that is being written to	Wizard updates automatically. Do <i>not</i> modify this manually.
DmlType	The DML type of the operation (insert, update, merge, write)	Wizard updates automatically. Do <i>not</i> modify this manually.
MappingsMetaDataURL	Reference to file containing relational-to-XML mappings (toplink_mappings.xml)	Wizard updates automatically. Do <i>not</i> modify this manually.

4.5.3.2 DBReadInteractionSpec

The following code example corresponds to the movie service to query the Movies table:

```
<binding name="movie_binding" type="tns:movie_ptt">
  <jca:binding />
  <operation name="movieSelect">
    <jca:operation
      InteractionSpec="oracle.tip.adapter.db.DBReadInteractionSpec"
      DescriptorName="BPELProcess1.Movies"
      QueryName="movieSelect"
      MappingsMetaDataURL="toplink_mappings.xml" />
    <input/>
  </operation>
  <operation name="queryByExample">
    <jca:operation
      InteractionSpec="oracle.tip.adapter.db.DBReadInteractionSpec"
      DescriptorName="BPELProcess1.Movies"
      IsQueryByExample="true"
      MappingsMetaDataURL="toplink_mappings.xml" />
    <input/>
  </operation>
</binding>
```

Table 4–13 describes the DBReadInteractionSpec parameters:

Table 4–13 DBReadInteractionSpec Parameters

Parameter	Description	Mechanism to Update
DescriptorName	Indirect reference to the root database table that is being queried	Wizard updates automatically. Do <i>not</i> modify this manually.
QueryName	Reference to the SELECT query inside the relational-to-XML mappings file	Wizard updates automatically. Do <i>not</i> modify this manually.
IsQueryByExample	Indicates if this query is a queryByExample or not	Wizard updates automatically. Do <i>not</i> modify this manually. This parameter is needed for queryByExample only.
MappingsMetaDataURL	Reference to file containing relational-to-XML mappings (toplink_mappings.xml)	Wizard updates automatically. Do <i>not</i> modify this manually.

4.5.3.3 DBActivationSpec

The following code example shows the binding section corresponding to the `MovieFetch` service to poll the `Movies` table using `DeletePollingStrategy`:

```
<binding name="MovieFetch_binding" type="tns:MovieFetch_ptt">
  <pc:inbound_binding/>
  <operation name="receive">
    <jca:operation
      ActivationSpec="oracle.tip.adapter.db.DBActivationSpec"
      DescriptorName="BPELProcess1.Movies"
      QueryName="MovieFetch"
      PollingStrategyName="DeletePollingStrategy"
      MaxRaiseSize="1"
      MaxTransactionSize="unlimited"
      PollingInterval="5"
      MappingsMetaDataURL="toplink_mappings.xml" />
    </input/>
  </operation>
</binding>
```

Table 4–14 describes the `DBActivationSpec` parameters:

Table 4–14 DBActivationSpec Parameters

Parameter	Description	Mechanism to Update
<code>DescriptorName</code>	Indirect reference to the root database table that is being queried	Wizard updates automatically. Do <i>not</i> modify this manually.
<code>QueryName</code>	Reference to the <code>SELECT</code> query inside the relational-to-XML mappings file	Wizard updates automatically. Do <i>not</i> modify this manually.
<code>PollingStrategyName</code>	Indicates the polling strategy to be used	Wizard updates automatically. Do <i>not</i> modify this manually.
<code>PollingInterval</code>	Indicates how often to poll the root database table for new events (in seconds)	Wizard updates automatically. Do <i>not</i> modify this manually.
<code>MaxRaiseSize</code>	Indicates the maximum number of database records that can be raised at a time to the BPEL engine	Modify manually in the generated WSDL.
<code>MaxTransactionSize</code>	Indicates the maximum number of rows to process as part of one database transaction	Modify manually in the generated WSDL.
<code>MappingsMetaDataURL</code>	Reference to file containing relational-to-XML mappings (<code>toplink_mappings.xml</code>)	Wizard updates automatically. Do <i>not</i> modify this manually.

The following code example is the binding section corresponding to the `MovieFetch` service to poll the `Movies` table using `LogicalDeletePollingStrategy`:

```
<binding name="PollingLogicalDeleteService_binding"
  type="tns:PollingLogicalDeleteService_ptt">
  <pc:inbound_binding/>
  <operation name="receive">
    <jca:operation
      ActivationSpec="oracle.tip.adapter.db.DBActivationSpec"
      DescriptorName="PollingLogicalDeleteStrategy.Movies"
      QueryName="PollingLogicalDeleteService"
      PollingStrategyName="LogicalDeletePollingStrategy"
      MarkReadFieldName="DELETED"
      MarkReadValue="TRUE"
      MarkReservedValue="MINE"
      MarkUnreadValue="FALSE"
    </input/>
  </operation>
</binding>
```

```

        MaxRaiseSize="1"
        MaxTransactionSize="unlimited"
        PollingInterval="10"
        MappingsMetaDataURL="toplink_mappings.xml" />
    </input>
</operation>
</binding>

```

Table 4–15 describes all of the additional DBActivationSpec parameters for LogicalDeletePollingStrategy:

Table 4–15 DBActivationSpec Parameters for LogicalDeletePollingStrategy

Parameter	Description	Mechanism to Update
MarkReadFieldName	Specifies the database column to use to mark the row as read	Wizard updates automatically. Do not modify this manually.
MarkReadValue	Specifies the value to which the database column is set to mark the row as read	Wizard updates automatically. Do <i>not</i> modify this manually.
MarkReservedValue	Specifies the value to which the database column is set to mark the row as reserved. This parameter is optional. You can use it when multiple adapter instances are providing the same database adapter service.	Wizard updates automatically. Do <i>not</i> modify this manually.
MarkUnreadValue	Specifies the value to which the database column is set to mark the row as unread. This parameter is optional. Use it when you want to indicate specific rows that the database adapter must process.	Wizard updates automatically. Do <i>not</i> modify this manually.

The following code example shows the binding section corresponding to the MovieFetch service to poll the Movies table using SequencingPollingStrategy:

```

<binding name="PollingLastReadIdStrategyService_binding"
    type="tns:PollingLastReadIdStrategyService_ptt">
  <pc:inbound_binding/>
  <operation name="receive">
    <jca:operation
      ActivationSpec="oracle.tip.adapter.db.DBActivationSpec"
      DescriptorName="PollingLastReadIdStrategy.Movies"
      QueryName="PollingLastReadIdStrategyService"
      PollingStrategyName="SequencingPollingStrategy"
      SequencingFieldName="SEQUENCENO"
      SequencingTableNameFieldValue="MOVIES"
      SequencingTableName="PC_SEQUENCING"
      SequencingTableNameFieldName="TABLE_NAME"
      SequencingTableValueFieldName="LAST_READ_ID"
      MaxRaiseSize="1"
      MaxTransactionSize="unlimited"
      PollingInterval="10"
      MappingsMetaDataURL="toplink_mappings.xml" />
    </input>
  </operation>
</binding>

```

Table 4–16 describes all of the additional DBActivationSpec parameters for SequencingPollingStrategy:

Table 4–16 DBActivationSpec Parameters for SequencingPollingStrategy

Parameter	Description	Mechanism to update
SequencingFieldName	Specifies the database column that is monotonically increasing	Wizard updates automatically. Do <i>not</i> modify this manually.
SequencingFieldType	Specifies the type of the database column that is monotonically increasing. This parameter is optional. Use it if the type is not NUMBER.	Wizard updates automatically. Do <i>not</i> modify this manually.
SequencingTableNameFieldValue	Specifies the root database table for this polling query	Wizard updates automatically. Do <i>not</i> modify this manually.
SequencingTableName	Name of the database table that is serving as the helper table	Wizard updates automatically. Do <i>not</i> modify this manually.
SequencingTableNameFieldName	Specifies the database column in the helper table that is used to store the root database table name	Wizard updates automatically. Do <i>not</i> modify this manually.
SequencingTableValueFieldName	Specifies the database column in the helper table that is used to store the sequence number of the last processed row in the root database table name	Wizard updates automatically. Do <i>not</i> modify this manually.

See [Deployment](#) for details about the service section of the WSDL.

4.5.4 XML Schema Definition (XSD)

From a database schema, the wizard generates an XML schema representation of that object. This schema is used by the BPEL process.

For example, from the table named `Movies`, the following is generated:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<xs:schema
targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/db/top/SelectAllByTitle"
xmlns="http://xmlns.oracle.com/pcbpel/adapter/db/top/SelectAllByTitle"
elementFormDefault="unqualified" attributeFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="MoviesCollection" type="MoviesCollection"/>
  <xs:element name="Movies" type="Movies"/>
  <xs:complexType name="MoviesCollection">
    <xs:sequence>
      <xs:element name="Movies" type="Movies" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Movies">
    <xs:sequence>
      <xs:element name="director" type="xs:string" minOccurs="0"
        nillable="true"/>
      <xs:element name="genre" type="xs:string" minOccurs="0" nillable="true"/>
      <xs:element name="rated" type="xs:string" minOccurs="0" nillable="true"/>
      <xs:element name="rating" type="xs:string" minOccurs="0"
        nillable="true"/>
      <xs:element name="releaseDate" type="xs:dateTime" minOccurs="0"
        nillable="true"/>
      <xs:element name="runTime" type="xs:double" minOccurs="0"
        nillable="true"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```



```

        <xs:element name="starring" type="xs:string" minOccurs="0"
            nillable="true"/>
        <xs:element name="status" type="xs:string" minOccurs="0"
            nillable="true"/>
        <xs:element name="synopsis" type="xs:string" minOccurs="0"
            nillable="true"/>
        <xs:element name="title" type="xs:string"/>
        <xs:element name="totalGross" type="xs:double" minOccurs="0"
            nillable="true"/>
        <xs:element name="viewerRating" type="xs:string" minOccurs="0"
            nillable="true"/>
    </xs:sequence>
</xs:complexType>
<xs:element name="findAllInputParameters" type="findAll"/>
<xs:complexType name="findAll">
    <xs:sequence/>
</xs:complexType>
<xs:element name="SelectAllByTitleServiceSelect_titleInputParameters"
type="SelectAllByTitleServiceSelect_title"/>
<xs:complexType name="SelectAllByTitleServiceSelect_title">
    <xs:sequence>
        <xs:element name="title" type="xs:string" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
</xs:complexType>
</xs:schema>

```

This is a generated file. Changes to this file do not affect the behavior of the adapter. It is a declaration of the XML file that the database adapter produces and consumes.

You may need to modify the XSD file if you update the underlying `toplink_mappings.xml`. In that case, regenerate both files by rerunning the Adapter Configuration Wizard in edit mode.

The generated XSD flags all elements as optional with `minOccurs=0`, except for the primary key attributes, which are mandatory.

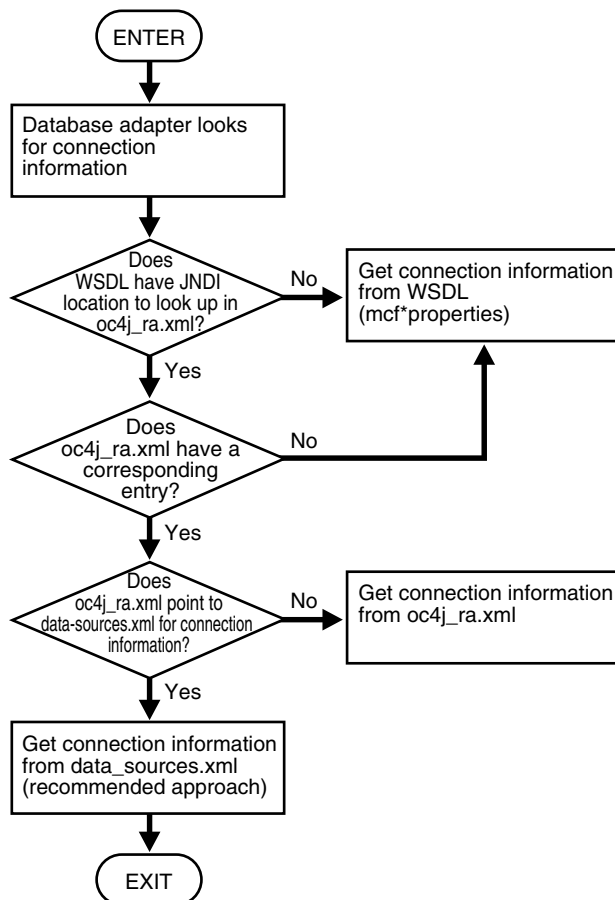
Note: Do not manually modify the XSD file to configure the database adapter.

4.5.5 Deployment

After you define the database adapter service and complete the design of your BPEL process, you compile and deploy the process to Oracle BPEL Server. In the Applications Navigator of JDeveloper BPEL Designer, right-click the BPEL project and select **Deploy**. See Chapter 19, "BPEL Process Deployment and Domain Management" in *Oracle BPEL Process Manager Developer's Guide* for more information about deployment.

4.5.5.1 How the Database Adapter Gets Connection Information

Figure 4-71 shows how the database adapter gets connection information from the three possible sources: the WSDL file, the `oc4j-ra.xml` file, or the `data-sources.xml` file, as described in the following sections.

Figure 4–71 How the Database Adapter Gets Connection Information

4.5.5.2 Out-of-the-box Deployment

When you deploy a process using the Database Adapter, the nature of the deployment varies based on whether a separate database adapter instance is already deployed and available on the application server. For information on production deployment, see [Production Deployment](#).

Note: In the out of the box deployment, when you run the DBAdapter wizard, connection information is required to import tables from the database, as shown in [Example 4–1](#). As a convenience feature, when the `wsdl` file is written it includes the connection information that was used by the design time. This way you can deploy the business process immediately, as the `wsdl` has all the information it needs to execute properly.

Out of box deployment is used only until production deployment is correctly setup. For production you may remove this connection information (`mcf.*properties`) in the `wsdl` keeping only the location attribute, which is required for production deployment. It is mandatory that you use production deployment for production.

[Example 4–1](#) shows database connection information specified in the `mcf.*properties` of the WSDL.

Example 4-1 WSDL Code Example Showing Database Connection Information in the mcf.* Properties

```

<!-- Your runtime connection is declared in
J2EE_HOME/application-deployments/default/DbAdapter/oc4j-ra.xml.
These 'mcf' properties here are from your design time connection and save you from
having to edit that file and restart the application server if eis/DB/scott is
missing.
These 'mcf' properties are safe to remove.
-->
<service name="get">
  <port name="get_pt" binding="tns:get_binding">
    <jca:address location="eis/DB/scott"
      UIConnectionName="scott"
      ManagedConnectionFactory="oracle.tip.adapter.db.DBManagedConnectionFactory"
      mcf.DriverClassName="oracle.jdbc.driver.OracleDriver"
      mcf.PlatformClassName="oracle.toplink.oraclespecific.Oracle9Platform"
      mcf.ConnectionString="jdbc:oracle:thin:@mypc.home.com:1521:orcl"
      mcf.UserName="scott"
      mcf.Password="7347B141D0FBCEA077C118A5138D02BE"
    />
  </port>
</service>

```

Note: When the database adapter uses connection information specified in the WSDL, you have no connection pooling.

4.5.5.3 Production Deployment

Please see [Section 4.1.2, "Design Overview"](#). During design time, you specified something like `eis/DB/<JdevConnectionName>` on the *Adapter Configuration Wizard: Service Connection* page. For production deployment a Database adapter instance with that JNDI name needs to be already deployed and available on the Application Server. This deployment section provides the information to do that.

Runtime instances are configured in the database adapter's `oc4j-ra.xml` file, similar to how connection pools are configured in `data-sources.xml`. These in turn refer through JNDI to the name of a connection pool in `data-sources.xml`.

For the Oracle BPEL Process Manager, `oc4j-ra.xml` is located at

```

Oracle_Home\bpel\system\appserver\oc4j\j2ee\home\
application-deployments\default\DbAdapter\oc4j-ra.xml

```

For the Oracle BPEL Process Manager for OracleAS Middle Tier installation, `oc4j-ra.xml` is located at

```

Oracle_Home\j2ee\OC4J_BPEL\application-deployments\default\DBAdapter\oc4j-ra.xml

```

Connection information in the `oc4j-ra.xml` file is not generated by the Adapter Configuration Wizard. You must manually update this file and restart Oracle BPEL Server for updates to `oc4j-ra.xml` to take effect.

The Application Server manages and pools connections for all applications. It provides cross-application sharing of a single connection pool, high availability, special RAC configuration, XA/transaction support, password indirection, and a single configuration file, `data-sources.xml`. Hence the DB, AQ, and JMS adapters should not contain connection information in their `oc4j-ra.xml` files but simply refer to a data source by name. This is the recommended and default approach in 10.1.3.1.

Note: The WSDL points to a JCA adapter (that is, `eis/DB/BPELSamples` in `oc4j-ra.xml`), which in turn points to a data source (that is, `jdbc/BPELSamples` in `data-sources.xml`). A common pitfall is to set the JNDI name in the wizard directly to the name of a data source.

The deployment descriptor file (`oc4j-ra.xml`) has four key properties: `location`, `xDataSourceName`, `dataSourceName`, and `platformClassName`.

Location

You may remember that when you ran the wizard you could enter a runtime JNDI name, which defaulted to `eis/DB/<JdeveloperConnectionName>`. Ensure that a corresponding entry exists in the `oc4j-ra.xml` deployment descriptor by production time.

xDataSourceName and nonXaDataSourceName

`xDataSourceName` and `dataSourceName` refer to global transaction and local transaction data sources, respectively.

For the Oracle BPEL Process Manager, `data-sources.xml` is at

```
Oracle_Home\bpel\system\appserver\oc4j\j2ee\home\
config\data-sources.xml
```

For the Oracle BPEL Process Manager for OracleAS Middle Tier installation, `data-sources.xml` is at

```
Oracle_Home\j2ee\OC4J_BPEL\config\data-sources.xml
```

If you have the following code in `data-sources.xml`,

```
<!-- Connection pool for oracle lite -->

<connection-pool name="BPELPM_CONNECTION_POOL">
  <connection-factory factory-class="oracle.lite.poljdbc.POLJDBCdriver"
    user="system"
    password="any"
    url="jdbc:polite4@127.0.0.1:100:orabpel" />
</connection-pool>

<managed-data-source name="BPELServerDataSource"
  connection-pool-name="BPELPM_CONNECTION_POOL"
  jndi-name="jdbc/BPELServerDataSource" tx-level="global"/>

<managed-data-source name="BPELServerDataSourceWorkflow"
  connection-pool-name="BPELPM_CONNECTION_POOL"
  jndi-name="jdbc/BPELServerDataSourceWorkflow" tx-level="local"/>
```

then the `oc4j-ra.xml` entry would look like the following:

```
<connector-factory location="eis/DB/BPELSamples" connector-name="Database
Adapter">
<config-property name="xDataSourceName" value="jdbc/BPELServerDataSource"/>
<config-property name="dataSourceName"
  value="jdbc/BPELServerDataSourceWorkflow"/>
...
```

It is recommended that you always use the `xDataSourceName` (pointing to a `tx-level="global"` data source). If you specify an `xDataSourceName` and a `dataSourceName` (pointing to a `tx-level="local"` data source), then the `dataSourceName` may occasionally be used for reading performance (for multiple inbound polling threads). Only if you specify `dataSourceName` will the DB Adapter not synchronize with global transactions.

PlatformClassName

This indicates whether you are connecting to Oracle, DB2, SQLServer or any other database.

[Table 4–17](#) shows the advanced properties, which are database platform variables. Set the `DatabasePlatform` name to one of the following variables.

Table 4–17 Database Platform Names

Database	PlatformClassName
Oracle9+ (including 10g)	<code>oracle.toplink.platform.database.Oracle9Platform</code>
Oracle9+ (optional): To workaround padding of CHAR (vs. VARCHAR2) values on select, see Outbound SELECT on a CHAR(X) or NCHAR Column Returns No Rows in Appendix A, "Troubleshooting and Workarounds"	<code>oracle.toplink.platform.database.Oracle9Platform</code>
Oracle8	<code>oracle.toplink.platform.database.Oracle8Platform</code>
Oracle7	<code>oracle.toplink.platform.database.OraclePlatform</code>
DB2	<code>oracle.toplink.platform.database.DB2Platform</code>
DB2 on AS400	<code>oracle.tip.adapter.db.toplinkext.DB2AS400Platform</code>
Informix	<code>oracle.toplink.platform.database.InformixPlatform</code>
Sybase	<code>oracle.toplink.platform.database.SybasePlatform</code>
SQLServer	<code>oracle.toplink.platform.database.SQLServerPlatform</code>
Any other database	<code>oracle.toplink.platform.database.DatabasePlatform</code>

4.5.5.4 Advanced Properties of the oc4j-ra.xml File

This section discusses the following advanced properties of the `oc4j-ra.xml` file:

- [Properties Configurable by Using Managed Connection Factory Entry](#)
- [Case for Property Names in ra.xml and oc4j-ra.xml](#)
- [Editing oc4j-ra.xml Through Oracle Enterprise Manager](#)

Properties Configurable by Using Managed Connection Factory Entry

The following properties are configurable by using the managed connection factory entry in the `oc4j-ra.xml` file:

```
String connectionString
String userName
String password
String encryptionClassName
Integer minConnections
Integer maxConnections
```

```
Boolean useReadConnectionPool
Integer minReadConnections
Integer maxReadConnections
String dataSourceName
String driverClassName
Integer cursorCode
String databaseName
String driverURLHeader
Integer maxBatchWritingSize
String platformClassName
String sequenceCounterFieldName
String sequenceNameFieldName
Integer sequencePreallocationSize
String sequenceTableName
String serverName
Boolean shouldBindAllParameters
Boolean shouldCacheAllStatements
Boolean shouldIgnoreCaseOnFieldComparisons
Boolean shouldForceFieldNamesToUpperCase
Boolean shouldOptimizeDataConversion
Boolean shouldTrimStrings
Integer statementCacheSize
Integer stringBindingSize
String tableQualifier
Integer transactionIsolation
Boolean usesBatchWriting
Boolean usesByteArrayBinding
Boolean usesDirectDriverConnect
Boolean usesExternalConnectionPooling
Boolean usesExternalTransactionController
Boolean usesJDBCBatchWriting
Boolean usesNativeSequencing
Boolean usesNativeSQL
Boolean usesStreamsForBinding
Boolean usesStringBinding
```

The following properties appear in the `oracle.toplink.sessions.DatabaseLogin` object.

See `TopLink` API reference information on `DBConnectionFactory` Javadoc and `DatabaseLogin` Javadoc at

http://download-east.oracle.com/docs/cd/B10464_02/web.904/b10491/index.html

To configure any of the preceding properties:

1. Add the following to the `ra.xml` file:

```
<config-property>
  <config-property-name>usesJDBCBatchWriting</config-property-name>
  <config-property-type>java.lang.Boolean</config-property-type>
  <config-property-value>true</config-property-value>
</config-property>
```

For Oracle BPEL Process Manager, `ra.xml` is at

```
Oracle_Home\bpel\system\appserver\oc4j\j2ee\home\connectors\
DbAdapter\DbAdapter\META-INF\ra.xml
```

For Oracle BPEL Process Manager for OracleAS Middle Tier, `ra.xml` is at

```
Oracle_Home\j2ee\OC4J_BPEL\connectors\DbAdapter\DbAdapter\META-INF\ra.xml
```

2. Add the following to the `oc4j-ra.xml` file:

```
<config-property name="usesJDBCBatchWriting" value="true"/>
```

3. Restart Oracle BPEL Server for the changes to take effect.

You can also update the default `oc4j-ra.xml` and `ra.xml` files before you deploy the database adapter. This way, you deploy once and do not need to restart the application server.

Case for Property Names in `ra.xml` and `oc4j-ra.xml`

The case for all property names must exactly match in the `ra.xml` and `oc4j-ra.xml` files. Otherwise, you receive an error message similar to the following during run time in the `domain.log` file:

```
Type=Dequeue_ptt, operation=Dequeue
<2005-03-14 15:20:43,484> <ERROR> <default.collaxa.cube.activation>
<AdapterFramework::Inbound> Error while performing endpoint Activation: ORABPEL-12510<br>
Unable to locate the JCA Resource Adapter via WSDL port element jca:address.
The Adapter Framework is unable to startup the Resource Adapter specified in
the WSDL jca:address element:
@ {http://xmlns.oracle.com/pcbpel/wsd/jca/}address:
location='eis/aqSample'
```

For example, if the `userName` property in the `Oracle_Home\bpel\system\appserver\oc4j\j2ee\home\application-deployment\default\AqAdapter\oc4j-ra.xml` file for the AQ adapter uses the following upper and lower case convention:

```
<config-property name="userName" value="scott"/>
```

Then this case must match the `userName` property in the corresponding `Oracle_Home\bpel\system\appserver\oc4j\j2ee\home\connectors\default\AqAdapter\AqAdapter\META-INF\ra.xml` file for the AQ adapter.

```
<config-property-name>userName</config-property-name>
```

Editing `oc4j-ra.xml` Through Oracle Enterprise Manager

You can deploy and undeploy resource adapters from Oracle Enterprise Manager, as well. The following steps show how you can edit both, `data-sources.xml` and `oc4j-ra.xml` through the Oracle Enterprise Manager:

1. Start the BPEL server process.
2. After the installation, open Windows Services.
3. Change the Oracle-`nnn`ProcessManager start up option to manual (where, `nnn` is the name you gave your instance). Next time you reboot, OPMN won't be started. To start it, follow these steps:

- a. Click **Start**, and then click **Run**.
- b. Enter `cmd`, and then click **Ok** to bring up a command window.
- c. Go to your installation, by typing the following command:

```
cd %SOAHOME%\opmn\bin
```

Ensure that you replace `%SOAHOME%` with your path. For example,

```
cd c:\oracle\product\soabeta\opmn\bin
```

- d. Enter the following command:

```
opmnctl startall
```

- e. Check to see if the services have started up correctly, by entering the following command:

```
opmnctl status
```

Note: To stop all of the services, if you ever need to, type:

```
opmnctl stopall
```

4. Log into Enterprise Manager by entering the following URL:

<http://<host>:<port>/em>

The default port is 8888; the default login userid is `oc4jadmin` and password is `welcome1`.

5. Create a connection pool and a data source, by using the following steps:

- a. Select the **Home** link.
- b. Select the **Administration** link.
- c. Click the **Go To Task** icon for **Services/JDBC Resources**.
- d. Under **Connection Pools**, click the **Create** button.
- e. Accept defaults, and then click **Continue**.
- f. Enter the information shown in the following table:

Field	Value
Name	appsSample_pool
JDBC URL	The URL for your database. For example: jdbc:oracle:thin:@<host>:1521:<sid>
Username	apps
Password	apps

Accept defaults for the rest of the fields.

- g. Click **Finish**.
- h. Click the **Test Connection** icon for your new connection pool.
- i. In the new screen, click **Test**.

Back on the main page, a successful connection message is displayed. If you get an error message, check the URL and credentials to ensure you have entered the right information.

- j. Click **Finish**.
- k. Under **Data Sources**, click **Create**.

- l. Accept the defaults, and then click **Continue**.
- m. Enter the information shown in the following table:

Field	Value
Name	appsDemoDS
JNDI Location	jdbc/appsSampleDataSource
Connection Pool	appsSample_pool

Accept defaults for the rest of the fields.

- n. Click **Finish**.
6. Create an Oracle Applications adapter, by using the following steps:
- a. At the top of the page, select the **OC4J:home** breadcrumb link.
 - b. Select the **Applications** link.
 - c. In the tree of applications, select the **default** link.
 - d. Under **Modules**, select the **AppsAdapter** link.
 - e. Select the **Connection Factories** link.
 - f. Under **Connection Factories**, click **Create**.

Note that you must use the **Create** button near the top of the screen, and not the one in the Shared Connection Pools section.

- g. Accept all the defaults, and then click **Continue**.
- h. For JNDI Location, enter `eis/Apps/appsSample`.
- i. Under **Configuration Properties**, For **xaDataSourceName**, enter `jdbc/appsSampleDataSource`.

Keep the default entries for all of the other fields.

- j. Click **Finish**.

4.5.5.5 Comparison: Pre-10.1.3 and Post-10.1.3

In Oracle BPEL Process Manager 10.1.3, the deployment descriptor no longer contains local connection information, such as `driverClassName`, `userName`, `password`, `connectionUrl`, `minConnections`, `maxConnections`, `minReadConnections`, `maxReadConnections`. Instead, the deployment descriptor points to another file, the `data-sources.xml`. This is to leave connection configuration and pooling to the app server. Also an application server connection pool must be used to support XA (commits to multiple databases in the same transaction) which is highly desirable.

Hence these 'TopLink Internal pooling' properties have been removed but they can still be readded to the `ra.xml` and used as advanced properties.

XA and application server pooling was supported pre 10.1.3 with a combination of `dataSourceName`, `usesExternalConnectionPooling`, and `usesExternalTransactionController`. These have been removed and replaced with `xaDataSourceName` and `nonXaDataSourceName` only.

Table shows the properties that should be changed when migrating to Oracle BPEL Process Manager 10.1.3

Table 4–18 Properties Used in Oracle BPEL Process Manager 10.1.3

Pre 10.1.3	Post 10.1.3
dataSourceName is non-empty; usesExternalConnectionPooling = true; useExternalTransactionController = true	Specify xDataSourceName, optionally add dataSourceName to take advantage of new read connection pooling support in the non-TopLink use case
dataSourceName is non-empty; usesExternalConnectionPooling = true; useExternalTransactionController = false	specify only dataSourceName
dataSourceName is non-empty; usesExternalConnectionPooling = false; useExternalTransactionController = false	set xDataSourceName and dataSourceName to empty strings. Re-add local connection info (driverClassName, userName, password, connectionUrl, minConnections, maxConnections, minReadConnections, maxReadConnections) as advanced properties.

4.5.6 Performance

The database adapter is preconfigured with many performance optimizations. You can, however, make some changes to reduce the number of round trips to the database, as described in the following sections.

4.5.6.1 Use Cases for Performance

Performance tuning is demonstrated in the following tutorials:

- DirectSQLPerformance
- MultiTablesPerformance
- ../polling/DistributedPolling

In 10.1.3.1 a new code path was added, which for simple scenarios, such as delete polling strategy, insert, flat tables, was heavily optimized for maximum performance. Hence there are no two main performance samples, one for the direct SQL and another for the more feature-rich adapter as a whole. The adapter will auto-detect when it is in a configuration that permits the slightly faster code path.

For these files, go to

`Oracle_Home\bpel\samples\tutorials\122.DBAdapter\advanced\endToEnd`

4.5.6.2 Performance Hit List

The following performance hit list was taken from the MultiTablesPerformance README. It contains a best configuration and observations on the relative impacts of the various performance options. Single change variations from *ideal* (280 rows per second throughput) configuration:

1. **Oracle 10g Windows Database on Other Machine (ping time about 40 ms) (vs. db on same machine):** 70 rows / second (10,000 rows) -75% degradation vs +300% improvement
2. **MaxTransactionSize/MaxRaiseSize=2/2 (vs. 100/100):** 80 rows / second (10,000 rows) -71% / +250%
3. **batch-reading=false (vs true in toplink_mappings.xml):** 104 rows / second (10,000 rows) -63% / +169%

4. **DeletePollingStrategy (vs. LogicalDelete, NumberOfThreads=1):** 120 rows / second (10,000 rows) -57% / +133%
5. **usesBatchWriting=false (vs true in oc4j-ra.xml):** 127 rows / second (10,000 rows) -55% / +120%
6. **OptimizeMerge="false" (vs true):** 175 rows / second (10,000 rows) -38% / +60%
7. **BPEL dehydration on (vs off):** 222 rows / second (10,000 rows) -21% / +26%
8. **NumberOfThreads=1 (vs 6):** 227 rows / second (10,000 rows) * -19% / +23%

Note: *Ideal* is exactly the same configuration as best possible configuration for DeletePollingStrategy, where throughput was 120 rows/second.

9. **No primary / foreign keys on database (vs. specified):** 270 rows / second (10,000 rows) -4% / +4% (267 rows / second (20,000 rows))
10. **Insert (vs Merge):** 303 rows / second (10,000 rows) -8% / +8%
11. **UseBatchDestroy="false" (vs. true):** 312 rows / second (10,000 rows) -10% / +11%

The first seven performance hit mentioned in the preceding list are all directly related almost exclusively to the number of roundtrips to the database, and the network cost of each trip. Plus deleting seems to be a very expensive operation, even worse with foreign key constraints NumberOfThreads must be set to 1.

4.5.6.3 Outbound Write: Should You Use Merge, Write, or Insert?

If you run through the Adapter Configuration Wizard and select **Insert or Update**, you get a WSDL with the following operations: `merge` (default), `insert`, `update`, and `write`. The latter three call TopLink queries of the same name, avoiding advanced functionality that you may not need for straightforward scenarios. You can make the change by double-clicking an `invoke` activity and selecting a different operation. The `merge` is the most expensive, followed by the `write` and then the `insert`.

The `merge` first does a read of every element and calculates what has changed. If a row has not changed, it is not updated. The extra reads (for existence) and the complex change calculation add considerable overhead. For simple cases, this can be safely avoided; that is, if you changed only two columns, it does not matter if you update all five anyway. For complex cases, however, the opposite is true. If you have a master record with 100 details, but you changed only two columns on the master and two on one detail, the `merge` updates those four columns on two rows. A `write` does a write of every column in all 101 rows. Also, the `merge` may appear slower, but can actually relieve pressure on the database by minimizing the number of writes.

The `insert` operation is the most performant because it uses no existence check and has no extra overhead. You have no reads, only writes. If you know that you will do an insert most of the time, use `insert`, and catch a unique key constraint SQL exception inside your BPEL process, which can then perform a `merge` or `update` instead. Use `merge` if you have a mix of new and existing objects (for instance, a master row containing several new details). The `update` is similar to `insert`.

To monitor performance, you can enable debug logging and then watch the SQL for various inputs.

4.5.6.4 The TopLink Cache: When Should You Use It?

Caching is an important performance feature of TopLink. However, issues with stale data can be difficult to manage. By default, the database adapter uses a `WeakIdentityMap`, meaning a cache is used only to resolve cyclical references, and entries are quickly reclaimed by the Java virtual machine. If you have no cycles (and you ideally should not for XML), you can switch to a `NoIdentityMap`. The TopLink default is a `SoftCacheWeakIdentityMap`. This means that the most frequently used rows in the database are more likely to appear already in the cache.

For a knowledge article on caching, go to

<http://www.oracle.com/technology/tech/java/newsletter/november04.html>

4.5.6.5 Existence Checking

One method of performance optimization for `merge` is to eliminate check database existence checking. The existence check is marginally better if the row is new, because only the primary key is returned, not the entire row. But, due to the nature of `merge`, if the existence check passes, the entire row must be read anyway to calculate what changed. Therefore, for every row to be updated, you see one extra round trip to the database during `merge`.

It is always safe to use check cache on the root descriptor/table and any child tables if A is master and B is a privately owned child. If A does not exist, B cannot exist. And if A exists, all its Bs are loaded as part of reading A; therefore, check cache works.

4.5.6.6 Inbound Polling: `maxRaiseSize`

This parameter indicates the maximum number of XML records that can be raised at a time to the BPEL engine. For example, if you set `maxRaiseSize = 10`, then 10 database records are raised at one time. On read (inbound) you can set `maxRaiseSize = 0` (unbounded), meaning that if you read 1000 rows, you will create one XML with 1000 elements, which is passed through a single Oracle BPEL Process Manager instance. A `merge` on the outbound side can then take all 1000 in one group and write them all at once with batch writing.

Use the `maxRaiseSize` parameter for publishing large payloads.

4.5.6.7 Inbound Polling: Choosing a Polling Strategy

Your choice of polling strategy matters too. Avoid the delete polling strategy because it must individually delete each row. The sequencing polling strategy can destroy 1000 rows with a single update to a helper table.

4.5.6.8 Relationship Reading (Batch Attribute and Joined Attribute Reading)

Batch reading of one-to-many and one-to-one relationships is on by default. You can also use joined reading for one-to-one relationships instead, which may offer a slight improvement.

4.5.6.9 Connection Pooling

You can configure a connection pool if using either the adapter's local connection pool or an application server data source. Creating a database connection is an expensive operation. Ideally you should only exceed the `minConnections` under heavy loads. If you are consistently using more connections than that at once, then you may spend a lot of time setting up and tearing down connections. The database adapter also has a read connection pool. A read connection is more performant because there is no limit

on how many users can use one connection for reading at the same time, a feature that most JDBC drivers support.

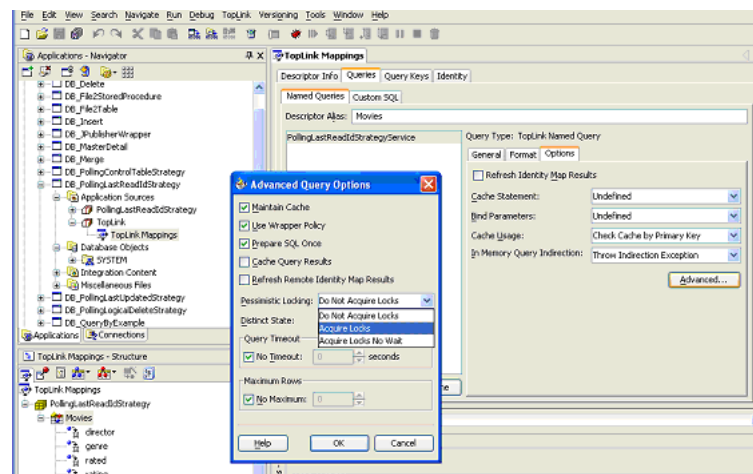
4.5.6.10 Inbound Distributed Polling

The database adapter is designed to scale to the number of unprocessed rows on the database. By default, it is possible to read and process one database row or 10,000 with as little as three round trips to the database. The most expensive operations are limited to a constant number. You can also configure the database adapter for a distributed environment.

Load Balancing: MaxTransactionSize and Pessimistic Locking

You can set a simple option that enables the database adapter to work safely in a distributed environment by making the first polling query acquire locks, as shown in [Figure 4-72](#). In SQL terms, you are making your first `SELECT` into a `SELECT . . . FOR UPDATE`.

Figure 4-72 Acquiring Locks



The behavior of all polling strategies is as follows:

1. Read all unprocessed rows.
2. Process those rows.
3. Commit.

If any adapter instance performs step 1 while another instance is between steps 1 and 3, then duplicate processing occurs. Acquiring locks on the first operation and releasing them in commit solves this problem, and may naturally order the polling instances.

To enable pessimistic locking, run through the wizard once to create an inbound polling query. In the **Applications Navigator** window, expand **Application Sources**, then **TopLink**, and click **TopLink Mappings**. In the **Structure** window, click the table name. In **Diagram View**, click the following tabs: **TopLink Mappings**, **Queries**, **Named Queries**, **Options**; then the **Advanced...** button, and then **Pessimistic Locking** and **Acquire Locks**. You see the message, "Set Refresh Identity Map Results?" If a query uses pessimistic locking, it must refresh the identity map results. Click **OK** when you see the message, "Would you like us to set Refresh Identity Map Results and Refresh Remote Identity Map Results to true?" Run the wizard again to regenerate

everything. In the new `toplink_mappings.xml` file, you see something like this for the query: `<lock-mode>1</lock-mode>`.

Note the following:

- The preceding procedure works in conjunction with every polling strategy where the first operation is a read.
- For the sequencing-based polling strategies, the `SELECT FOR UPDATE` is applied to the `SELECT` on the helper table only. The `SELECT` on the polled table does not acquire locks because you do not have `write` access to those tables.
- If an adapter instance fails while polling records, those records are returned to the unprocessed pool (no commit happens).
- No individual adapter instance is special. In an ideal distributed system, coordination between instances is minimal (here effected with locking). No master acts as a weak link, and every part is identically configured and interchangeable.
- Other than the `SELECT FOR UPDATE`, no extra reads or writes are performed by the database adapter in a distributed environment.

Load Balancing: MaxTransactionSize and Pessimistic Locking

After you enable pessimistic locking on a polling query, the `maxTransactionSize` activation property automatically behaves differently.

Assume that there are 10,000 rows at the start of a polling interval and that `maxTransactionSize` is 100. In standalone mode, a cursor is used to iteratively read and process 100 rows at a time until all 10,000 have been processed, dividing the work into $10,000 / 100 = 100$ sequential transactional units. In a distributed environment, a cursor is also used to read and process the first 100 rows. However, the adapter instance will release the cursor, leaving 9,900 unprocessed rows (or 99 transactional units) for the next polling interval or another adapter instance.

For load balancing purposes, it is dangerous to set the `maxTransactionSize` too low in a distributed environment (where it becomes a speed limit). It is best to set the `maxTransactionSize` close to the per CPU throughput of the entire business process. This way, load balancing occurs only when you need it.

4.5.7 detectOmissions Feature

The following are the features of the `detectOmission` feature:

Available Since

Release 10.1.3

Configurable

Yes

Default Value

Design Time: `true`, unless explicitly set to `false`

Use Case

Users may pass incomplete or partial xml to a merge, update, or insert, and see that every column they left unspecified in the xml is set to null in the database.

It allows `DBAdapter` merge, insert or update to differentiate between null value and the absence of a value (omission) in xml documents. On a case by case basis, it

determines which information in the xml is meaningful and which is not. In this way the xml is seen as a partial representation of a database row, as opposed to a complete representation. The following table lists examples for null values, and values that can be omitted.

Element Type	Omission	Null
Column	<code><director></director></code> <code><director /></code> <code><!-- director>...</director --></code>	<code><director xsi:nil="true" /></code>
1-1	<code><!-- dept> ... </dept --></code>	<code><dept xsi:nil="true" /></code>
1-M	<code><!-- empCollection>...</code> <code></empCollection --></code>	<code></empCollection></code> <code></empCollection> (empty)</code>

Note: The 1-1 representation `<dept />` denotes an empty department object, and should not be used.

For 1-M, `<empCollection />` actually means a collection of 0 elements, and is considered a meaningful value.

For columns, `<director></director>` is not considered an omission in cases where it represents an empty string.

A value considered omitted will be omitted from UPDATE or INSERT sql. For an update operation, existing (meaningful) values on the database will not be overwritten. And for an insert operation, the default value on the database will be used, as no explicit value is provided in the SQL.

A DBAdapter receive will not be able to produce xml with omissions, and makes use of `xsi:nil="true"`. If you are unable to produce input xml with `xsi:nil="true"`, or are concerned about the difference between `<director />` and `<director></director>`, then it is best to set `DetectOmissions="false"` in the wsdl.

To treat all null values as omissions, check out the IgnoreNullsMerge sample, which comes with a custom TopLink plugin. The plugin works similar to this feature, but cannot detect subtleties between null and omission.

When you are expecting an update you can improve performance, by omitting 1-1 and 1-M relationships. Because, the merge operation can skip considering the detail records completely.

Alternatively, map only those columns you are interested in, and create separate mappings for different invokes. If two updates should update two different sets of columns, create two separate partnerlinks.

Performance

By default, xml will not be input to the database adapter containing omissions. Until an xml with omissions is detected, there is no performance overhead. Once omissions are detected, a TopLink descriptor event listener is added. This event listener has some overhead, and every `modifyRow` about to become a `SQLUpdate` or `SQLInsert` needs to be iterated over, to check for omissions. Hence, every column value sent to the database is checked. If the input xml has mostly omissions then the cost overhead should be more than compensated by sending fewer values to the database.

Incompatible Interactions

`DirectSQL="true"` and `DetectOmissions="true"` - `DetectOmissions` takes precedence. The following are some examples for incompatible interactions:

- `DetectOmissionsMerge`
- `IgnoreNullsMerge`
- `OptimizeMerge`

Note: For migrated old BPEL project, you must re-run the DB Adapter wizard in order to regenerate the WSDL file. When you do this, the `DetectOmissions` and `OptimizeMerge` options will appear in the WSDL file with default values as `DetectOmissions="false"` and `OptimizeMerge="true"`.

See the following for more information:

You can also access the forums from Oracle Technology Network at

- The Oracle BPEL Process Manager forum at
<http://forums.oracle.com/forums/forum.jsp?forum=212>
- The TopLink forum at
<http://forums.oracle.com/forums/forum.jsp?forum=48>

This site contains over 2,000 topics, such as implementing native sequencing, optimistic locking, and JTA-managed connection pools with TopLink

<http://www.oracle.com/technology>

4.5.8 OutputCompletedXml Feature

`OutputCompletedXml` is a feature of the outbound `write` activity. The following are some of the features of the `OutputCompletedXml` feature:

Available Since

Release 10.1.2.0.2

Configurable

`OutputCompletedXml` appears in wsdl only when default is `true`.

Default Value

It is `true` when TopLink sequencing is configured to assign primary keys on insert from a database sequence, otherwise it is `false`.

Issue

You can have primary keys auto-assigned on `insert` from a database sequence. However the usefulness of this feature is diminished, because `insert/merge` have no output message, and so there is no way to tell which primary keys were assigned.

Note: After configuring sequencing (link), run the wizard again, so that the `insert/merge wsdl` operations can be regenerated with an output message, and `wsdl` property `OutputCompletedXml="true"`.

Performance

An output xml is only provided when the output xml would be significantly different, so if `TopLink` sequencing is not used, then this feature is disabled and there is no performance hit. Further, this feature can be explicitly disabled. Likewise, the original input xml is updated and returned, a completely new xml is not built. Also only a shallow update of the xml is performed, if primary keys were assigned to detail records these will not be reflected in the output xml.

Incompatible Interactions

`DirectSQL="true"` and `"OutputCompletedXml"` - `OutputCompletedXml` takes precedence.

4.6 Third-Party JDBC Driver and Database Connection Configuration

The following section discusses how to connect to third-party databases. You can use one vendor's database for design time and another for run time because BPEL processes are database-platform neutral.

Note: To create an Oracle Lite database connection, follow the steps for a third-party JDBC driver (because the existing wizard and libraries for the Oracle Lite database require extra configuration). [Table 4–19](#) provides information for connecting to an Oracle Lite database.

See [Problems Importing Third-Party Database Tables](#) for more information.

The following steps generally apply when you create a database connection using a third-party JDBC driver. For specific third-party databases, see the following topics:

- [Using a Microsoft SQL Server](#)
- [Using an IBM DB2 Database](#)
- [Using a Sybase Database](#)
- [Using an InterSystems Caché Database](#)
- [Using a MySQL 4 Database](#)
- [Summary of Third-Party and Oracle Lite Database Connection Information](#)
- [Location of JDBC Driver JAR Files and Setting the Class Path](#)

To create a database connection when using a third-party JDBC driver:

1. Select **Connection Navigator** from **View**.
2. Right-click **Database** and select **New Database Connection**.
3. Click **Next** in the Welcome window.
4. Enter a connection name.
5. Select **Third Party JDBC Driver** from **Connection Type**.

6. Enter your username, password, and role information.
7. Click **New** for **Driver Class**.
8. Enter the driver name (for example, *some.jdbc.Driver*) for **Driver Class**.
9. Click **New** for **Library**.
10. Click **Edit** to add each JAR file of your driver to **Class Path**.
11. Click **OK** twice to exit the Create Library windows.
12. Click **OK** to exit the Register JDBC Driver window.
13. Enter your connection string name for **URL** and click **Next**.
See [Table 4–19](#) for some commonly used URLs. Also, sample entries appear in the deployment descriptor file (`oc4j-ra.xml`).
14. Click **Test Connection**.
15. If the connection is successful, click **Finish**.

4.6.1 Using a Microsoft SQL Server

When using a Microsoft SQL Server database, follow the database connection steps in [Design Time: Using the Command-Line Utility](#) and use the following information:

- [MS JDBC Driver](#)
- [DataDirect Driver](#)

4.6.1.1 MS JDBC Driver

URL:

```
jdbc:microsoft:sqlserver://localhost\NAME:1433;SelectMethod=cursor;database=???
```

Driver Class: `com.microsoft.jdbc.sqlserver.SQLServerDriver`

Driver Jar: `.\SQLServer2000\msbase.jar, msutil.jar, mssqlserver.jar`

4.6.1.2 DataDirect Driver

URL: `jdbc:oracle:sqlserver://localhost`

Driver Class: `com.oracle.ias.jdbc.sqlserver.SQLServerDriver`

Driver Jar: `.\DataDirect\YMbase.jar, YMoc4j.jar, YMutil.jar, YMsqlserver.jar`

Note the following when connecting to a SQL Server database:

- User name and password
 - SQL Server 2005 installs with Windows authentication as the default. Therefore, you do not log in with a user name and password; rather, your Windows user account either has privilege or does not. JDBC requires you to provide a user name and password.

According to `support.microsoft.com`, "Microsoft SQL Server 2000 driver for JDBC does not support connecting by using Windows NT authentication." See

<http://support.microsoft.com/default.aspx?scid=kb;en-us;313100>

However, the DataDirect driver specification states that it does.

If you use your Windows user name and password, you may see something like the following:

```
[Microsoft][SQLServer 2000 Driver for JDBC][SQLServer]Login failed for user
'DOMAIN\USER'. The user is not associated with a trusted SQL Server
connection.[Microsoft][SQLServer 2000 Driver for JDBC]
An error occurred while attempting to log onto the database.
```

You must select `mixed mode authentication` on a clean installation.

- On a Microsoft SQL Server 2000 Express Edition installation, the system username is `sa` and the password is whatever you provide.

- **Connect string**

From the `sqlcmd` login, you can infer your connect string, as in the following examples:

Example 1:

```
sqlcmd
1>
jdbc:microsoft:sqlserver://localhost:1433
```

Example 2:

```
sqlcmd -S user.mycompany.com\SQLEXPRESS
1>
jdbc:microsoft:sqlserver://user.mycompany.com\SQLEXPRESS:1433
```

Example 3:

```
sqlcmd -S user.mycompany.com\SQLEXPRESS -d master
1>
jdbc:microsoft:sqlserver://user.mycompany.com\SQLEXPRESS:1433;databaseName=
master
```

A full URL is as follows:

```
jdbc:microsoft:sqlserver://serverName[\instanceName]:tcpPort[;SelectMethod=curs
or][;databaseName=databaseName]
```

- **Database name**

If you must explicitly supply the database name, but do not know it, go to

```
C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data
```

If you see a file named `master.mdf`, then one of the database names is `master`.

- **TCP port**

Make sure that SQL Server Browser is running and that your SQL Server service has TCP/IP enabled and is listening on static port 1433. Disable dynamic ports. In SQL Native Client Configuration/Client Protocols, make sure that TCP/IP is enabled and that the default port is 1433.

- **JDBC drivers**

You must download the JDBC drivers separately. From www.microsoft.com, click **Downloads** and search on *jdbc*. You can also try using the DataDirect driver.

4.6.2 Using an IBM DB2 Database

When using an IBM DB2 database, follow the database connection steps in [Design Time: Using the Command-Line Utility](#) and use the following information:

4.6.2.1 DataDirect Driver

URL: `jdbc:db2:localhost:NAME`

Driver Class: `COM.ibm.db2.jdbc.net.DB2Driver`

Driver Jar (v8.1): `.\IBM-net\db2java_81.zip, db2jcc_81.jar`

4.6.2.2 JT400 Driver (AS400 DB2)

URL: `jdbc:as400://hostname;translate binary=true`

Driver Class: `com.ibm.as400.access.AS400JDBCdriver`

Driver Jar: `jt400.jar`

For correct character set translation, use `translate binary=true`.

4.6.3 Using a Sybase Database

When using a Sybase database, follow the database connection steps in [Design Time: Using the Command-Line Utility](#) and use the following information:

4.6.3.1 jconn Driver

URL: `jdbc:sybase:Tds:localhost:5001/NAME`

Driver Class: `com.sybase.jdbc2.jdbc.SybDriver`

Driver Jar: `.\Sybase-jconn\jconn2.jar`

4.6.3.2 DataDirect Driver

URL: `jdbc:oracle:sybase://localhost:5001`

Driver Class: `com.oracle.ias.jdbc.sybase.SybaseDriver`

Driver Jar: `.\DataDirect\Ymbase.jar, Ymoc4j.jar, Yutil.jar, Ymsybase.jar`

4.6.4 Using an InterSystems Caché Database

When using an InterSystems Caché database, follow the database connection steps in [Design Time: Using the Command-Line Utility](#) and use the following information:

URL: `jdbc:Cache://machinename_running_Cache_DB_Server:1972/Cache_Namespace`

Driver Class: `com.intersys.jdbc.CacheDriver`

Driver Jar: `C:\CacheSys\Dev\Java\Lib\CacheDB.jar`

The default login is `_SYSTEM/sys`.

4.6.5 Using a MySQL 4 Database

When using a MySQL 4 database, follow the database connection steps in [Design Time: Using the Command-Line Utility](#) and use the following information:

URL: `jdbc:mysql://hostname:3306/dbname`

Driver Class: `com.mysql.jdbc.Driver`

Driver Jar: `mysql-connector-java-3.1.10-bin.jar`

4.6.6 Summary of Third-Party and Oracle Lite Database Connection Information

[Table 4–19](#) summarizes the preceding connection information for common third-party databases and for Oracle Olite. Also see [Table 4–17, "Database Platform Names"](#) for `PlatformClassName` information.

Table 4–19 Information for Connecting to Third-Party Databases and Oracle Olite Database

Database	URL	Driver Class	Driver Jar
Microsoft SQL Server	MS JDBC driver: <code>jdbc:microsoft:sqlserver://localhost\NAME:1433;SelectMethod=cursor;databasename=???</code> DataDirect driver: <code>jdbc:oracle:sqlserver://localhost</code>	MS JDBC driver: <code>com.microsoft.jdbc.sqlserver.SQLServerDriver</code> DataDirect driver: <code>com.oracle.ias.jdbc.sqlserver.SQLServerDriver</code>	MS JDBC driver: <code>.\SQLServer2000\msbase.jar, msutil.jar, mssqlserver.jar</code> DataDirect driver: <code>.\DataDirect\Ymbase.jar, YMoc4j.jar, YMutil.jar, YMsqlserver.jar</code>
IBM DB2	DataDirect driver: <code>jdbc:db2:localhost:NAME</code> JT400 driver (AS400 DB2): <code>jdbc:as400://hostname;translate binary=true</code> Example: <code>jdbc:as400://localhost;translate binary=true</code>	DataDirect driver: <code>COM.ibm.db2.jdbc.net.DB2Driver</code> JT400 driver (AS400 DB2): <code>com.ibm.as400.access.AS400JDBCDriver</code>	DataDirect driver (v8.1): <code>.\IBM-net\db2java_81.zip, db2jcc_81.jar</code> JT400 driver (AS400 DB2): <code>jt400.jar</code>
Sybase	jconn driver: <code>jdbc:sybase:Tds:localhost:5001/NAME</code> DataDirect driver: <code>jdbc:oracle:sybase://localhost:5001</code>	jconn driver: <code>com.sybase.jdbc2.jdbc.SybDriver</code> DataDirect driver: <code>com.oracle.ias.jdbc.sybase.SybaseDriver</code>	jconn driver: <code>.\Sybase-jconn\jconn2.jar</code> DataDirect driver: <code>.\DataDirect\Ymbase.jar, YMoc4j.jar, YMutil.jar, YMsybase.jar</code>
InterSystems Caché	<code>jdbc:Cache://machinename_running_Cache_DB_Server:1972/Cache_Namespace</code> where 1972 is the default port Example: <code>jdbc:Cache://127.0.0.1:1972/Samples</code>	<code>com.intersys.jdbc.CacheDriver</code>	<code>C:\CacheSys\Dev\Java\Lib\CacheDB.jar</code>
MySQL 4	<code>jdbc:mysql://hostname:3306/dbname</code> Example: <code>jdbc:mysql://localhost:3306/test</code>	<code>com.mysql.jdbc.Driver</code>	<code>mysql-connector-java-3.1.10-bin.jar</code>
Oracle Olite Database	<code>jdbc:polite4@localhost:100:orabpel</code>	<code>oracle.lite.poljdbc.POLJDBCDriver</code>	<code>Oracle_Home\bpel\lib\olite40.jar</code>

4.6.7 Location of JDBC Driver JAR Files and Setting the Class Path

At run time, put the driver JAR files in the application server class path in either of the following ways:

- Edit the class path in the following files:

For the Oracle BPEL Process Manager, go to

Oracle_Home/bpel/system/appserver/oc4j/j2ee/home/config/server.xml

For the Oracle BPEL Process Manager for OracleAS Middle Tier installation, go to

Oracle_Home/j2ee/OC4J_BPEL/config/server.xml

Or

- Drop the JAR files into the following directories:

For the Oracle BPEL Process Manager, go to

Oracle_Home/bpel/system/appserver/oc4j/j2ee/home/applib

For the Oracle BPEL Process Manager for OracleAS Middle Tier installation, go to

Oracle_Home/j2ee/OC4J_BPEL/applib

4.7 Stored Procedure and Function Support

This section describes how the database adapter supports the use of stored procedures and functions for Oracle databases only. Stored procedures and functions for non-Oracle databases are not supported out-of-the-box.

This section contains the following topics:

- [Design Time: Using the Adapter Configuration Wizard](#)
- [Design Time: Using the Command-Line Utility](#)
- [Design Time: WSDL and XSD Generation](#)
- [Run Time: Before Stored Procedure Invocation](#)
- [Run Time: After Stored Procedure Invocation](#)
- [Advanced Topics](#)

4.7.1 Design Time: Using the Adapter Configuration Wizard

The Adapter Configuration Wizard – Stored Procedures is used to generate an adapter service WSDL and the necessary XSD. The adapter service WSDL encapsulates the underlying stored procedure or function as a Web service with a WSIF JCA binding. The XSD describes the procedure or function, including all the parameters and their types. This XSD provides the definition used to create instance XML that is submitted to the database adapter at run time.

This section comprises the following:

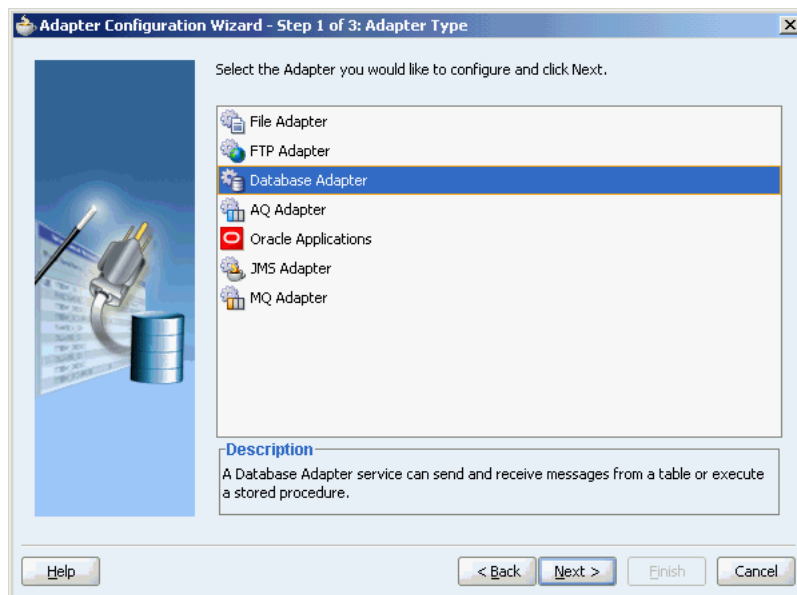
- [Using Top-Level Standalone APIs](#)
- [Using Packaged APIs and Overloading](#)

4.7.1.1 Using Top-Level Standalone APIs

This section describes how to use the wizard with APIs that are not defined in PL/SQL packages. You use the Adapter Configuration Wizard – Stored Procedures to select a procedure or function and generate the XSD. See [Database Adapter Use Case for Oracle BPEL Process Manager](#) if you are not familiar with how to start the wizard.

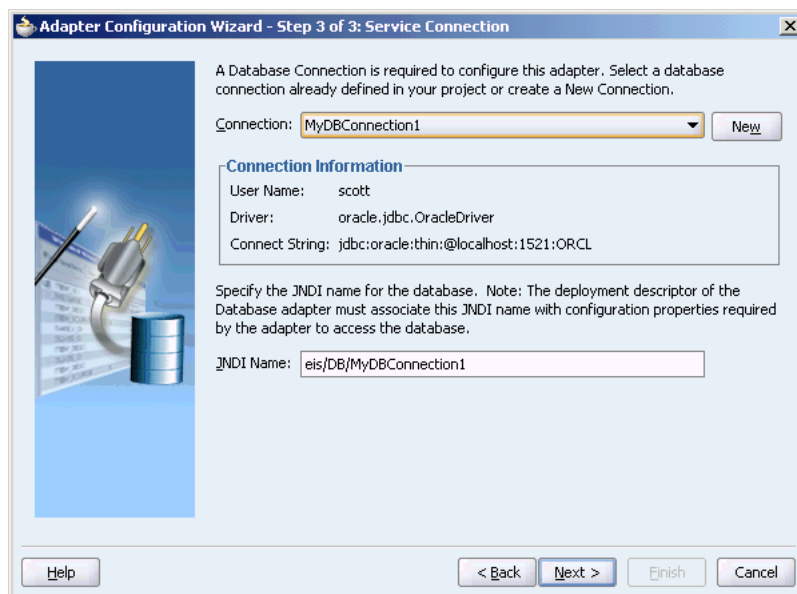
In the wizard, select **Database Adapter**, as shown in [Figure 4-73](#).

Figure 4-73 Selecting the Database Adapter in the Adapter Configuration Wizard



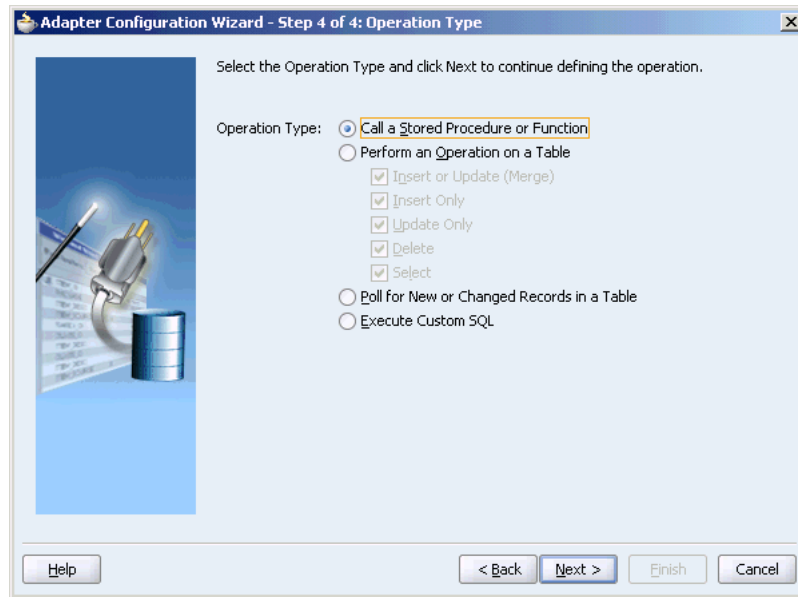
After entering a service name (for example, `ProcedureProc`) and an optional description for the service, you associate a connection with the service, as shown in [Figure 4-74](#). You can select an existing connection from the list or create a new connection.

Figure 4-74 Setting the Database Connection in the Adapter Configuration Wizard



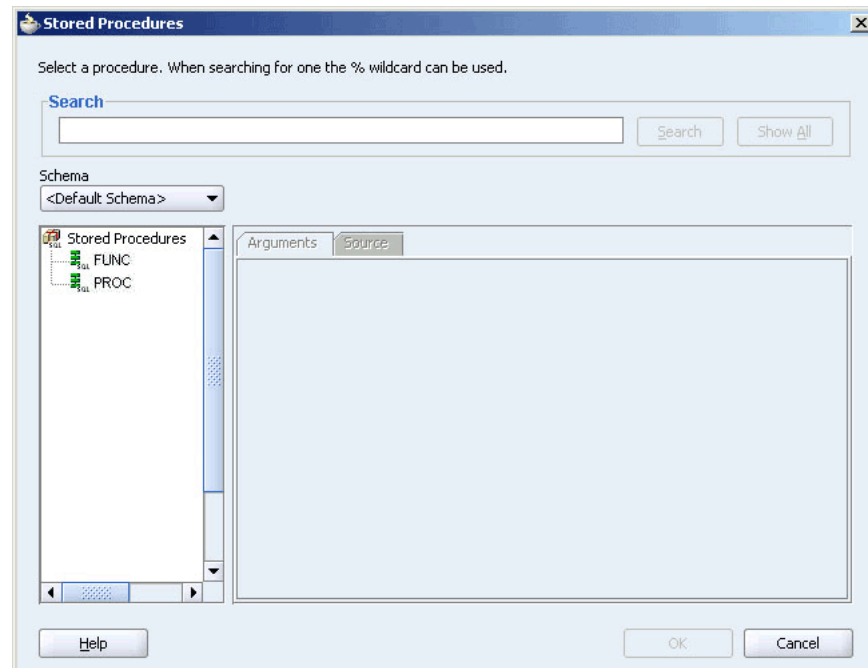
For the **Operation Type**, select **Call a Stored Procedure or Function**, as shown in [Figure 4-75](#).

Figure 4-75 *Calling for a Stored Procedure or Function in the Adapter Configuration Wizard*



Next you select the schema and procedure or function. You can select a schema from the list or select `<Default Schema>`, in which case the schema associated with the connection is used. If you know the procedure name, enter it in the Procedure field. If the procedure is defined inside a package, then you must include the package name, as in `EMPLOYEE.GET_NAME`.

If you do not know the schema and procedure names, click **Browse** to access the Stored Procedures window, shown in [Figure 4-76](#).

Figure 4–76 Searching for a Procedure or Function

Select a schema from the list or select `<Default Schema>`. The available procedures are displayed in the left window. To search for a particular API in a long list of APIs, enter search criteria in the **Search** field. For example, to find all APIs that begin with `XX`, enter `XX%` and click the **Search** button. Clicking the **Show All** button displays all available APIs.

[Figure 4–77](#) shows how you can select the **PROC** procedure and click the **Arguments** tab. The **Arguments** tab displays the parameters of the procedure, including their names, type, mode (IN, IN/OUT or OUT) and the numeric position of the parameter in the definition of the procedure.

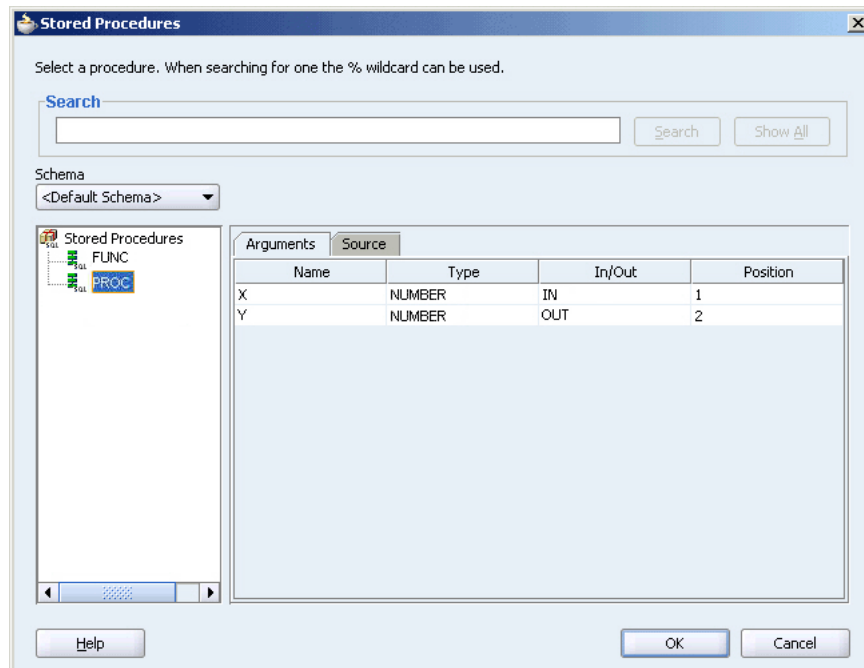
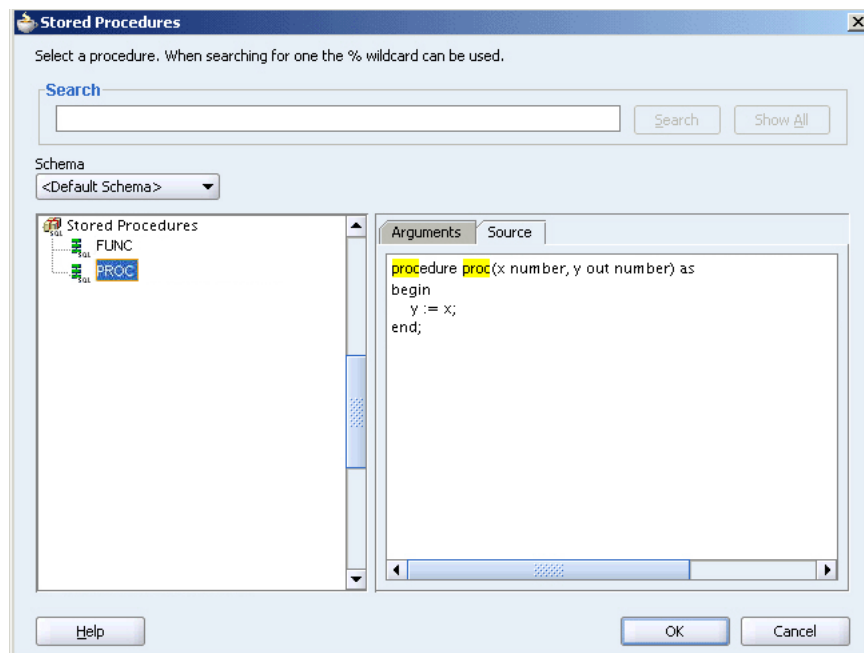
Figure 4–77 Viewing the Arguments of a Selected Procedure

Figure 4–78 shows how the **Source** tab displays the code that implements the procedure. Text that matches the name of the procedure is highlighted.

Figure 4–78 Viewing the Source Code of a Selected Procedure

After you select a procedure or function and click **OK**, information about the API is displayed, as shown in Figure 4–79. Use **Back** or **Browse** to make revisions, or **Next** followed by **Finish** to conclude.

Figure 4–79 Viewing Procedure or Function Details in the Adapter Configuration Wizard

Enter a stored procedure, or a function. The procedure's package name can be included, for example, EMPLOYEE.GET_NAME, where the package name is EMPLOYEE and the procedure is GET_NAME. If the procedure does not belong in a package, enter the procedure's name. You can also browse and search for a procedure. The term 'procedure' is used to mean both stored procedures as well as functions.

Schema: <Default Schema>

Procedure: PROC Browse...

Arguments

Name	Type	In/Out	Position
X	NUMBER	IN	1
Y	NUMBER	OUT	2

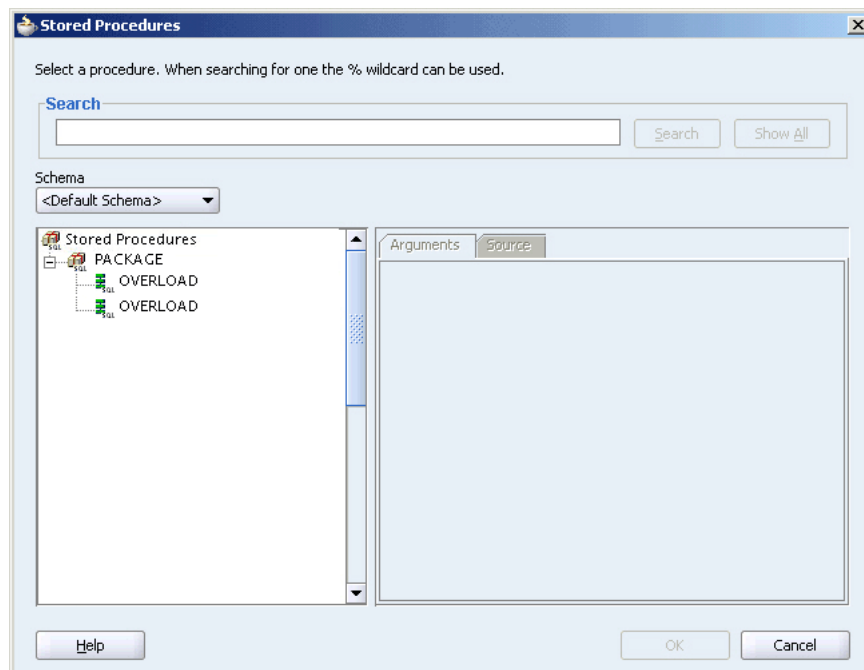
Help < Back Next > Finish Cancel

When you have finished using the Adapter Configuration Wizard, two files are added to the existing project: *servicename.wsdl* (for example, *ProcedureProc.wsdl*) and the generated XSD. The generated XSD file is named *schema_package_procedurename.xsd*. In this case, *SCOTT_PROC.xsd* is the name of the generated XSD file.

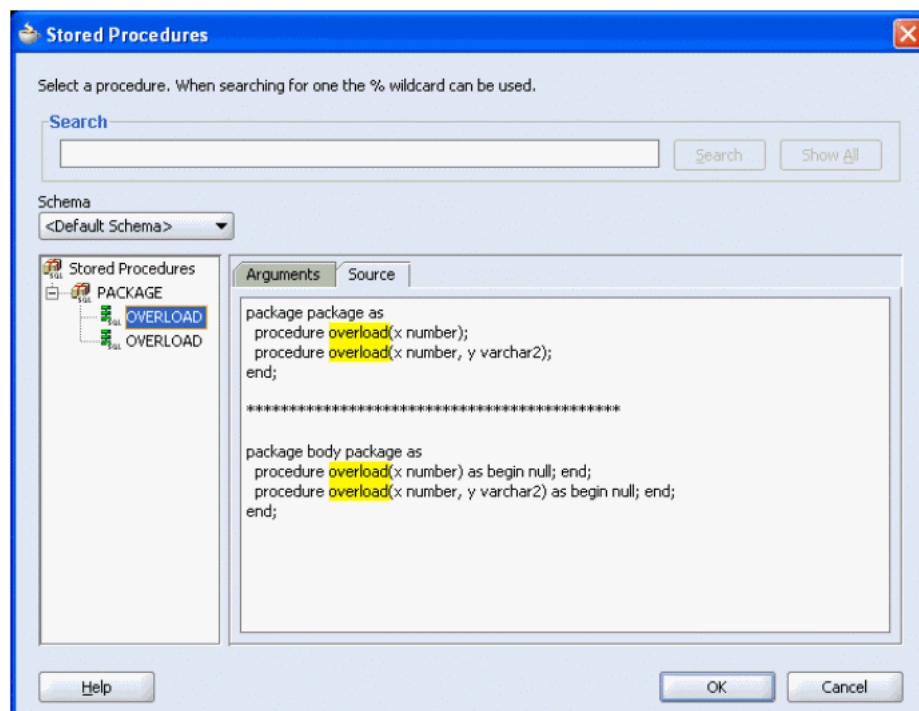
4.7.1.2 Using Packaged APIs and Overloading

Using APIs defined in packages is similar to using standalone APIs. The only difference is that you can expand the package name to see a list of all the APIs defined within the package, as shown in [Figure 4–80](#).

APIs that have the same name but different parameters are called overloaded APIs. As shown in [Figure 4–80](#), the package called **PACKAGE** has two overloaded procedures called **OVERLOAD**.

Figure 4–80 A Package with Two Overloaded Procedures

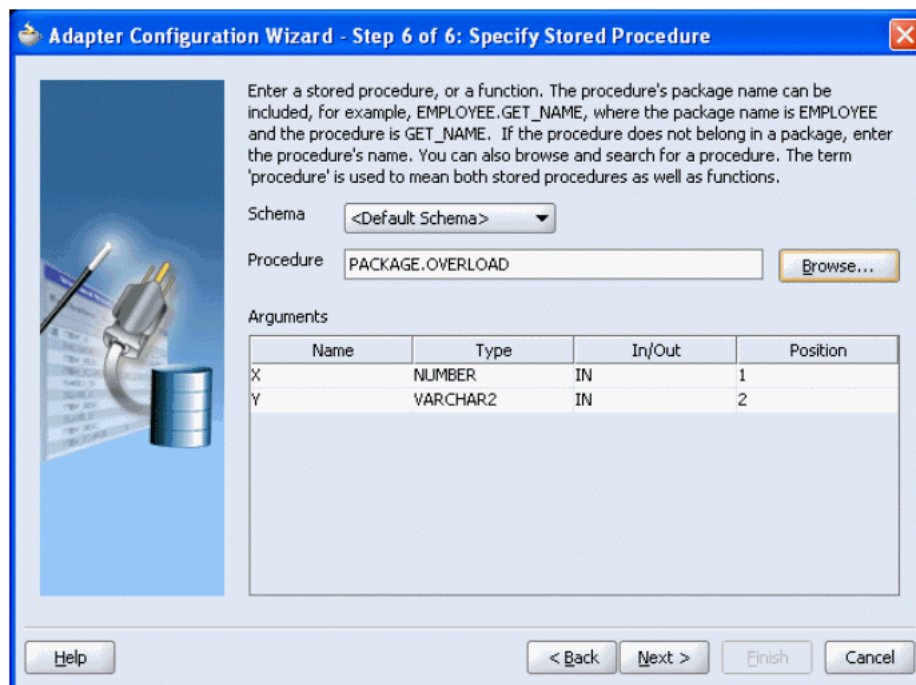
As [Figure 4–81](#) shows, the code for the entire PL/SQL package is displayed, regardless of which API from the package is selected when you view the **Source** tab. Text that matches the name of the procedure is highlighted.

Figure 4–81 Viewing the Source Code of an Overloaded Procedure

After you select a procedure or function and click **OK**, information about the API is displayed, as shown in [Figure 4–82](#). The schema, procedure name, and parameter list

are displayed. Note how the procedure name is qualified with the name of the package (**PACKAGE.OVERLOAD**). Use **Back** or **Browse** to make revisions, or **Next** followed by **Finish** to conclude.

Figure 4–82 Viewing Procedure or Function Details in the Adapter Configuration Wizard



When you have finished using the Adapter Configuration Wizard, two files are added to the existing project: `Overload.wsdl` and `SCOTT_PACKAGE_OVERLOAD_2.xsd`. The `_2` appended after the name of the procedure in the XSD filename differentiates the overloaded APIs.

4.7.2 Design Time: Using the Command-Line Utility

The command-line utility is invoked with a properties file and the filename of a template WSDL that will be used to create the desired service WSDL. This section comprises the following:

- [Common Command-Line Functionality](#)
- [Generated Output](#)
- [Supported Third-Party Database](#)

4.7.2.1 Common Command-Line Functionality

Each of the additional databases shares some common functionality that is provided by the command-line utility. Several properties are shared by all supported databases. The following is a list of properties shared by the supported databases:

ProductName

This is the name of the additional database. ProductName supports IBM DB2: IBM DB2v8.2, and Microsoft SQL Server: SQL Server 2000 or 2005.

DriverClassName

This is the class name of the JDBC driver.

ConnectionString

This is the JDBC connection URL.

Username

This is the database user name.

Password

This is the password associated with the user name.

ProcedureName

This is the name of the stored procedure or the function.

ServiceName

This is the service name for the desired operation.

DatabaseConnection

This is the JNDI name of the connection. For example,
eis/DB/<DatabaseConnection>.

Destination

This is the destination directory for the generated files, for example, C:\Temp.

```
C:\java oracle.tip.adapter.db.sp.artifacts.GenerateArtifacts <properties>
```

Where <properties> is the name of the properties file. The utility will verify that all mandatory properties exist including any database-specific properties, as well.

4.7.2.2 Generated Output

The command-line utility generates three files, an XSD representing the signature of the chosen API, an adapter service WSDL and a project WSDL imported by the service WSDL. The name of the generated WSDL is derived from the value of the `ServiceName` property. The name of the XSD is derived from the `ProcedureName` property and other property values specific to the database. For example, the schema and package name for Oracle database.

The contents of the generated service WSDL are derived using a WSDL template and the values of the required properties. The contents of the XSD are derived from the qualified procedure name and a type mapping tables specific to the database in use. The elements that represent parameters of the stored procedure have the same number and type of attributes as those that are generated using the adapter wizard.

4.7.2.3 Supported Third-Party Database

The command-line utility can be used to generate the required service artifacts for IBM DB2 v8.2, SQL Server 2000 and 2005. The data types and properties that are supported vary for each database. This section comprises the following:

- [Microsoft SQL Server 2000 and 2005](#)
- [IBM DB2 v8.2](#)

4.7.2.3.1 Microsoft SQL Server 2000 and 2005

SQL Server supports functions, in addition to stored procedures. A property is necessary to indicate to the command-line utility that the chosen API is a function as opposed to a procedure. The following table list the additional properties used with Microsoft SQL Server:

Property	Description
IsFunction	If the API is a function, then its value is <code>true</code> or <code>false</code> , by default.
DatabaseName	The name of the database where the API is defined
SchemaName	The name of the schema to which the procedure belongs

The `IsFunction` property is optional. The default value is `false`. If the API is a procedure, then this property need not be specified. If the API is a function or the API is a procedure that returns a value, then this property must be set to `true`, `True`, or `TRUE`. The following is an example of a procedure that returns a value:

```
1> create procedure ... as begin ...;return 1; end
2> go
```

If the value of `IsFunction` is not set, or if it is set to `false` or some other value, then the return value will not be included in the generated XML after the API executes.

The `SchemaName` and `DatabaseName` properties are used to further qualify the stored procedure. For example, `<DatabaseName>.<SchemaName>.<ProcedureName>`. These properties are optional, so they need not be specified.

[Table 4–20](#) lists the data types that are supported only for stored procedures for use with SQL Server database:

Table 4–20 Data Types Supported for Stored Procedures for Use with SQL Server Database

SQL Data Type	XML Schema Type
BIGINT	long
BINARY	base64Binary
IMAGE	
TIMESTAMP	
VARBINARY	
BIT	boolean
CHAR	string
SQL_VARIANT	
SYSNAME	
TEXT	
UNIQUEIDENTIFIER	
VARCHAR	
XML (2005 only)	
DATETIME	dateTime
SMALLDATETIME	
DECIMAL	decimal
MONEY	
NUMERIC	
SMALLMONEY	

Table 4–20 (Cont.) Data Types Supported for Stored Procedures for Use with SQL Server Database

SQL Data Type	XML Schema Type
FLOAT	float
REAL	
INT	int
SMALLINT	short
TINYINT	unsignedByte

Besides, the data types mentioned in the preceding table, alias data types are also supported for stored procedure. The alias data types are created by using the `sp_addtype` database engine stored procedure or the `CREATE TYPE` Transact-SQL statement (only for SQL Server 2005.) Note that the use of the Transact-SQL statement is the preferred method for creating alias data types. The use of `sp_addtype` is being deprecated.

If the data type of a parameter in a stored procedure is defined using an alias data type, then the underlying base SQL data type will be determined, and this data type and its mappings will be used for the parameter in the generated XSD. Consider the following example:

```
1>create type myint from int
2>go
3>create procedure aliastype @x myint as ...
4>go
```

The type of the parameter in the procedure, in the preceding example is the alias type, `myint`. The underlying database SQL data type of `myint` is `INT`, as shown in the following example, whose type mappings will be used for that parameter in the stored procedure.

```
<element name="x" type="int" ... db:type="INT" ... />
```

Structured data types (user-defined) were introduced in SQL Server 2005. The command-line utility, however, does not support them. Therefore, structured data types may not be used as the type of a parameter in a stored procedure.

4.7.2.3.2 IBM DB2 v8.2

IBM DB2 utilizes an additional property to further qualify the stored procedure. Note that only SQL stored procedures are supported. External procedures and user-defined functions are not supported.

In IBM DB2 the `SchemaName` property is mandatory. `SchemaName` is the name of the schema to which the procedure belongs. It is used by the command-line utility to qualify the stored procedure so that it can verify that the procedure exists. [Table 4–21](#) lists the data types supported only for stored procedures for use with DB2 database:

Table 4–21 Data Types Supported for Stored Procedures by IBM DB2 Database

SQL Data Type	XML Schema Type
BIGINT	long
BLOB	base64Binary
CHAR FOR BIT DATA	
VARCHAR FOR BIT DATA	

Table 4–21 (Cont.) Data Types Supported for Stored Procedures by IBM DB2 Database

SQL Data Type	XML Schema Type
CHARACTER	string
CLOB	
VARCHAR	
DATE	dateTime
TIMESTAMP	
DECIMAL	decimal
DOUBLE	double
INTEGER	int
REAL	float
SMALLINT	short

Note that the names of other data types are also supported implicitly. For example, NUMERIC is equivalent to DECIMAL (as is DEC and NUM as well.)

Distinct data types are also supported only for stored procedures. These are similar to alias data types in SQL Server. They are created using the `CREATE DISTINCT TYPE` statement, as shown in the following example:

```
db2 => create distinct type myint as integer with comparisons
db2 => create procedure distincttype(in x myint, ...) begin ... end
```

The underlying database SQL data type of `myint` is `INTEGER`, whose type mappings will be used for that parameter in the stored procedure.

```
<element name="X" type="int" ... db:type="INTEGER" ... />
```

IBM DB2 supports structured data types (user-defined). However, there is no support for these types in the JDBC drivers. Consequently, a structured data type may not be used as the type of a parameter in a stored procedure. IBM DB2 also supports user-defined functions. The adapter, however, does not support these.

4.7.3 Design Time: WSDL and XSD Generation

The Adapter Configuration Wizard – Stored Procedures is capable of creating a WSDL and a valid XSD that describes the signature of a stored procedure or function. The following sections describe the relevant structure and content of both the WSDL and the XSD, and their relationship with each other.

This section comprises the following:

- [The WSDL–XSD Relationship](#)
- [Supported Primitive Data Types](#)
- [Generated XSD Attributes](#)
- [User-Defined Types](#)
- [Complex User-Defined Types](#)
- [Object Type Inheritance](#)
- [Object References](#)
- [Referencing Types in Other Schemas](#)

4.7.3.1 The WSDL–XSD Relationship

In the paragraphs that follow, the operation name, `ProcedureProc`, and procedure name, `PROC`, are taken from an example cited previously (see [Figure 4–79](#) on page 4-105). The generated WSDL imports the XSD.

```
<types>
  <schema xmlns="http://www.w3.org/2001/XMLSchema">
    <import
      namespace="http://xmlns.oracle.com/pcbpel/adapter/db/SCOTT/PROC/"
      schemaLocation="SCOTT_PROC.xsd"/>
    </schema>
  </types>
```

The namespace is derived from the schema, package, and procedure name, and appears as the `targetNamespace` in the generated XSD.

A root element called `InputParameters` is created in the XSD for specifying elements that correspond to the `IN` and `IN/OUT` parameters of the stored procedure. Another root element called `OutputParameters` is also created in the XSD for specifying elements only if there are any `IN/OUT` or `OUT` parameters. Note that `IN/OUT` parameters appear in both root elements.

These root elements are represented in the XSD as an unnamed `complexType` definition whose sequence includes one element for each parameter. If there are no `IN` or `IN/OUT` parameters, the `InputParameters` root element is still created; however, the `complexType` is empty. A comment in the XSD indicates that there are no such parameters. An example of one of these root elements follows.

```
<element name="InputParameters"
  <complexType>
    <sequence>
      <element ...>
        ...
    </sequence>
  </complexType>
</element>
```

The WSDL defines message types whose parts are defined in terms of these two root elements.

```
<message name="args_in_msg"
  <part name="InputParameters" element="db:InputParameters"/>
</message>
<message name="args_out_msg"
  <part name="OutputParameters" element="db:OutputParameters"/>
</message>
```

The `db` namespace is the same as the `targetNamespace` of the generated XSD. Note that the `args_in_msg` message type always appears in the WSDL while `args_out_msg` is included only if the `OutputParameters` root element is generated in the XSD.

An operation is defined in the WSDL whose name is the same as the adapter service and whose input and output messages are defined in terms of these two message types.

```
<portType name="ProcedureProc_ptt">
  <operation name="ProcedureProc">
    <input message="tns:args_in_msg"/>
    <output message="tns:args_out_msg"/>
  </operation>
</portType>
```

The input message always appears while the output message depends on the existence of the `OutputParameters` root element in the XSD. The `tns` namespace is derived from the operation name and is defined in the WSDL as

```
xmlns:tns="http://xmlns.oracle.com/pcbpel/adapter/db/ProcedureProc/"
```

The root elements in the XSD define the structure of the parts used in the messages that are passed into and sent out of the Web service encapsulated by the WSDL.

The input message in the WSDL corresponds to the `InputParameters` root element from the XSD. The instance XML supplies values for the `IN` and `IN/OUT` parameters of the stored procedure. The output message corresponds to the `OutputParameters` root element. This is the XML that gets generated after the stored procedure has executed. It holds the values of any `IN/OUT` and `OUT` parameters.

4.7.3.2 Supported Primitive Data Types

Many primitive data types have well-defined mappings and therefore are supported by both the design-time and run-time components. In addition, you can use user-defined types such as `VARRAY`, nested tables, and `OBJECT`. [Table 4–22](#) lists the primitive data types supported by the database adapter for stored procedures.

Table 4–22 *Primitive Data types Supported by the Database Adapter for Stored Procedures*

SQL or PL/SQL Type	XML Schema Type
BINARY_DOUBLE	double
DOUBLE PRECISION	
BINARY_FLOAT	float
FLOAT	
REAL	
BINARY_INTEGER	int
INTEGER	
PLS_INTEGER	
SMALLINT	
BLOB	base64Binary
LONG RAW	
RAW	
CHAR	string
CLOB	
LONG	
VARCHAR2	
DATE	dateTime
TIMESTAMP	
DECIMAL	decimal
NUMBER	

4.7.3.3 Generated XSD Attributes

[Table 4–23](#) lists the attributes used in the generated XSDs. Attributes prefixed with `db:` are specific to the database adapter.

Table 4–23 *Generated XSD Attributes*

Attribute	Example	Purpose
name	name="param"	Name of an element
type	type="string"	XML schema type
db:type	db:type="VARCHAR2"	SQL or PL/SQL type
db:index	db:index="1"	Position of a parameter
db:default	db:default="true"	Has a default clause
minOccurs	minOccurs="0"	Minimum occurrences
maxOccurs	maxOccurs="1"	Maximum occurrences
nillable	nillable="true"	Permits null values

The `db` namespace is used to distinguish attributes used during run time from standard XML schema attributes. The `db:type` attribute is used to indicate what the database type is so that a suitable JDBC type mapping can be obtained at run time. The `db:index` attribute is used as an optimization by both the design-time and run-time components to ensure that the parameters are arranged in the proper order. Parameter indices begin at 1 for procedures and 0 for functions. The return value of a function is represented as an `OutputParameter` element whose name is the name of the function and whose `db:index` is 0. The `db:default` attribute is used to indicate whether or not a parameter has a default clause.

The `minOccurs` value is set to 0 to allow for an `IN` parameter to be removed from the XML. This is useful when a parameter has a default clause defining a value for the parameter (for example, `X IN INTEGER DEFAULT 0`). At run time, if no element is specified for the parameter in the XML, the parameter is omitted from the invocation of the stored procedure, thus allowing the default value to be used. Each parameter can appear at most once in the invocation of a stored procedure or function. Therefore, `maxOccurs`, whose default value is always 1, is always omitted from elements representing parameters.

The `nillable` attribute is always set to `true` to allow the corresponding element in the instance XML to have a null value (for example, `<X/>` or `<X></X>`). In some cases, however, to pass schema validation, an element such as this, which does have a null value, must state this explicitly (for example, `<X xsi:nil="true"/>`). The namespace, `xsi`, used for the `nillable` attribute, must be declared explicitly in the instance XML (for example, `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"`).

4.7.3.4 User-Defined Types

The wizard can also generate valid definitions for user-defined types such as collections (VARRAY and nested tables) and `OBJECT`. These are created as `complexType` definitions in the XSD.

For `VARRAY`, the `complexType` definition defines a single element in its sequence, called `name_ITEM`, where `name` is the name of the `VARRAY` element. All array elements in the XML are so named. Given the following `VARRAY` type definition,

```
SQL> CREATE TYPE FOO AS VARRAY (5) OF VARCHAR2 (10);
```

and a `VARRAY` element, `X`, whose type is `FOO`, the following `complexType` is generated:

```
<complexType name="FOO">
```

```

<sequence>
  <element name="X_ITEM" db:type="VARCHAR2" minOccurs="0" maxOccurs="5"
nillable="true"/>
  <simpleType>
    <restriction base="string">
      <maxLength value="10"/>
    </restriction>
  </simpleType>
</sequence>
</complexType>

```

The `minOccurs` value is 0 to allow for an empty collection. The `maxOccurs` value is set to the maximum number of items that the collection can hold. Note that the `db:index` attribute is not used. Having `nillable` set to `true` allows individual items in the VARRAY to be null.

Note the use of the restriction specified on the element of the VARRAY, `FOO`. This is used on types such as `CHAR` and `VARCHAR2`, whose length is known from the declaration of the VARRAY (or nested table). It specifies the type and maximum length of the element. An element value that exceeds the specified length causes the instance XML to fail during schema validation.

The attribute values of a parameter declared to be of type `FOO` look as follows in the generated XSD:

```

<element name="X" type="db:FOO" db:type="Array" db:index="1" minOccurs="0"
nillable="true"/>

```

The `type` and `db:type` values indicate that the parameter is represented as an array defined by the `complexType` called `FOO` in the XSD. The value for `db:index` is whatever the position of that parameter is in the stored procedure.

A nested table is treated almost identically to a VARRAY. The following nested table type definition,

```
SQL> CREATE TYPE FOO AS TABLE OF VARCHAR2 (10);
```

is also generated as a `complexType` with a single element in its sequence, called `name_ITEM`. The element has the same attributes as in the VARRAY example, except that the `maxOccurs` value is unbounded because nested tables can be of arbitrary size.

```

<complexType name="FOO">
  <sequence>
    <element name="X_ITEM" ... maxOccurs="unbounded" nillable="true">
      ...
    </element>
  </sequence>
</complexType>

```

An identical restriction is generated for the `X_ITEM` element in the VARRAY. The attributes of a parameter, `X`, declared to be of this type, are the same as in the VARRAY example.

Note that collections (Varray and nested table) are not supported if they are defined inside of a PL/SQL package specification. For example:

```

SQL> create package pkg as
  >   type vary is varray(10) of number;
  >   type ntbl is table of varchar2(100);
  >   procedure test(v in vary, n in ntbl);
  > end;
  > /

```

If a user selects the *test* procedure in the DBAdapter wizard for stored procedures, an error will occur stating that the types are not supported. However, if the *vary* and *ntbl* type definitions were defined at the root-level, outside of the package, then choosing the *test* procedure will work without issue. The supported way to use collection types (*Varray* and nested table) is shown in the following example:

```
SQL> create type vary as varray(10) of number;
SQL> create type ntbl as table of varchar2(10);
SQL> create package pkg as
  >   procedure test(v in vary, n in ntbl);
  > end;
  /
```

An *OBJECT* definition is also generated as a *complexType*. Its sequence holds one element for each attribute in the *OBJECT*.

The following *OBJECT*,

```
SQL> CREATE TYPE FOO AS OBJECT (X VARCHAR2 (10), Y NUMBER);
```

is represented as a *complexType* called *FOO* with two sequence elements.

```
<complexType name="FOO">
  <sequence>
    <element name="X" db:type="VARCHAR2" minOccurs="0" nillable="true"/>
      <simpleType>
        <restriction base="string">
          <maxLength value="10"/>
        </restriction>
      </simpleType>
    <element name="Y" type="decimal" db:type="NUMBER" minOccurs="0"
nillable="true"/>
  </sequence>
</complexType>
```

The *minOccurs* value is 0 to allow for the element to be removed from the XML. This causes the value of the corresponding attribute in the *OBJECT* to be set to null at run time. The *nillable* value is *true* to allow empty elements to appear in the XML, annotated with the *xsi:nil* attribute, to indicate that the value of the element is null. Again, the *db:index* attribute is not used.

Note the use of a restriction on the *VARCHAR2* attribute. The length is known from the declaration of the attribute in the *OBJECT*.

4.7.3.5 Complex User-Defined Types

User-defined types can be defined in arbitrarily complex ways. An *OBJECT* can contain attributes whose types are defined as any of the aforementioned user-defined types. This means that the type of an attribute in an *OBJECT* can be another *OBJECT*, *VARRAY* or a nested table, and so on. The base type of a *VARRAY* or a nested table can also be an *OBJECT*. Allowing the base type of a collection to be another collection supports multidimensional collections.

4.7.3.6 Object Type Inheritance

The wizard is capable of generating a valid XSD for parameters whose types are defined using *OBJECT*-type inheritance. Given the following type hierarchy,

```
SQL> CREATE TYPE A AS OBJECT (A1 NUMBER, A2 VARCHAR2 (10)) NOT FINAL;
SQL> CREATE TYPE B UNDER A (B1 VARCHAR2 (10));
```

and a procedure containing a parameter, X, whose type is B,

```
SQL> CREATE PROCEDURE P (X IN B) AS BEGIN ... END;
```

the wizard generates an `InputParameters` element for parameter X as

```
<element name="X" type="db:B" db:index="1" db:type="Struct" minOccurs="0"
nillable="true"/>
```

where the definition of OBJECT type B in the XSD is generated as the following `complexType`.

```
<complexType name="B">
  <sequence>
    <element name="A1" type="decimal" db:type="NUMBER" minOccurs="0"
nillable="true"/>
    <element name="A2" db:type="VARCHAR2" minOccurs="0" nillable="true">
      ...
    </element>
    <element name="B1" db:type="VARCHAR2" minOccurs="0" nillable="true">
      ...
    </element>
  </sequence>
</complexType>
```

Restrictions on the maximum length of attributes A2 and B1 are added appropriately. Notice how the OBJECT type hierarchy is flattened into a single sequence of elements that corresponds to all of the attributes in the entire hierarchy.

4.7.3.7 Object References

The wizard can also generate a valid XSD for parameters that are references to OBJECT types (for example, object references), or are user-defined types that contain an object reference somewhere in their definition. In this example,

```
SQL> CREATE TYPE FOO AS OBJECT (...);
SQL> CREATE TYPE BAR AS OBJECT (F REF FOO, ...);
SQL> CREATE PROCEDURE PROC (X OUT BAR, Y OUT REF FOO) AS BEGIN ... END;
```

the wizard generates `complexType` definitions for FOO and BAR as already indicated, except that for BAR, the element for the attribute, F, is generated as

```
<element name="F" type="db:FOO" db:type="Ref" minOccurs="0" nillable="true"/>
```

where together, the type and `db:type` attribute values indicate that F is a reference to the OBJECT type FOO.

For a procedure PROC, the following elements are generated in the `OutputParameters` root element of the XSD:

```
<element name="X" type="db:BAR" db:index="1" db:type="Struct" minOccurs="0"
nillable="true"/>
<element name="Y" type="db:FOO" db:index="2" db:type="Ref" minOccurs="0"
nillable="true"/>
```

For Y, note the value of the `db:type` attribute, Ref. Together with the type attribute, the element definition indicates that Y is a reference to FOO.

Note that there is a restriction on the use of object references that limits their parameter mode to OUT only. Passing an IN or IN/OUT parameter into an API that is either

directly a REF or, if the type of the parameter is user-defined, contains a REF somewhere in the definition of that type, is not permitted.

4.7.3.8 Referencing Types in Other Schemas

You can refer to types defined in other schemas, provided that the necessary privileges to access them have been granted. For example, suppose type OBJ was declared in SCHEMA1:

```
SQL> CREATE TYPE OBJ AS OBJECT (...);
```

The type of a parameter in a stored procedure declared in SCHEMA2 can be type OBJ from SCHEMA1:

```
CREATE PROCEDURE PROC (O IN SCHEMA1.OBJ) AS BEGIN ... END;
```

This is possible only if SCHEMA1 granted permission to SCHEMA2 to access type OBJ:

```
SQL> GRANT EXECUTE ON OBJ TO SCHEMA2;
```

If the required privileges are not granted, an error occurs when trying to create procedure PROC in SCHEMA2:

```
PLS-00201: identifier "SCHEMA1.OBJ" must be declared
```

Because the privileges have not been granted, type OBJ from SCHEMA1 is not visible to SCHEMA2; therefore, SCHEMA2 cannot refer to it in the declaration of parameter O.

4.7.4 Run Time: Before Stored Procedure Invocation

This section discusses important considerations of stored procedure support and a brief overview of some important details regarding what happens prior to the invocation of a stored procedure or function.

This section comprises the following:

- [Value Binding](#)
- [Data Type Conversions](#)

4.7.4.1 Value Binding

Consider the extraction of values from the XML and how the run time works given those values. The possible cases for data in the XML corresponding to the value of a parameter whose type is one of the supported primitive data types value are as follows:

1. The value of an element is specified (for example, `<X>100</X>`, here `X=100`.)
2. The value of an element is not specified (for example, `<X/>`, here `X=null`.)
3. The value is explicitly specified as null (for example, `<X xsi:nil="true"/>`, here `X=null`.)
4. The element is not specified in the XML at all (for example, `X = <default value>`).

In the first case, the value is taken from the XML as-is and is converted to the appropriate object according to its type. That object is then bound to its corresponding parameter during preparation of the stored procedure invocation.

In the second and third cases, the actual value extracted from the XML is null. The type converter accepts null and returns it without any conversion. The null value is

bound to its corresponding parameter regardless of its type. Essentially, this is the same as passing null for parameter X.

The fourth case has two possibilities. The parameter either has a default clause or it does not. If the parameter has a default clause, then the parameter is completely excluded from the invocation of the stored procedure. This allows the default value to be used for the parameter. On the other hand, if the parameter does not have a default clause, then the parameter is included in the invocation of the procedure. However, for functions, elements for all parameters must be specified. If an element in the instance XML is missing, then the function will be invoked with fewer arguments than the expected. IBM DB2 does not provide a mechanism for specifying a default value for a parameter in a stored procedure. Therefore, an element must exist in the instance XML for every parameter.

A null value is bound to the parameter by default:

```
SQL> CREATE PROCEDURE PROC (X IN INTEGER DEFAULT 0) AS BEGIN ... END;
```

Here, no value is bound to the parameter. In fact, the parameter is completely excluded from the invocation of the stored procedure. This allows the value of 0 to default for parameter X.

To summarize, the following PL/SQL is executed in each of the four cases:

1. "BEGIN PROC (X=>?); END;" - X = 100
2. "BEGIN PROC (X=>?); END;" - X = null
3. "BEGIN PROC (X=>?); END;" - X = null
4. There are two possibilities:
 - a. "BEGIN PROC (); END;" - X = 0 (X has a default clause)
 - b. "BEGIN PROC (X=>?); END;" - X = null (X does not have a default clause)

With the exception of default clause handling, these general semantics also apply to item values of a collection or attribute values of an OBJECT whose types are one of the supported primitive data types. The semantics of <X/> when the type is user-defined are, however, quite different.

For a collection, whether it is a VARRAY or a nested table, the following behavior can be expected given a type definition such as

```
SQL> CREATE TYPE ARRAY AS VARRAY (5) OF VARCHAR2 (10);
```

and XML for a parameter, X, which has type ARRAY, that appears as follows:

```
<X>
  <X_ITEM xsi:nil="true"/>
  <X_ITEM>Hello</X_ITEM>
  <X_ITEM xsi:nil="true"/>
  <X_ITEM>World</X_ITEM>
</X>
```

The first and third elements of the VARRAY are set to null. The second and fourth are assigned their respective values. No fifth element is specified in the XML; therefore, the VARRAY instance has only four elements.

Assume an OBJECT definition such as

```
SQL> CREATE TYPE OBJ AS OBJECT (A INTEGER, B INTEGER, C INTEGER);
```

and XML for a parameter, X, which has type OBJ, that appears as

```
<X>
  <A>100</A>
  <C xsi:nil="true"/>
</X>
```

The value 100 is assigned to attribute A and null is assigned to attributes B and C. Because there is no element in the instance XML for attribute B, a null value is assigned.

The second case, <X/>, behaves differently if the type of X is user-defined. Rather than assigning null to X, an initialized instance of the user-defined type is created and bound instead.

In the preceding VARRAY example, if <X/> or <X></X> is specified, then the value bound to X is an empty instance of the VARRAY. In PL/SQL, this is equivalent to calling the type constructor and assigning the value to X. For example,

```
X := ARRAY();
```

Similarly, in the preceding OBJECT example, an initialized instance of OBJ, whose attribute values have all been null assigned, is bound to X. Like the VARRAY case, this is equivalent to calling the type constructor. For example,

```
X := OBJ(NULL, NULL, NULL);
```

To specifically assign a null value to X when the type of X is user-defined, add the `xsi:nil` attribute to the element in the XML, as in

```
<X xsi:nil="true"/>
```

4.7.4.2 Data Type Conversions

This section describes the conversion of data types such as CLOB, DATE, TIMESTAMP and binary data types including RAW, LONG RAW and BLOB, as well as similar data types supported by third-party databases.

For CLOB parameters, a temporary CLOB is first created. The data extracted from the XML is then written to it before binding the CLOB to its corresponding parameter. The temporary CLOB is freed when the interaction completes. For other character types, such as CHAR and VARCHAR2, the data is simply extracted and bound as necessary. Note that it is possible to bind an XML document to a CLOB (or VARCHAR2 if it is large enough). However, appropriate substitutions for <, >, and so on, must first be made (for example, < for < and > for >).

Note that the XML schema type, `dateTime`, represents both DATE and TIMESTAMP. This means that the XML values for both data types must adhere to the XML schema representation for `dateTime`. Therefore, a simple DATE string, 01-JAN-05, is invalid. XML schema defines `dateTime` as YYYY-MM-DDTHH:mm:ss. Therefore, the correct DATE value is 2005-01-01T00:00:00.

Data for binary data types must be represented in a human readable manner. The chosen XML schema representation for binary data is `base64Binary`. The type converter uses the `javax.xml.internet.MimeUtility` encode and decode APIs to process binary data. The encode API must be used to encode all binary data into `base64Binary` form so that it can be used in an XML file. The type converter uses the decode API to decode the XML data into a byte array. This is then bound either directly, as is the case with RAW and LONG RAW parameters, or is used to create a temporary BLOB, which is then bound to its associated BLOB parameter. The temporary BLOB is freed when the interaction completes.

Conversions for the remaining data types are straightforward and require no additional information.

4.7.5 Run Time: After Stored Procedure Invocation

After the procedure (or function) executes, the values for any IN/OUT and OUT parameters are retrieved. These correspond to the values of the elements in the `OutputParameters` root element in the generated XSD.

This section comprises the following:

- [Data Type Conversions](#)
- [Null Values](#)
- [Function Return Values](#)

4.7.5.1 Data Type Conversions

Conversions of data retrieved are straightforward. However, BLOB, CLOB (and other character data), RAW, LONG RAW and BLOB conversions, as well as conversions for similar data types supported by third-party databases, require special attention.

When a CLOB is retrieved, the entire contents of that CLOB are written to the corresponding element in the generated XML. Standard DOM APIs are used to construct the XML. This means that character data, as for types like CLOB, CHAR, and VARCHAR2, is massaged as needed to make any required substitutions so that the value is valid and can be placed in the XML for subsequent processing. Therefore, substitutions for `<and>`, for example, in an XML document stored in a CLOB are made so that the value placed in the element within the generated XML for the associated parameter is valid.

Raw data, such as for RAW and LONG RAW types, is retrieved as a byte array. For BLOBs, the BLOB is first retrieved, and then its contents are obtained, also as a byte array. The byte array is then encoded using the `javax.mail.internet.MimeUtility.encodeAPI` into base64Binary form. The encoded value is then placed in its entirety in the XML for the corresponding element. The `MimeUtility.decodeAPI` must be used to decode this value back into a byte array.

Conversions for the remaining data types are straightforward and require no additional information.

4.7.5.2 Null Values

Elements whose values are null appear as empty elements in the generated XML and are annotated with the `xsi:nil` attribute. This means that the `xsi` namespace is declared in the XML that is generated. Generated XML for a procedure PROC, which has a single OUT parameter, X, whose value is null, looks as follows:

```
<db:OutputParameters ...
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <X xsi:nil="true"/>
</db:OutputParameters>
```

The `db` namespace is also declared (that is, `xmlns:db="..."`). Note that XML elements for parameters of any type (including user-defined types) appear this way if their value is null.

4.7.5.3 Function Return Values

The return value of a function is treated as an OUT parameter at position 0 whose name is the name of the function itself. For example,

```
CREATE FUNCTION FACTORIAL (X IN INTEGER) RETURN INTEGER AS
BEGIN
    IF (X <= 0) THEN RETURN 1;
    ELSE RETURN FACTORIAL (X - 1);
    END IF;
END;
```

An invocation of this function with a value of 5, for example, results in a value of 120 and appears as <FACTORIAL>120</FACTORIAL> in the XML generated in `OutputParameters`.

4.7.6 Runtime: Common Third-Party Database Functionality

Both SQL Server and DB2 share the same functionality for handling `ResultSet`s. The following is a SQL Server example of an API that returns a `ResultSet`:

```
1> create procedure foo ... as select ... from ...;
2> go
```

A `RowSet` defined in the generated XSD represents a `ResultSet`. A `RowSet` consists of zero or more rows, each having one or more columns. A row corresponds with a row returned by the query. A column corresponds with a column item in the query. The generated XML for the API shown in the preceding example after it executes is shown in the following example:

```
<RowSet>
  <Row>
    <Column name="<column name>" sqltype="<sql datatype>">value</Column>
    ...
  </Row>
  ...
</RowSet>
...
```

The column *name* attribute stores the name of the column appearing in the query while the *sqltype* attribute stores the SQL datatype of that column, for example `INT`. The *value* will be whatever the value is for that column.

Note that it is possible for an API to return multiple `ResultSet`s. In such cases, there will be one `RowSet` for each `ResultSet` in the generated XML. All `RowSet`s will always appear first in the generated XML.

4.7.7 Advanced Topics

This section discusses scenarios for types that are not supported directly using the stored procedure functionality that the database adapter provides. The following sections describe workarounds that address the need to use these data types:

- [Support for REF CURSOR](#)
- [Support for PL/SQL Boolean, PL/SQL Record, and PL/SQL Table Types](#)

4.7.7.1 Support for REF CURSOR

Neither the design-time nor run-time components support `REF CURSOR` types directly. The solution is to use a collection of an `OBJECT` type. Because the number of

rows returned by a REF CURSOR is usually unknown, it is best to use a nested table as the collection type. This solution involves using a Java stored procedure to convert a `ResultSet` into an instance of the declared collection type. A sample tutorial illustrating this is provided in the following directory:

`Oracle_Home\bpel\samples\tutorials\122.DBAdapter\ResultSetConverter`

This section comprises the following:

- [Design Time](#)
- [Runtime](#)

4.7.7.1.1 Design Time

PL/SQL cursor variables, known as ref cursors, are supported using `RowSets`. The wizard is incapable of determining what columns are included in the query of a cursor variable at design time. This makes it impossible to generate an XSD that represents the contents of a cursor variable. However, it is possible to generalize the structure of its corresponding `RowSet`.

A `RowSet` consists of zero or more rows. Each row consists of one or more columns where each column corresponds with a column item in the query of the cursor variable. Each column has, among other things, a name and a SQL datatype. It is therefore possible to create a definition in the XSD that corresponds with this generalization. Consider the following example:

```
<complexType name="RowSet">
  <sequence>
    <element name="Row" minOccurs="0" maxOccurs="unbounded" nillable="true">
      <complexType>
        <sequence>
          <element name="Column" maxOccurs="unbounded" nillable="true">
            <complexType>
              <simpleContent>
                <extension base="string">
                  <attribute name="name" type="string"
use="required"/>
                  <attribute name="sqltype" type="string"
use="required"/>
                </extension>
              </simpleContent>
            </complexType>
          </element>
        </sequence>
      </complexType>
    </element>
  </sequence>
</complexType>
```

Notice in the XSD that the XML Schema type, *string*, represents all column values. This is acceptable for all of the primitive datatypes such as `NUMBER`, `INTEGER` and `TIMESTAMP`.

Unfortunately, a simple *string* doesn't capture the structure of user-defined datatypes such as `Object`, `Varray` and `Nested Table`. Because the structure of these datatypes is unknown when the XSD is generated, it is not possible to generate definitions for them.

However, it is possible to ascertain the structure of user-defined datatypes at runtime and generate a *string* that contains the value of the datatype. The value of these

datatypes will therefore be an XML string that resembles their actual structure. Because this value is itself an XML string, it will appear as fully escaped XML in the corresponding element of the generated XML.

4.7.7.1.2 Runtime

Suppose you have the following package:

```
CREATE PACKAGE PKG AS
    TYPE REF_CURSOR IS REF CURSOR;
    PROCEDURE TEST(C OUT REF_CURSOR);
END;

CREATE PACKAGE BODY PKG AS
    PROCEDURE TEST(C OUT REF_CURSOR) AS
    BEGIN
        OPEN C FOR SELECT DEPTNO, DNAME FROM DEPT;
    END;
END;
```

The REF_CURSOR is a weakly typed cursor variable because the query is not specified. After the procedure executes, the following XML will be generated for parameter, C:

```
<C>
  <Row>
    <Column name="DEPTNO" sqltype="NUMBER">10</Column>
    <Column name="DNAME" sqltype="VARCHAR2">ACCOUNTING</Column>
  </Row>
  <Row>
    <Column name="DEPTNO" sqltype="NUMBER">20</Column>
    <Column name="DNAME" sqltype="VARCHAR2">RESEARCH</Column>
  </Row>
  ...
</C>
```

There will be a total of four rows, each consisting of two columns, DEPTNO and DNAME.

Ref cursors are represented by Java ResultSets. It is not possible to create a ResultSet programmatically using APIs provided by the JDBC driver. Therefore, ref cursors may not be passed IN to a stored procedure. They can only be passed as IN/OUT and OUT parameters with one caveat. An IN/OUT ref cursor will be treated strictly as an OUT parameter. Because no IN value can be provided, a null will be bound when invoking the stored procedure.

4.7.7.2 Support for PL/SQL Boolean, PL/SQL Record, and PL/SQL Table Types

The wizard provides a mechanism that detects when these types are used and then invokes Oracle JPublisher to generate the necessary wrappers automatically. Oracle JPublisher generates two SQL files, one to create schema objects, and another to drop them. The SQL that creates the schema objects is automatically executed from within the wizard to create the schema objects in the database schema before the XSD is generated. For example, suppose the following package specification is declared:

```
CREATE PACKAGE PKG AS
    TYPE REC IS RECORD (X NUMBER, Y VARCHAR2 (10));
    TYPE TBL IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
    PROCEDURE PROC (R REC, T TBL, B BOOLEAN);
END;
```

Figure 4–83 shows the step in the wizard that is displayed when procedure PROC from package PKG is selected.

Figure 4–83 Specifying a Stored Procedure in the Adapter Configuration Wizard

Adapter Configuration Wizard - Step 6 of 6: Specify Stored Procedure

Enter a stored procedure, or a function. The procedure's package name can be included, for example, EMPLOYEE.GET_NAME, where the package name is EMPLOYEE and the procedure is GET_NAME. If the procedure does not belong in a package, enter the procedure's name. You can also browse and search for a procedure. The term 'procedure' is used to mean both stored procedures as well as functions.

Schema: <Default Schema>

Procedure: PKG.PROC Browse...

Arguments

Name	Type	In/Out	Position
R	REC	IN	1
T	TBL	IN	2
B	BOOLEAN	IN	3

A wrapper procedure will be created in a package, enter a package name.

bpel_UseJPub

☐ Overwrite specified wrapper package, if it exists

Help < Back Next > Finish Cancel

As [Figure 4–83](#) shows, the original procedure name is fully qualified, `PKG.PROC`. The type of parameter, `R`, is the name of the RECORD. The type of `T` is the name of the TABLE. The type of `B` is `BOOLEAN`. The name of the wrapper package that is generated is derived from the service name, `bpel_ServiceName` (for example, `bpel_UseJPub`). This is the name of the generated package that contains the wrapper procedure. The check box can be used to force the wizard to overwrite an existing package when the schema objects are created.

Clicking **Next** reveals the **Finish** page of the wizard, as shown in [Figure 4–84](#).

Figure 4–84 Defining a Database Adapter Service: Finish Page

Adapter Configuration Wizard - Finish

You have finished defining Database Adapter Service : UseJPub

When you click Finish, the wizard will create the C:\JDev10.1.3\jdev\mywork\Application\Documentation\bpel\UseJPub.wsd file in your project directory. An XSD file, SCOTT_BPTEL_USEJPUB_PKG-24PROC.xsd, which contains the schema for the procedures arguments, will also be created.

The selected procedure requires that a wrapper procedure be created because it has one or more argument(s) of type: PL/SQL Boolean, PL/SQL Record, or PL/SQL Table. SQL files will be created in your project directory. It may take a while to create the wrapper procedure.

Wrapper Procedure: PKG\$PROC
Wrapper Package: BPEL_USEJPUB
Schema: SCOTT

SQL Files: BPEL_USEJPUB.sql
 BPEL_USEJPUB_drop.sql

Help < Back Next > Finish Cancel

The contents of this page describe what the wizard has detected and what actions are performed when the **Finish** button is clicked. The following summarizes the contents of this page:

1. The name of the generated WSDL is `UseJPub.wsdl`.
2. Two SQL scripts are created and added to the BPEL process project:
 - a. `BPEL_USEJPUB.sql` – Creates the schema objects.
 - b. `BPEL_USEJPUB_drop.sql` – Drops the schema objects.
3. The name of the generated XSD is `SCOTT_USEJPUB_PKG-24PROC.xsd`.

When you click **Finish**, Oracle JPublisher is invoked to generate the SQL files and load the schema objects into the database. The process of generating wrappers may take quite some time to complete. Processing times for wrappers that are generated in the same package usually require less time after an initial wrapper has been generated for another procedure within the same package.

The following user-defined types are generated to replace the PL/SQL types from the original procedure:

```
SQL> CREATE TYPE PKG_REC AS OBJECT (X NUMBER, Y VARCHAR2 (10));
SQL> CREATE TYPE PKG_TBL AS TABLE OF NUMBER;
```

The naming convention for these types is *OriginalPackageName_OriginalTypeName*. `BOOLEAN` is replaced by `INTEGER` in the wrapper procedure.

Acceptable values for the original `BOOLEAN` parameter now that it is an `INTEGER` are 1 for true and 0 for false. Any value other than 1 is considered false. The generated wrapper procedure uses APIs from the `SYS.SQLJUTL` package to convert from `INTEGER` to `BOOLEAN` and vice-versa.

A new wrapper package called `BPEL_USEJPUB` is created that contains the wrapper for procedure `PROC`, called `PKG$PROC`, as well as conversion APIs that convert from the PL/SQL types to the user-defined types and vice-versa. If the original procedure is a root-level procedure, the name of the generated wrapper procedure is `TOPLEVEL$OriginalProcedureName`.

The generated XSD represents the signature of wrapper procedure `PKG$PROC` and not the original procedure. The name of the XSD file is URL-encoded, which replaces \$ with -24.

Note the naming conventions for the generated artifacts:

- The service name is used in the names of the WSDL and SQL files. It is also used as the name of the wrapper package.
- The name of the generated XSD is derived from the schema name, service name, and the original package and procedure names.
- The name of a user-defined type is derived from the original package name and the name of its corresponding PL/SQL type.
- The name of the wrapper procedure is derived from the original package and procedure names. `TOPLEVEL$` is used for root-level procedures.

The name of the generated wrapper package is limited to 30 characters. The name of the wrapper procedure is limited to 29 characters. If the names generated by Oracle JPublisher are longer than these limits, they are truncated.

When the PartnerLink that corresponds with the service associated with the procedure is invoked, the generated wrapper procedure is executed instead of the original procedure.

4.7.7.2.1 Default Clauses in Wrapper Procedures

If a procedure contains a special type that requires a wrapper to be generated, the default clauses on any of the parameters are *not* carried over to the wrapper. For example, consider

```
SQL> CREATE PROCEDURE NEEDSWRAPPER (
      >      B BOOLEAN DEFAULT TRUE, N NUMBER DEFAULT 0) IS BEGIN ... END;
```

Assuming that this is a root-level procedure, the signature of the generated wrapper procedure is

```
TOPLEVEL$NEEDSWRAPPER (B INTEGER, N NUMBER)
```

The `BOOLEAN` type has been replaced by `INTEGER`. The default clauses on both parameters are missing in the generated wrapper. Parameters of generated wrapper procedures never have a default clause, even if they did in the original procedure.

In this example, if an element for either parameter is not specified in the instance XML, then a null value is bound to that parameter during the invocation of the wrapper procedure. The default value of the parameter that is specified in the original procedure is not used.

To address this, the generated SQL file that creates the wrapper must be edited, restoring the default clauses to the parameters of the wrapper procedure. The wrapper and any additional schema objects must then be reloaded into the database schema. After editing the SQL file, the signature of the wrapper procedure is

```
TOPLEVEL$NEEDSWRAPPER (B INTEGER DEFAULT 1, N NUMBER DEFAULT 0)
```

For `BOOLEAN` parameters, the default value for true is 1 and the default value for false is 0.

As a final step, the XSD generated for the wrapper must be edited. A special attribute must be added to elements representing parameters that now have default clauses. Add `db:default="true"` to each element representing a parameter that now has a default clause. For example,

```
<element name="B" ... db:default="true" .../>
<element name="N" ... db:default="true" .../>
```

This attribute is used at run time to indicate that, if the element is missing from the instance XML, the corresponding parameter must be omitted from the procedure call. The remaining attributes of these elements remain exactly the same.

4.8 Use Case for Creating and Configuring a Stored Procedure in JDeveloper BPEL Designer

This tutorial describes how to integrate a stored procedure into Oracle BPEL Process Manager with JDeveloper BPEL Designer. Other tutorials that demonstrate stored procedures and functions are `File2StoredProcedure`, `JPublisherWrapper`, and `ResultSetConverter`. Go to

```
Oracle_Home\bpel\samples\tutorials\122.DBAdapter
```

This section contains the following topics:

- [Creating a Stored Procedure](#)
- [Creating a New BPEL Project](#)
- [Creating a Partner Link](#)
- [Creating an Invoke Activity](#)
- [Creating an Initial Assign Activity](#)
- [Creating a Second Assign Activity](#)
- [Validating, Compiling, and Deploying the Greeting Process](#)
- [Running the Greeting Process](#)

4.8.1 Creating a Stored Procedure

1. Connect to the `scott` schema of the Oracle database using SQL*Plus. This is the schema in which to create the stored procedure. This example assumes `tiger` is the password.

```
sqlplus scott/tiger
```

2. Create the stored procedure (note the blank space after `Hello`):

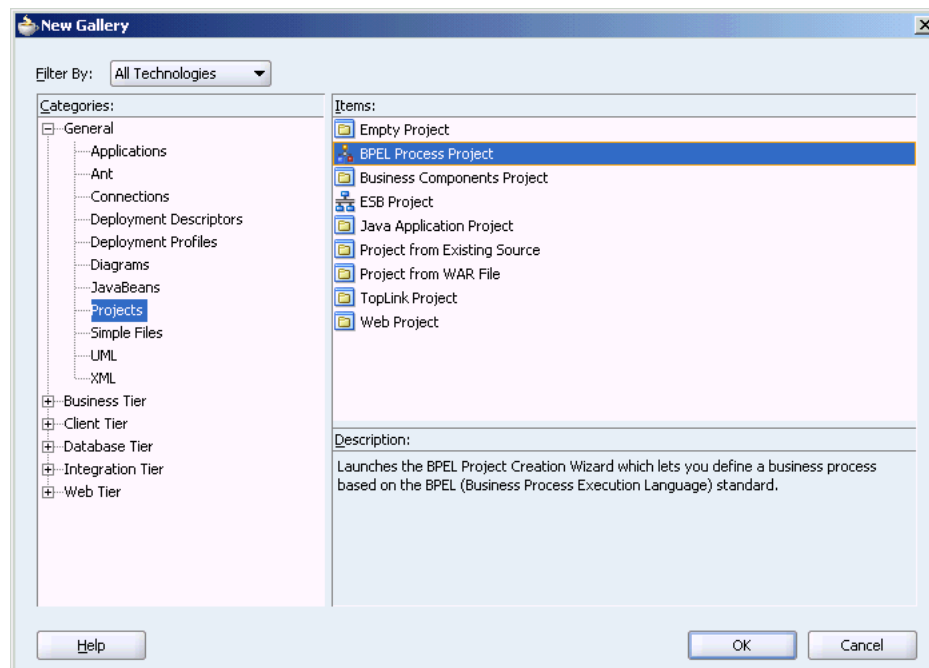
```
SQL> CREATE PROCEDURE HELLO (NAME IN VARCHAR2, GREETING OUT VARCHAR2) AS
2 BEGIN
3     GREETING := 'Hello ' || NAME;
4 END;
5 /
```

```
Procedure created.
```

4.8.2 Creating a New BPEL Project

The following are the steps to create a new BPEL project:

1. Open Oracle JDeveloper.
2. From the **File** menu, select **New**.
The New Gallery dialog box is displayed.
3. Select **All Technologies** from the **Filter By** box. This displays a list of available categories.
4. Expand the **General** node, and then select **Projects**.
5. Select **ESB Project** from the Items group, as shown in [Figure 4-85](#)

Figure 4–85 Creating a New BPEL Project**6. Click OK.**

The BPEL Project Creation Wizard -Project Settings dialog box is displayed.

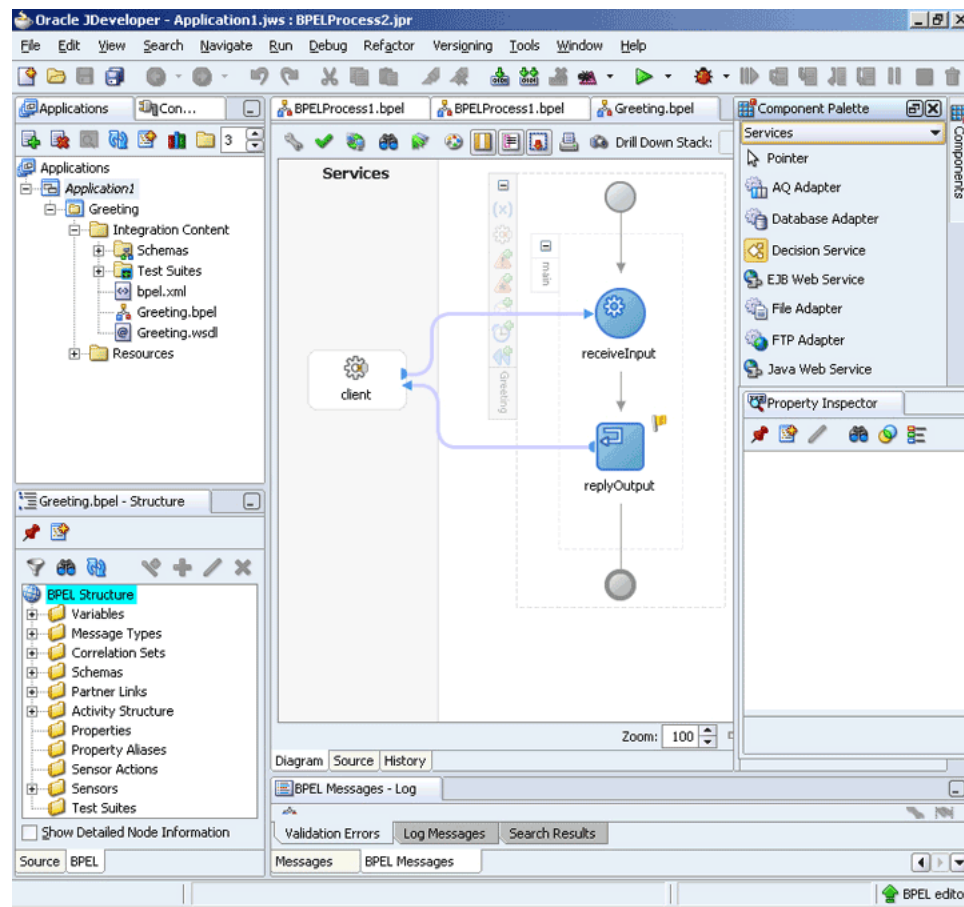
- **Name:** Enter a process name. In this example, type **Greeting**.
- **Namespace:** Retain the default values.
- **Template:** Select **Synchronous BPEL Process**.

7. Click Next.

The BPEL Project Creation Wizard -Input/Output Elements dialog box is displayed.

8. Accept default values, and then click Finish.

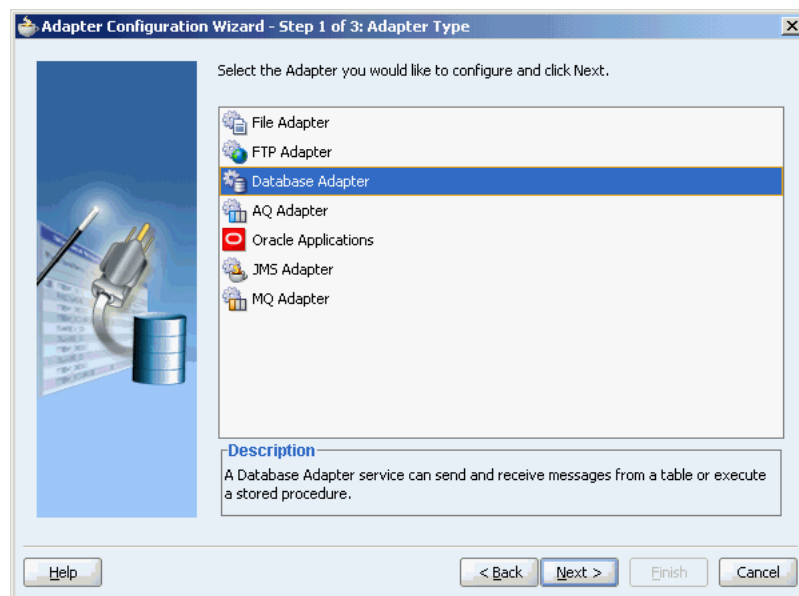
You have completed creating a BPEL Project, as shown in [Figure 4–86](#).

Figure 4–86 The Application Screen

4.8.3 Creating a Partner Link

This section describes how you create a partner link. The following are the steps to create a partner link:

1. From the Components Palette, select **Services**.
2. From the list of services, drag and drop Partner Link onto the BPEL project window.
The Create Partner Link dialog box is displayed.
3. Enter **Hello** in the **Name** field, and then click the **Define Adapter Service** icon (the third icon) under WSDL Settings.
The Adapter Configuration wizard - Welcome screen is displayed.
4. Click **Next**.
The Adapter Type dialog box is displayed.
5. Select **Database Adapter**, as shown in [Figure 4–87](#) and then click **Next**.

Figure 4–87 The Adapter Type Dialog Box

6. Enter **Hello** in the **Service Name** field on the Service Name window, and then click **Next**. This is the same name as that of the partner link.

The Service Connection dialog box is displayed.

7. Click **New** to create a new connection.

The Welcome dialog box is displayed.

8. Click **Next**.

The Type dialog box is displayed.

9. Enter a name (for example, **myConnection**) in the **Connection Name** field of the Type dialog box.

10. Select the database connection type (for example, **Oracle (JDBC)**) from the **Connection Type** list, and click **Next**.

The Authentication dialog box is displayed.

11. Enter **scott** in the **Username** field of the Authentication window.

12. Enter the password for scott in the **Password** field (**tiger** for this example).

13. Leave the remaining fields as they are, and click **Next**.

The Connection dialog box is displayed.

14. Enter the following connection information. If you do not know this information, contact your database administrator.

Field	Example of Value
Driver	thin
Host Name	localhost
JDBC Port	1521
SID	ORCL

15. Click Next.

The Test dialog box is displayed.

16. Click Test Connection on the Test window.

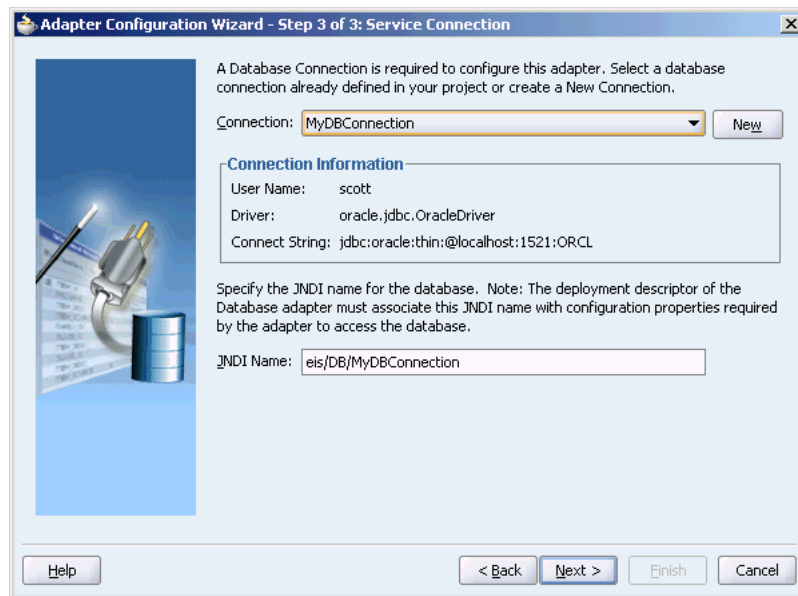
If the connection was successful, the following message appears:

Success!

17. Click Finish.

The Service Connection dialog box is displayed, with all fields populated, as shown in [Figure 4–88](#).

Figure 4–88 The Service Connection Dialog Box

**18. Click Next.**

The Operation Type dialog box is displayed.

19. Select Call a Stored Procedure or Function on the Operation Type window, and then click **Next**.

The Specify Stored Procedure dialog box is displayed.

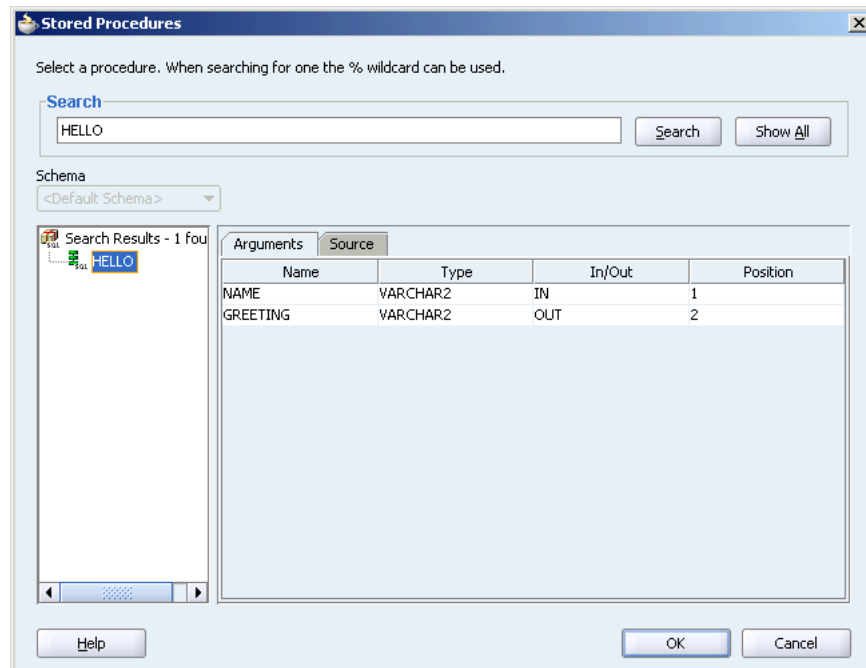
20. Click Browse to the right of the **Procedure** field.

The Stored Procedures dialog box is displayed.

21. Leave <Default Schema> selected in the **Schema** list. This defaults to the `scott` schema in which the stored procedure is defined.**22. Select Hello** in the **Stored Procedures** navigation tree.

Note: As an alternative, you can also enter **Hello** in the **Search** field, click **Search** to display this stored procedure for selection in the **Stored Procedures** navigation tree, and then select it.

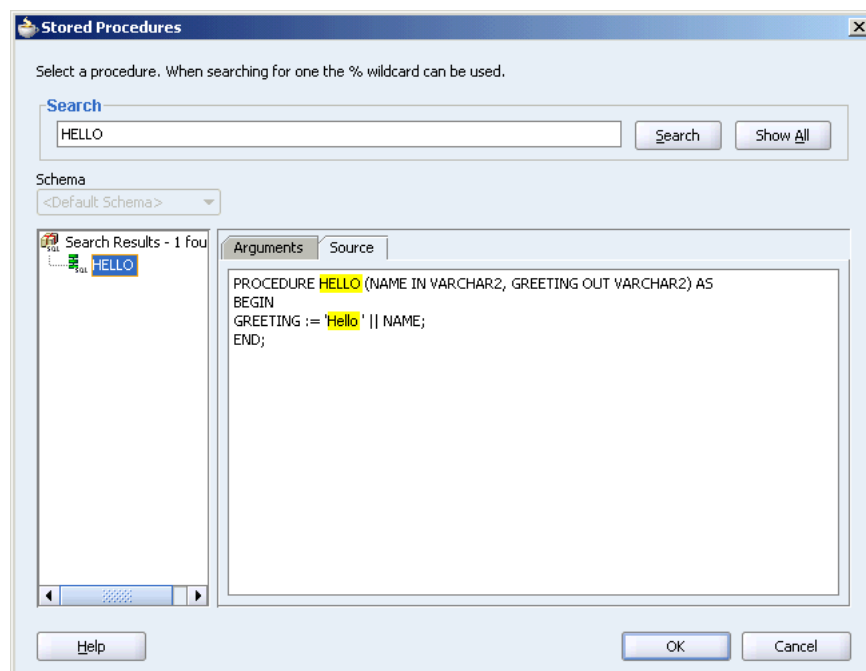
The **Arguments** tab displays the parameters of the stored procedure.



23. Click the **Source** tab to display the **Hello** stored procedure source code. You entered this syntax when you created the stored procedure using SQL*Plus in [Creating a Stored Procedure](#).

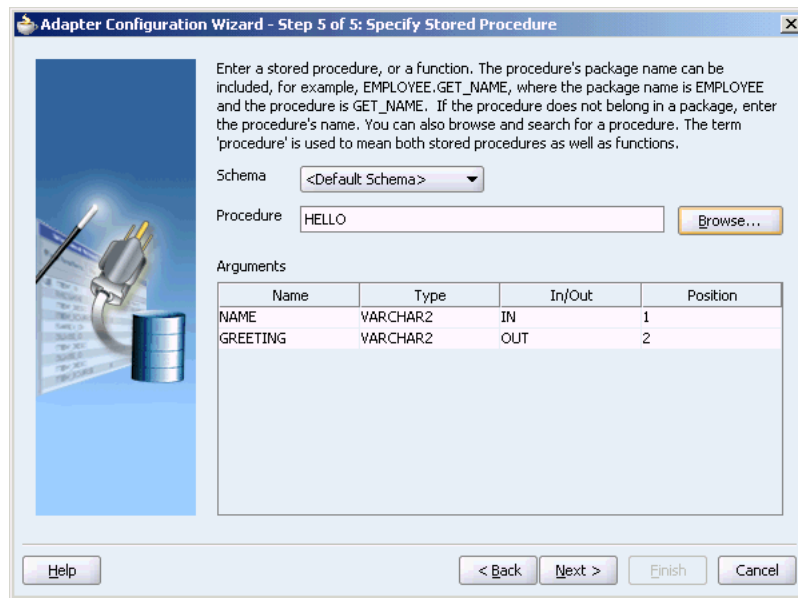
Figure 4–89 shows the source code of the stored procedure Hello.

Figure 4–89 Source Code of Stored Procedure Hello



24. Click **OK**.

The Specify Stored Procedure window displays your selections. They appear as shown in [Figure 4–90](#).

Figure 4–90 The Specify Stored Procedure Dialog Box

25. Click **Next**.

The Finish screen is displayed.

26. Click **Finish** to complete adapter configuration.

The Create Partner Link window is automatically completed.

27. Click **Apply**.

28. Click **OK**.

29. Select **Save All** from the **File** main menu.

The following files appear under **Greeting > Integration Content** in the **Applications Navigator**. These files contain the parameters you specified with the Adapter Configuration Wizard.

- **Hello.wsdl**

Corresponds with the new stored procedure partner link

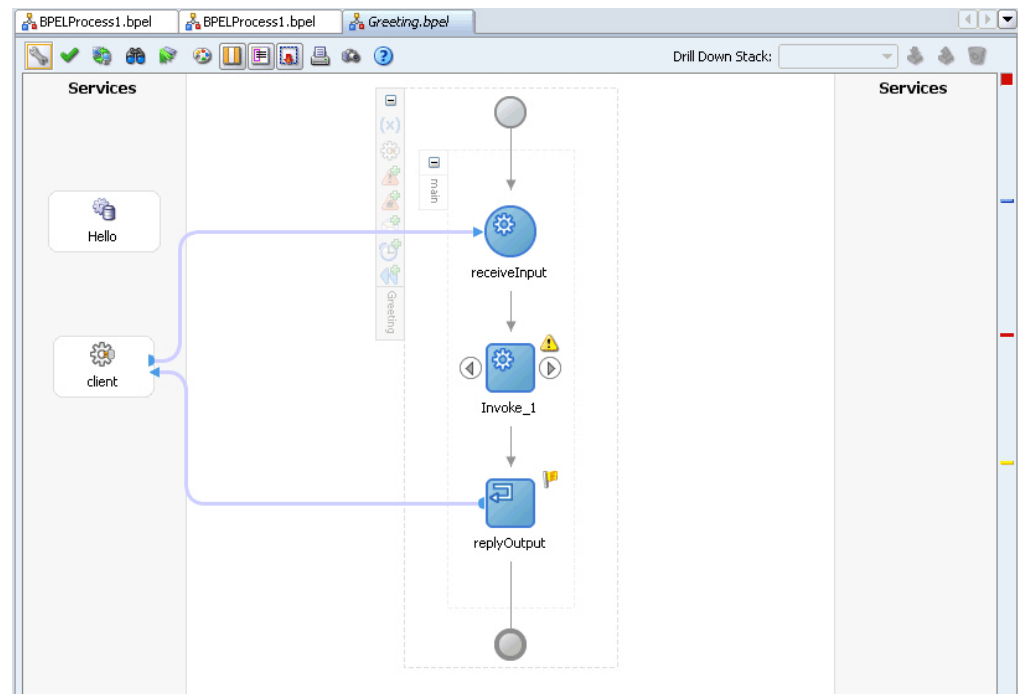
- **SCOTT_HELLO.xsd**

Provides the definition of the stored procedure, including its parameters

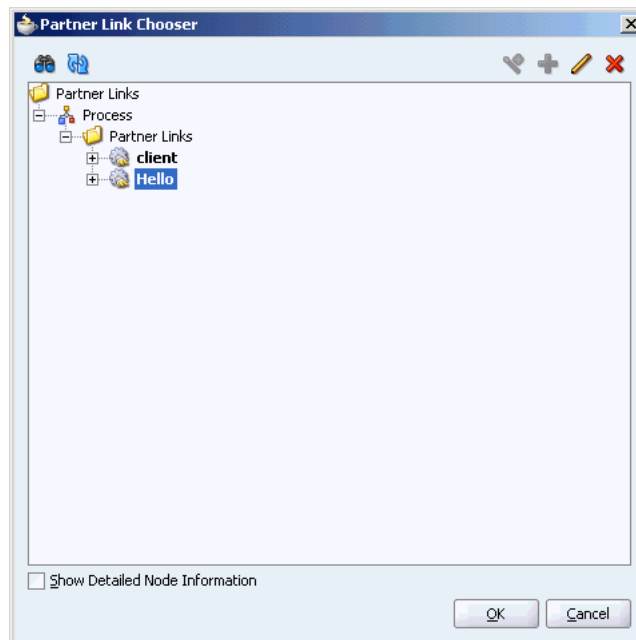
4.8.4 Creating an Invoke Activity

You now create an **invoke** activity to specify an operation you want to invoke for the service (identified by the **Hello** partner link).

1. Drag and drop an **invoke** activity below **receiveInput** activity, as shown in fig.

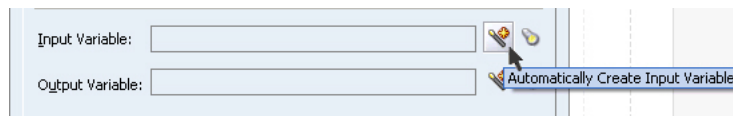
Figure 4–91 The Invoke Activity

2. Double-click the **invoke** activity to display the Invoke dialog box.
3. In the Invoke dialog box, specify the perform the following actions:
 - a. In the **Name** field type **Greet**.
 - b. Click the **Browse PartnerLinks** icon on the right of the **Partner Link** field.
The Partner Link Chooser dialog box is displayed.
 - c. Select **Hello**, as shown in [Figure 4–92](#), and then click **OK**.

Figure 4–92 The Partner Link Chooser Dialog Box

The Invoke dialog box is displayed with the **Name**, **Partner Link**, and **Operation** fields filled in. The **Operation** field automatically filled in with the value, **Hello**.

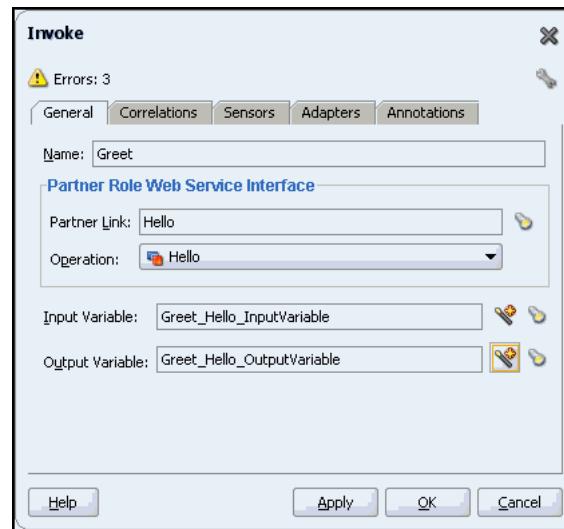
4. Click the **Automatically Create Input Variable** icon, which is the first icon to the right of the **Input Variable** field.



The Create Variable dialog box is displayed. A variable named **Greet_Hello_InputVariable** automatically appears in the **Name** field. This variable provides the value for the **in** parameter of the stored procedure. The type is **http://xmlns.oracle.com/pcbpel/adapter/db/Hello/args_in_msg**.

5. Ensure that **Global Variable** is selected.
6. Click **OK** on the Create Variable dialog box.
7. Click the first icon to the right of the **Output Variable** field.
8. A variable named **Greet_Hello_OutputVariable** automatically appears in the **Name** field. This variable stores the value of the **out** parameter of the procedure after it executes. The type is **http://xmlns.oracle.com/pcbpel/adapter/db/Hello/args_out_msg**.
9. Ensure that **Global Variable** is selected.
10. Click **OK** in the Create Variable dialog box.

Your selections for the Invoke window appears, as shown in [Figure 4–93](#).

Figure 4–93 The Invoke Window

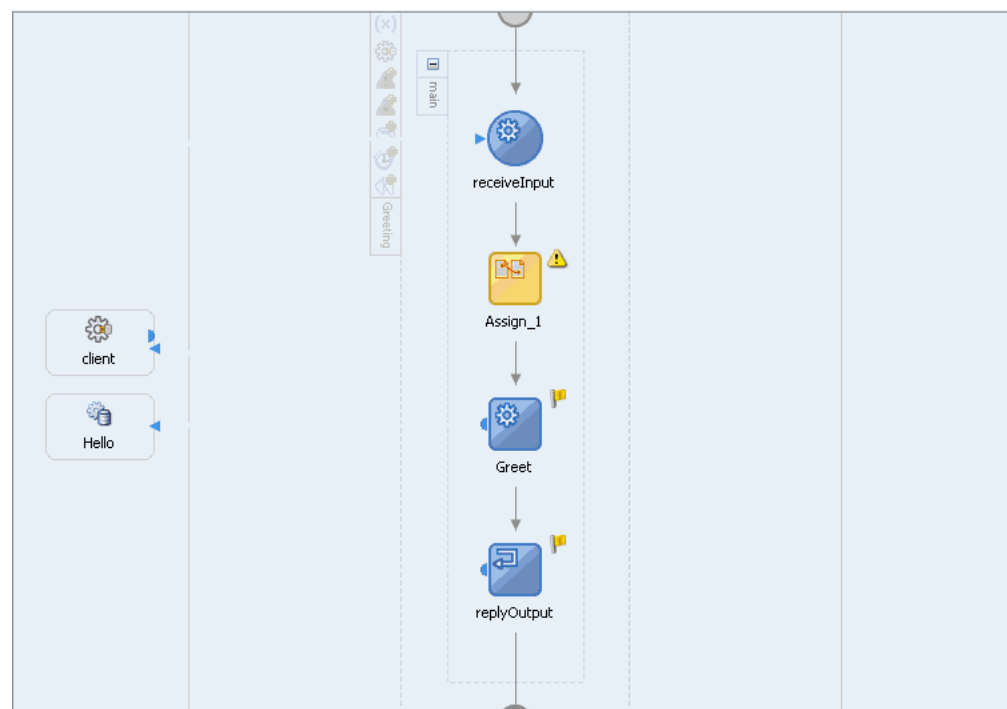
11. Click **OK** in the Invoke window.
12. Select **Save All** from the **File** main menu.

The process displays a link from the **Greet Invoke** activity to the **Hello** partner link.

4.8.5 Creating an Initial Assign Activity

You now create an Assign activity to assign the input value to the `in` parameter of the stored procedure.

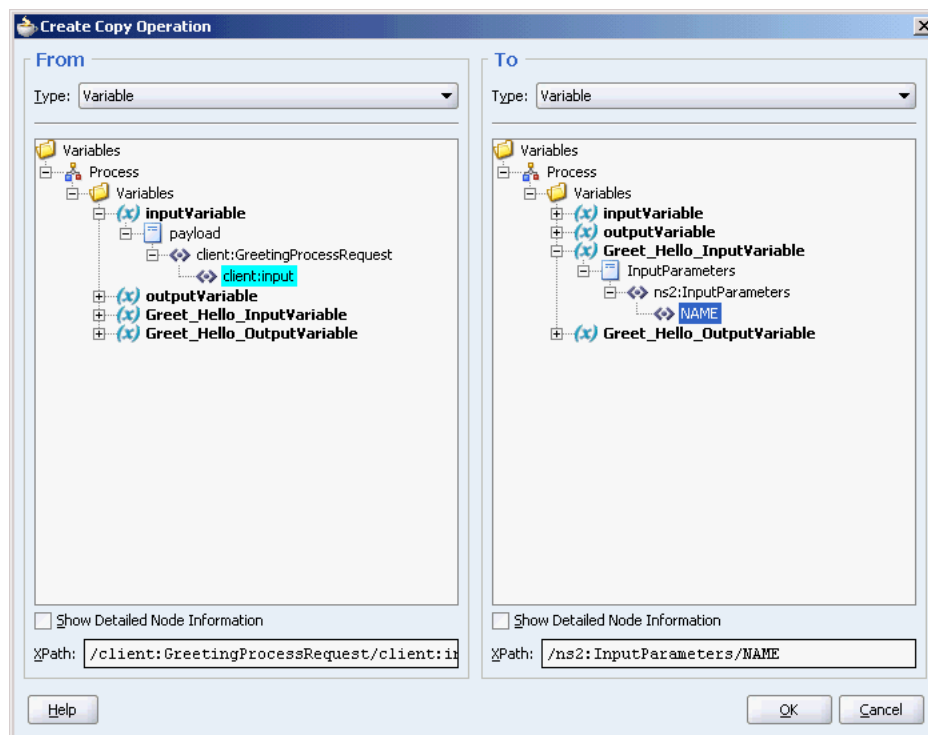
1. Drag and drop an **Assign** activity from the **Component Palette** section above the **Greet Invoke** activity.



2. Double-click the **assign** icon to display the Assign window.
3. Click the **General** tab.
4. Enter **Input** in the **Name** field.
5. Click **Apply**.
6. Click the **Copy Rules** tab.
7. Select **Copy Operation** from the **Create** drop-down list.
The Create Copy dialog box is displayed.
8. Enter the following values:

Field	Value
From	
■ Type	Variable
■ Variables	Expand and select Variables , then inputVariable , then payload , then client:GreetingProcessRequest , and then client:input .
To	
■ Type	Variable
■ Variables	Expand and select Variables , then Greet_Hello_InputVariable , then InputParameters , then ns2:InputParameters , and then NAME .

The Create Copy Operation Dialog Box appears as follows:

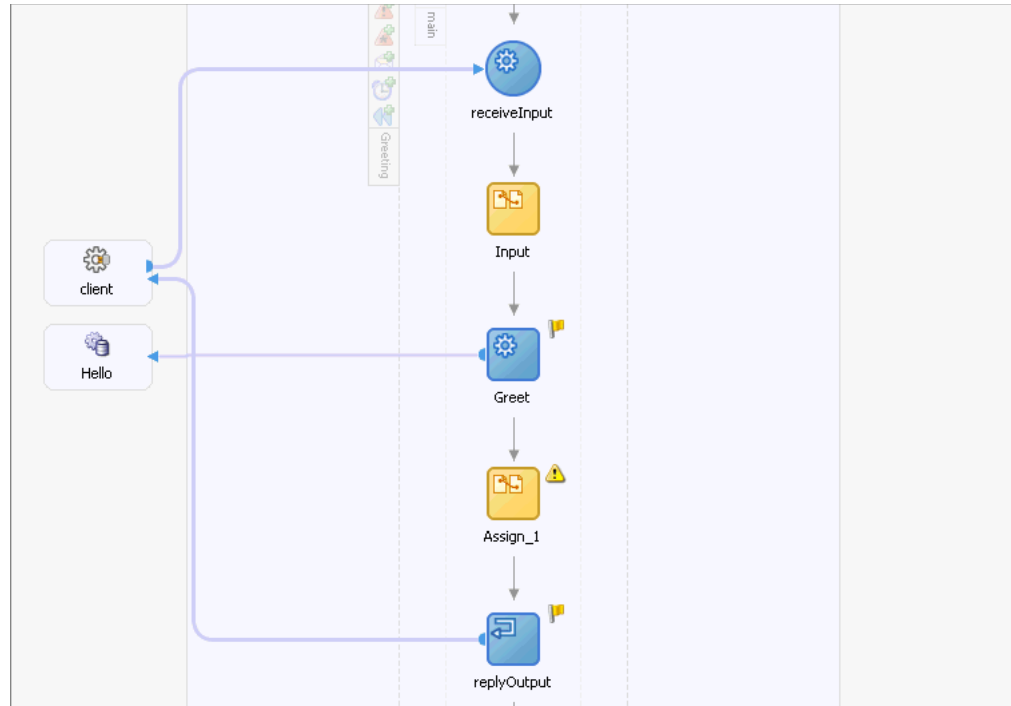


9. Click **OK** to close the Create Copy Rule window.
10. Click **OK** to close the Assign window.
11. Select **Save All** from the **File** main menu.

4.8.6 Creating a Second Assign Activity

You now create an Assign activity to retrieve the value of the out parameter of the stored procedure.

1. Drag and drop an **Assign** activity from the **Component Palette** section below the **Greet Invoke** activity.



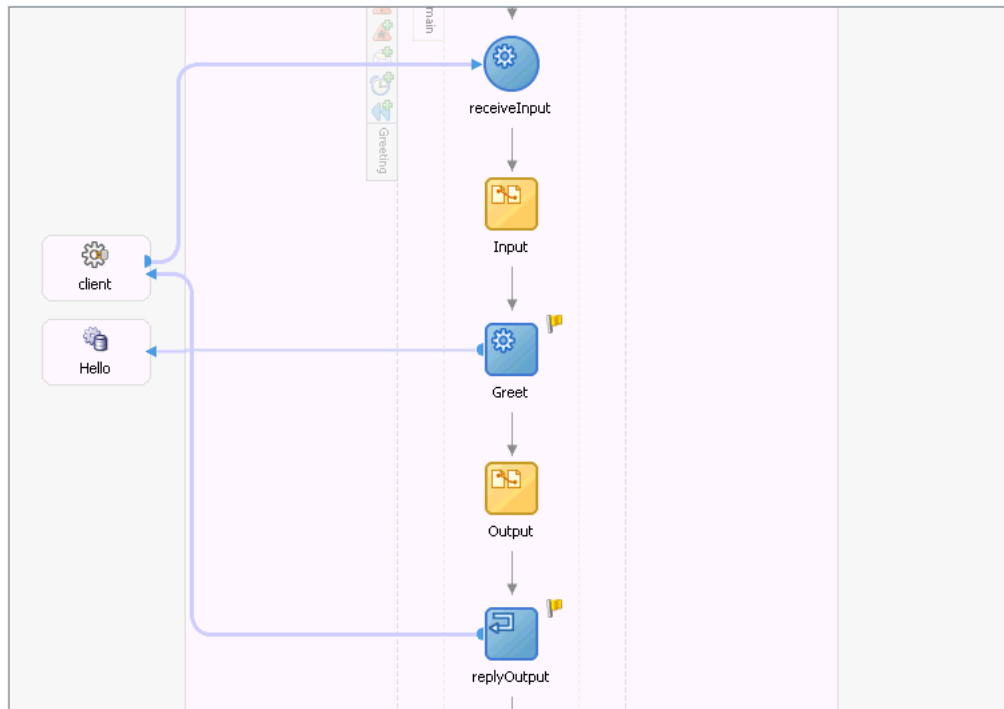
2. Double-click the **assign** icon to display the Assign window.
3. Click the **General** tab.
4. Enter **Output** in the **Name** field.
5. Click **Apply**.
6. Click the **Copy Rules** tab.
7. Click **Create** to display the Create Copy Operation dialog box.
8. Enter the following values:

Field	Value
From	
■ Type	Variable
■ Variables	Expand and select Variable , then Greet_Hello_OutputVariable , then OutputParameters , then ns2:OutputParameters , and then GREETING .
To	
■ Type	Variable
■ Variables	Expand and select Variables , then outputVariable , then payload , then client:GreetingProcessResponse , and then client:result .

9. Click **OK** to close the Create Copy Operation dialog box.

- Click **OK** to close the Assign window.

The **Greeting** process appears as follows in JDeveloper BPEL Designer.



- Select **Save All** from the **File** main menu.

4.8.7 Validating, Compiling, and Deploying the Greeting Process

- Go to the **Applications Navigator** section.
- Right-click **Greeting**.
- Select **Deploy**, then **LocalBPELServer**, and then **Deploy to default domain**.
- Enter the domain password (initially set to **bpel**) when prompted.
- Click **OK**.

This compiles the BPEL process. Review the bottom of the window for any errors. If there are no errors, deployment was successful.

4.8.8 Running the Greeting Process

- Log in to Oracle BPEL Control using Internet Explorer by selecting **Start**, then **All Programs**, then **Oracle - Oracle_Home**, then **Oracle BPEL Process Manager**, and then **Oracle BPEL Control**, or by running the `$ORACLE_HOME/bpel/bin/startorabpel.sh` script for UNIX.
- Enter the password (initially set to **bpel**) when prompted.
The **Dashboard** tab of Oracle BPEL Control appears. Note that your BPEL process, **Greeting**, now appears in the **Deployed BPEL Processes** list.
- Click **Greeting**.
The Testing this BPEL Process page appears with the **Initiate** tab selected.
- Enter your first name in the **input** field (for example, **John**).

5. Click Post XML Message.

After the procedure executes and the BPEL process finishes the value appears as follows:

```
Value: <GreetingProcessResponse>
      <result>Hello John</result>
</GreetingProcessResponse>
```

6. Click Audit Instance.

The **Instances** tab of Oracle BPEL Control appears, along with the sequence of process activities.

7. Click More... on the Greet activity to see the input to and output from the stored procedure.

Note the <NAME> tag and its value in the <InputParameters> element. This value came from the **inputVariable** and was set by the **Input Assign** activity.

Note the <GREETING> tag and its value in the <OutputParameters> element. This value came from the output parameter of the stored procedure. The value was then assigned to the outputVariable by the Output Assign activity.



8. Click the Flow tab to view the process flow.

The process diagram appears.

9. Click any of the activities to view the XML as it passed through the BPEL process. For example, click the Greet Invoke activity to display the following:

Invoked 2-way operation "Hello" on partner "Hello".

```
<messages>
<Greet_Hello_InputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  name="InputParameters">
  <InputParameters
    xmlns="http://xmlns.oracle.com/pcbpel/adapter/db/SCOTT/HELLO/">
    <NAME xmlns="">John</NAME>
  </InputParameters>
</part>
</Greet_Hello_InputVariable>
<Greet_Hello_OutputVariable>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  name="OutputParameters">
  <db:OutputParameters
    xmlns:db="http://xmlns.oracle.com/pcbpel/adapter/db/SCOTT/HELLO/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <GREETING>Hello John</GREETING>
  </db:OutputParameters>
</part>
<part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  name="response-headers">null</part>
</Greet_Hello_OutputVariable>
</messages>
```

Oracle Application Server Adapter for Java Message Service

This chapter describes how to use the Oracle Application Server Adapter for Java Message Service (JMS adapter), which enables a BPEL process or an ESB service to interact with JMS.

This chapter contains the following topics:

- [Section 5.1, "Introduction to the JMS Adapter"](#)
- [Section 5.2, "JMS Adapter Use Cases for Oracle BPEL Process Manager"](#)
- [Section 5.3, "JMS Adapter Use Cases for Oracle Enterprise Service Bus"](#)

5.1 Introduction to the JMS Adapter

The JMS architecture uses one client interface to many messaging servers. The JMS model has two messaging domains: point-to-point and publish-subscribe. In the point-to-point domain, messages are exchanged through a queue and each message is delivered to only one receiver. In the publish-subscribe model, messages are sent to a topic and can be read by many subscribed clients. For JMS adapter example files, go to

Oracle_Home\bpel\samples\tutorials\123.JmsAdapter

This section comprises the following topics:

- [JMS Adapter Features](#)
- [JMS Adapter Integration with Oracle BPEL Process Manager](#)
- [JMS Adapter Integration with Oracle Enterprise Service Bus](#)

5.1.1 JMS Adapter Features

The JMS adapter includes the following features:

- Based on JMS version 1.0.2b
- Is a generic JMS adapter and can work with any JMS provider. It has been certified against OEMS JMS, TIBCO JMS, and IBM Websphere MQSeries. (JMS providers OJMS 8.1.7, 9.0.1.4, and 9.2, and IBM MQSeries JMS 5.2 and 5.3).
- Supports JMS topics and queues
- Supports byte and text message types only in this release. The Adapter Configuration Wizard provides the Native Format Builder Wizard for consuming native data payloads at run time. The Native Format Builder Wizard creates XSD definitions for the underlying native data.

- Supports JMS headers and properties
- Supports the JMS message selector for performing filtering while subscribing to JMS topics and queues. This parameter is based on the SQL 92 language for filtering messages based on fields present in the JMS header and properties section.
- Supports specifying a durable JMS subscriber
- Supports persistent and nonpersistent modes of a JMS publisher
- Connection retry functionality for MQ provider is currently not supported
- Outbound retry functionality for AQJMS on Solaris is not supported

Note: When you use the JMS adapter to connect to an AQJMS provider, and if the database that hosts the AQ destination is 10.1.0.4, then the adapter retry mechanism on the outbound direction will fail to reconnect to the database server if the database server goes down. This is because of a client JDBC issue with `ojdbc14.jar`. The resolution to this issue involves downloading the 10.1.0.4 JDBC drivers and using them in the mid tier by replacing the libraries, specifically `ojdbc14.jar` in `$MIDTIER_ORACLE_HOME/jdbc`. To resolve this issue, refer to Metalink Note 317385.1 for detailed explanation.

- The JMS API specifies three types of acknowledgments that can be sent by the JMS publisher:
 - `DUPS_OK_ACKNOWLEDGE`, for consumers that are not concerned about duplicate messages
 - `AUTO_ACKNOWLEDGE`, in which the session automatically acknowledges the receipt of a message
 - `CLIENT_ACKNOWLEDGE`, in which the client acknowledges the message by calling the message's `acknowledge` method

A transaction enables an application to coordinate a group of messages for production and consumption, treating messages sent or received as a single unit. When an application commits a transaction, all messages it received within the transaction are removed by the JMS provider. The messages it sent within the transaction are delivered as one unit to all JMS consumers. If the application rolls back the transaction, the messages it received within the transaction are returned to the messaging system and the messages it sent are discarded. The JMS adapter supports JMS transactions. A JMS-transacted session supports transactions that are located within the session. A JMS-transacted session's transaction does not have any effects outside of the session.

5.1.2 JMS Adapter Integration with Oracle BPEL Process Manager

Adapter Framework is used for the bidirectional integration of the J2CA 1.5 resource adapters with BPEL Process Manager. Adapter Framework is based on standards and employs the Web service Invocation Framework (WSIF) technology for exposing the underlying J2CA interactions as Web services.

See *Oracle Application Server Adapter Concepts* for information on JMS adapter architecture, adapter integration with Oracle BPEL Process Manager, and adapter deployments.

5.1.3 JMS Adapter Integration with Oracle Enterprise Service Bus

Oracle Enterprise Service Bus supports the Oracle adapters and enables you to define inbound and outbound adapter services for each. An inbound adapter service receives data from an external data source and transforms it into an XML message. An outbound adapter service sends data to a target application by transforming an XML message into the native format of the given adapter.

In the case of JMS adapter service, using Oracle Enterprise Service Bus you can send or receive messages from a JMS queue or topic.

BPEL pre-dates ESB and most of this guide and the samples implicitly assume use with BPEL. However, the adapters work equally well with either BPEL or ESB. For any mention of BPEL here you may substitute ESB instead.

5.2 JMS Adapter Use Cases for Oracle BPEL Process Manager

This section comprises the following two use cases:

- [Case One: Configuring a JMS Adapter](#)
- [Case Two: MQSeries Queue Integration Through the OracleAS Adapter for JMS](#)

5.2.1 Case One: Configuring a JMS Adapter

The following use cases demonstrate the procedure for configuring a JMS adapter and examines the resulting WSDL files and associated `oc4j-ra.xml` files.

This section comprises the following topics:

- [Concepts](#)
- [Using the Adapter Configuration Wizard to Configure a JMS Adapter](#)
- [Generated WSDL File](#)
- [oc4j-ra.xml file](#)
- [Produce Message Procedure](#)
- [Configuring for OJMS](#)
- [Configuring for OC4J JMS](#)
- [Configuring for TIBCO JMS](#)
- [Configuring for IBM Websphere JMS](#)

5.2.1.1 Concepts

Messaging is any mechanism that allows communication between programs. Messages are structured data that one application sends to another. Message-oriented middleware (MOM) is an infrastructure that supports scalable enterprise messaging. MOM ensures fast, reliable asynchronous communication, guaranteed message delivery, receipt notification, and transaction control. JMS is a Java interface developed by Sun Microsystems for producing, sending, and receiving messages of an enterprise messaging system. JMS is an API that JMS vendors implement. Oracle provides two implementations of JMS: OC4J JMS and Oracle JMS based on Oracle advanced queues. A JMS producer creates JMS messages and a JMS consumer consumes JMS messages.

JMS supports two messaging paradigms: point-to-point (queues) and publish/subscribe (topics).

This section comprises the following topics:

- [Point-to-Point](#)
- [Publish/Subscribe](#)
- [Destination, Connection, Connection Factory, and Session](#)
- [Structure of a JMS Message](#)
- [JMS Header Properties](#)

Point-to-Point

In point-to-point messaging, the messages are stored in a queue until they are consumed. One or more producers write to the queue and one or more consumers extract messages from the queue. The JMS consumer sends an acknowledgment after consumption of a message; this results in purging of the message from the queue.

Publish/Subscribe

In publish/subscribe messaging, producers publish messages to a topic and the consumer subscribes to a particular topic. Many publishers can publish to the same topic, and many consumers can subscribe to the same topic. All messages published to the topic by the producers are received by all consumers subscribed to the topic. By default, subscribers receive messages only when they are active. However, JMS API supports durable subscriptions that ensure that consumers receive messages that were published even when they are not up and running. The durable subscription involves registering the consumer with a unique ID for retrieving messages that were sent when the consumer was inactive. These messages are persisted by the JMS provider and are either sent to the consumer when it becomes active again or purged from storage if the message expires. The JMS producer can be set either to persistent or nonpersistent mode. The messages are not persisted in the latter case and can be used only for nondurable subscriptions.

Note: JMS adapter does not remove durable subscriptions after the BPEL partner link, which was using a particular durable subscription, is no longer using it. You need to remove the durable subscribers manually.

The JMS API supports both synchronous or asynchronous communication for message consumption. In the synchronous case, the consumer explicitly calls the `receive()` method on the topic or queue. In the asynchronous case, the JMS client registers a message listener for the topic or queue and the message is delivered by calling the listener's `onMessage()` method.

Destination, Connection, Connection Factory, and Session

The destination property contains the addressing information for a JMS queue or topic.

Connections represent a physical connection to the JMS provider. The connection factory is used to create JMS connections. A session is used to create a destination, JMS producer, and JMS consumer objects for a queue or topic.

Note: There is a known limitation in OC4J JMS; on a single OC4J JMS connection listener, you cannot have a subscriber and publisher to the same topic. Hence, BPEL processes publishing and subscribing to the same topic cannot have the same connection, that is, they cannot share the same JNDI. The workaround to this issue is either of the following:

1. Use different JNDI connection factories (different location attribute values in the WSDL `jca:address` element)
 2. In the `jca:operation` element of the WSDL specify `useMessageListener=false` (can also be set through Oracle JDeveloper JMS Adapter Wizard)
-

Structure of a JMS Message

The JMS message has a mandatory standard header element, optional properties element, and optional standard payload element. The payload can be a text message, byte message, map message, stream message, or object message. The properties element is JMS provider-specific and varies from one JMS provider to another.

JMS Header Properties

[Table 5–1](#) describes the JMS header properties.

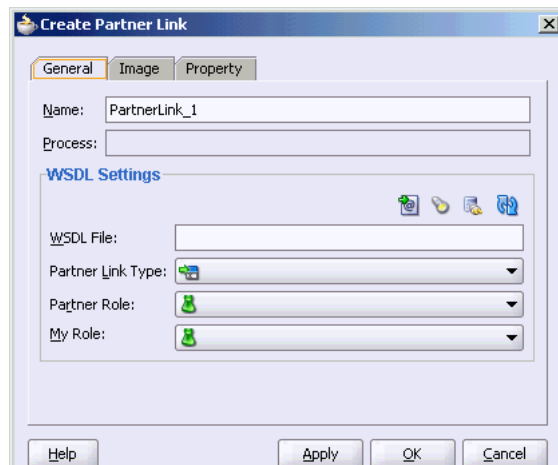
Table 5–1 JMS Header Properties

Property Name	Description
JMSDestination	The destination to which the message is sent, and is set by the JMS producer
JMSDeliveryMode	Set to persistent or nonpersistent mode by the JMS consumer
JMSExpiration	Duration of the message before the expiration is set by the consumer
JMSPriority	Number between 0 and 9 set by the consumer. Larger numbers represent a higher priority.
JMSMessageID	Unique message identifier set by the consumer
JMSTimestamp	Time stamp when the message was sent to the JMS provider for forwarding
JMSCorrelationId	Set by both producers and consumers for linking the response message with the request message. This is an optional attribute.
JMSReplyTo	Optional attribute indicating the destination to which to send a message reply. Can be set by the producer and consumer.
JMSType	JMS message type
JMSRedelivered	Set by the JMS provider to indicate that the provider has tried to send this message once before to the consumer and has failed

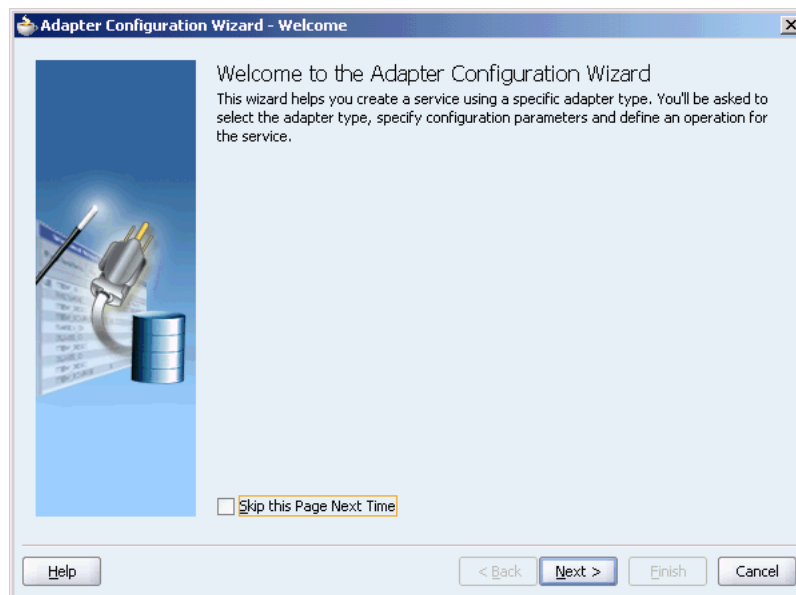
5.2.1.2 Using the Adapter Configuration Wizard to Configure a JMS Adapter

This section describes how to create an adapter service for a partner link.

1. Click **Define Adapter Service** (third icon) in the Create Partner Link window, as shown in [Figure 5–1](#):

Figure 5–1 Create Partner Link Window

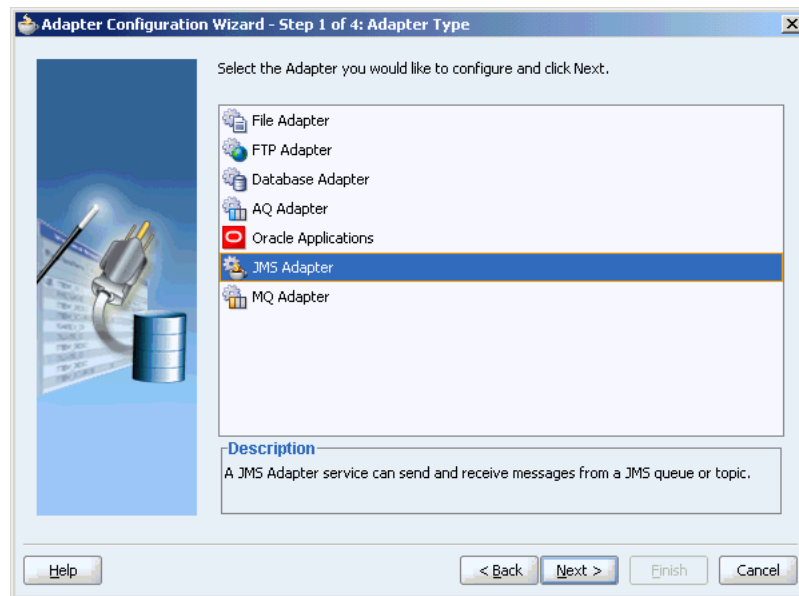
The Adapter Configuration Wizard - Welcome window is displayed, as shown in [Figure 5–2](#).

Figure 5–2 Welcome Window for the Adapter Configuration Wizard

Note: If you do not want to see this window each time you use the Adapter Configuration Wizard, select the **Skip this Page Next Time** check box.

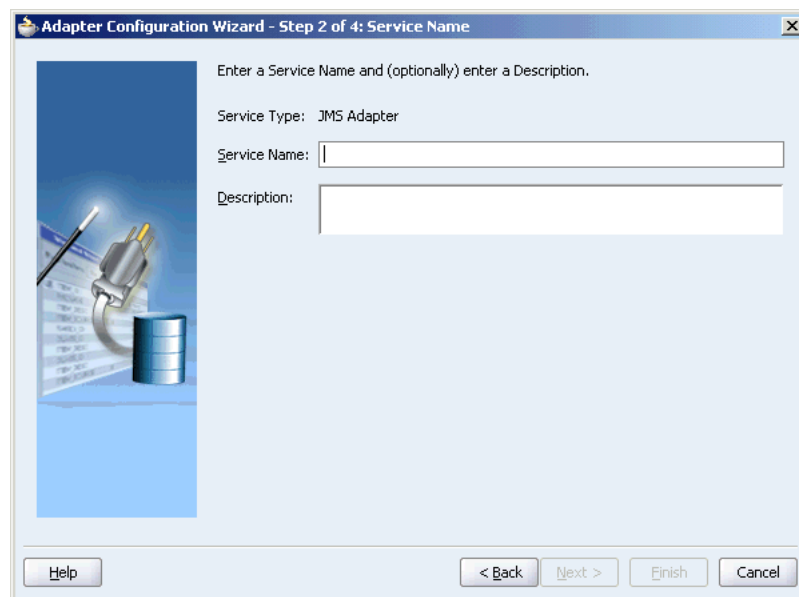
2. Click **Next**.

The Adapter Type dialog box is displayed, as shown in [Figure 5–3](#).

Figure 5–3 Adapter Type Window

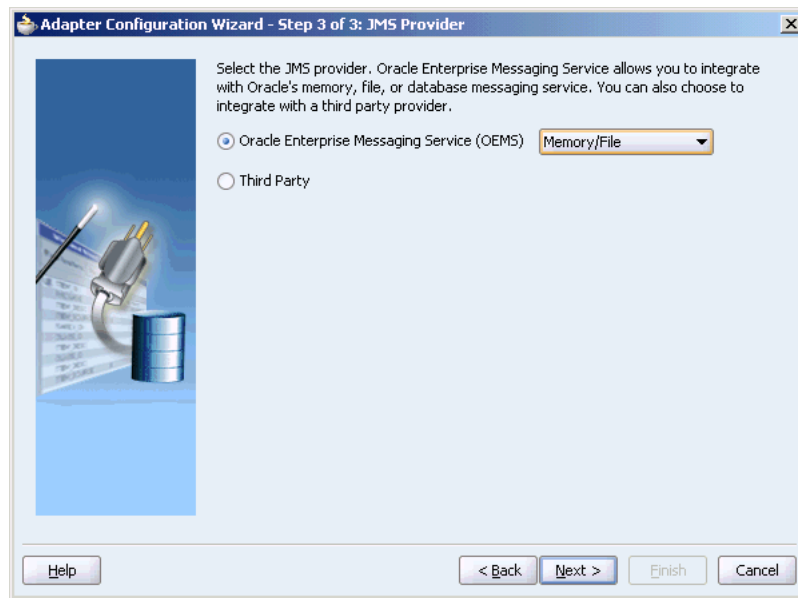
3. Select **JMS Adapter** from the list of available adapter types, and then click **Next**.

The Service Name window is displayed, as shown in [Figure 5–4](#).

Figure 5–4 Service Name Window

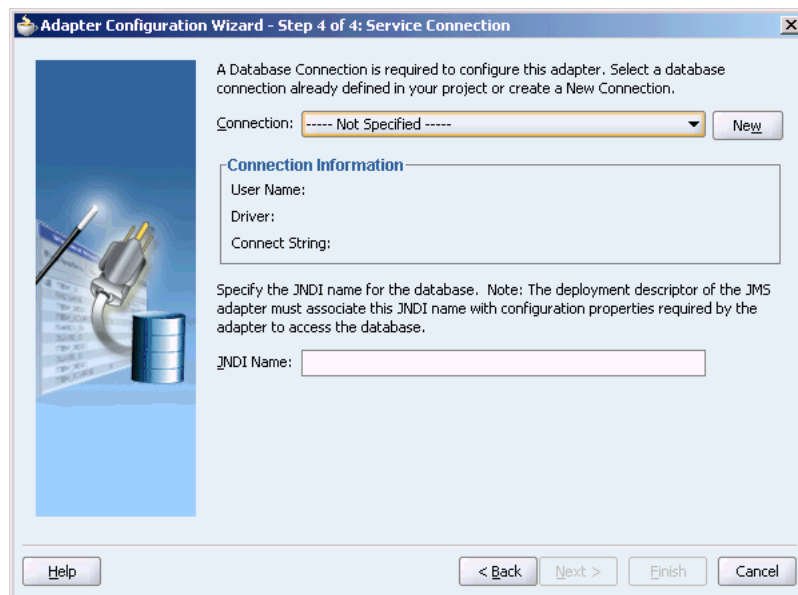
4. Enter a name for the service. You may also add an optional description. Click **Next**.

The JMS Provider dialog box is displayed, as shown in [Figure 5–5](#).

Figure 5–5 The JMS Provider Dialog Box

5. Select any one operation. In this operation select **Oracle Enterprise Messaging Service (OEMS)**, and then select **Database**.
 - **Oracle Enterprise Messaging Service (OEMS):** This allows you to integrate with Oracle's memory, file or database messaging service.
 - **Third Party:** Select this option to integrate with a third party provider.
6. Click **Next**.

The Service Connection dialog box is displayed, as shown in [Figure 5–6](#).

Figure 5–6 Creating a New Database Connection

7. Enter the Java Naming and Directory Interface (JNDI) name in the **Database Server JNDI Name** field. The JNDI name acts as a placeholder for the connection

used when your service is deployed to the BPEL server. This enables you to use different databases for development and later production.

See Also: *Oracle Application Server Adapter Concepts* for understanding JNDI concepts

8. Click **New** to define a database connection.

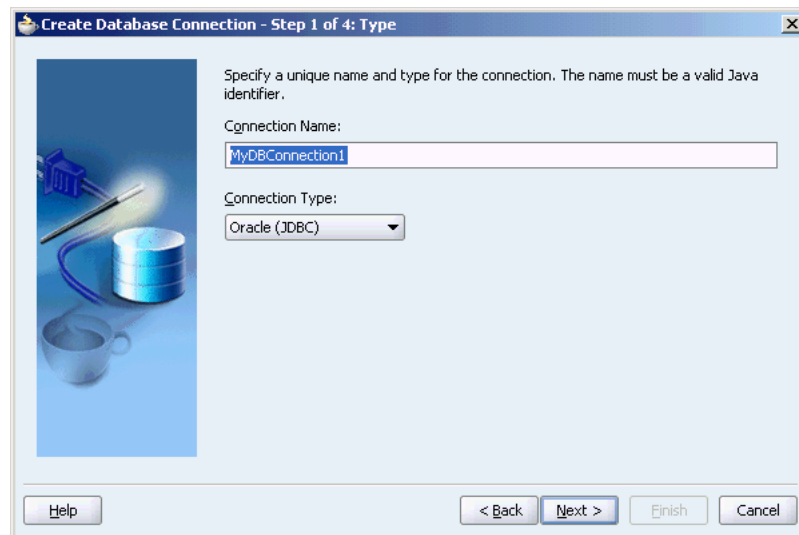
The Create Database Connection Wizard is displayed.

Note: You must connect to the database where Oracle Applications is running.

9. Click **Next**.
10. Enter the following information in the Type dialog box:
 - a. In the **Connection Name** field, specify a unique name for the database connection. In this example, type **MyDBConnection1**.
 - b. From the **Connection Type** box, select the type of connection for your database connection. In this example, retain the default connection type, Oracle (JDBC).

Figure 5–7 shows the Type dialog box.

Figure 5–7 Specifying the Connection Name and Type of Connection

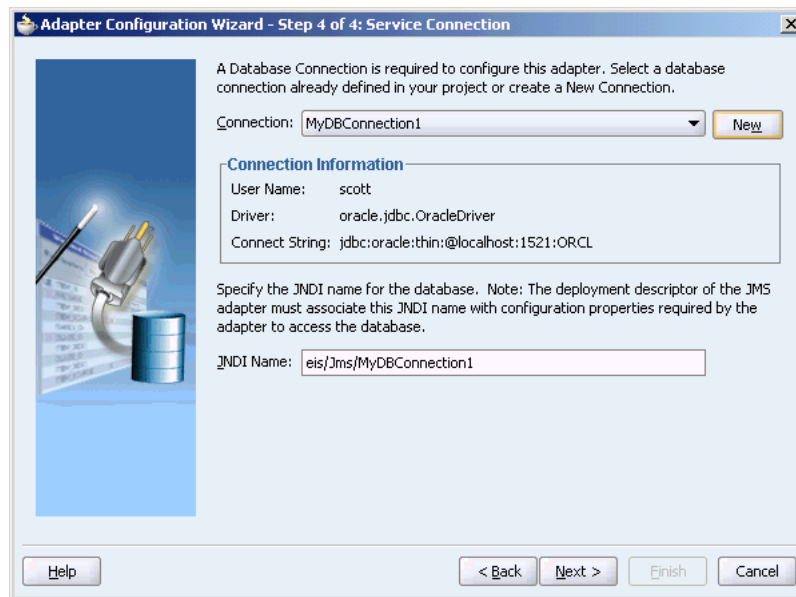


11. Click **Next**. The Authentication dialog box is displayed.
12. Enter information in the following fields:
 - a. In the **UserName** field, specify a unique name for the database connection. In this example, type **scott**.
 - b. In the **Password** field, specify a password for the database connection. In this example, type **tiger**.

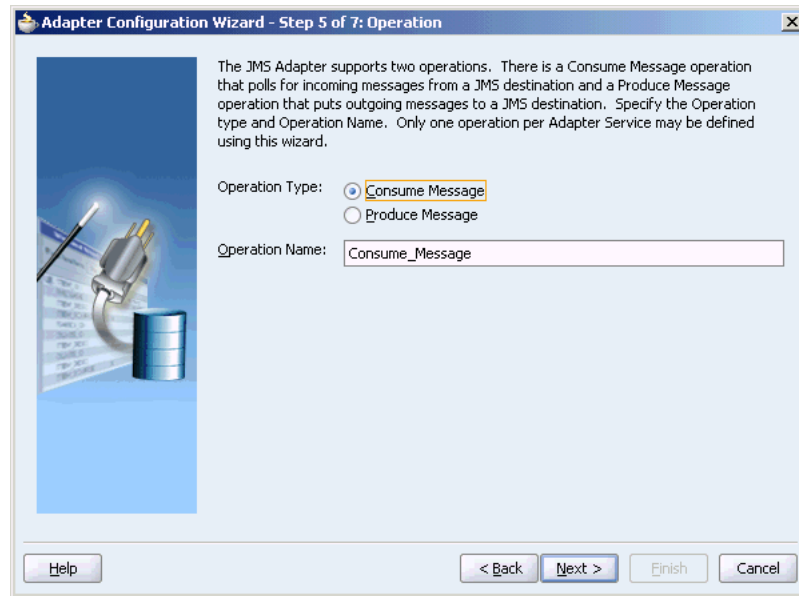
Note: You should have access to SCOTT schema in some database.

13. Click **Next**. The Connection dialog box is displayed.
14. Enter information in the following fields:
 - a. From the **Driver** list, select **Thin**.
 - b. In the **Host Name** field, specify the host name for the database connection.
 - c. In the **JDBC Port** field, specify the port number for the database connection. In this example, enter **1521**.
 - d. In the **SID** field, specify a unique SID value for the database connection. In this example, enter **ORCL**.
15. Click **Next**.
The Test dialog box is displayed.
16. Click **Test Connection** to determine whether the specified information establishes a connection with the database.
17. Click **Finish** to complete the process of creating a new database connection.
The Service Connection dialog box is displayed, providing a summary of the database connection, as shown in [Figure 5–8](#).

Figure 5–8 Service Connection Dialog Box



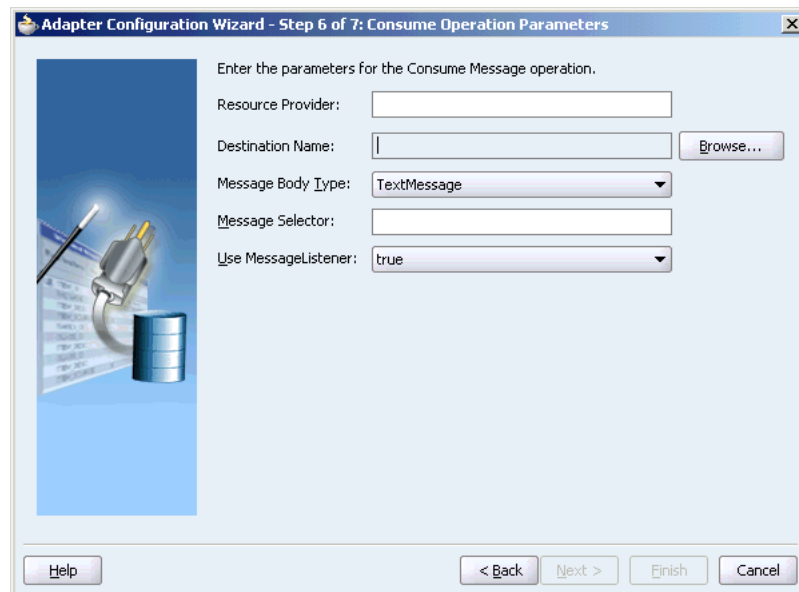
18. Click **Next**.
The Operation dialog box is displayed.
19. Select **Consume Message** or **Produce Message**. In this example, select **Consume Message**.
The operation name is filled in automatically, as shown in [Figure 5–9](#).

Figure 5–9 Operation Window

The **Consume Message** option enables the adapter to consume (receive) inbound messages from a JMS destination.

20. Click Next.

The Consume Operation Parameters dialog box is displayed, as shown in [Figure 5–10](#).

Figure 5–10 The Consume Operation Parameters Dialog Box

21. Enter values for the following fields:

- **Resource Provider**

OC4J provides a ResourceProvider interface to transparently plug in JMS providers. The ResourceProvider interface of OC4J allows EJBs, servlets,

and OC4J clients to access many different JMS providers. The resources are available under `java:comp/resource/`. Oracle JMS is accessed using the `ResourceProvider` interface.

Thus, the resource provider must be entered by the user for OJMS (database option) in the JMS Adapter Wizard

- **Destination Name**

This is the JNDI name of the JMS queue or topic from which to receive the message. This is not an editable field. You must click **Browse** to browse for the queue or topic. The queue or topic to be chosen is based on the type of JMS provider you are using. See the following sections for details:

- [Configuring for OJMS](#)
- [Configuring for OC4J JMS](#)
- [Configuring for TIBCO JMS](#)
- [Configuring for IBM Websphere JMS](#)

- **Message Body Type**

Select either `TextMessage` or `BytesMessage`.

The **StreamMessage** and **MapMessage** message types are not supported in this release.

- **Durable Subscriber ID**

This field is optional. If you are setting up a durable subscriber, then the durable subscriber ID is required. Normally a subscriber loses messages if it becomes disconnected, but a durable subscriber downloads stored messages when it reconnects.

Note: When the JMS provider is memory, file or database messaging service, the Durable Subscriber option will show up only when a topic is selected. However, the Durable Subscriber option always appears when the JMS provider is third party.

- **Message Selector**

This field is also optional. It filters messages based on header and property information. The message selector rule is a Boolean expression. If the expression is **true**, then the message is consumed. If the expression is **false**, then the message is rejected.

For example, you can enter logic, such as:

- **JMSPriority > 3**. Based on this, messages with a priority greater than 3 are consumed; all other messages are rejected.
- **JMSType = 'car' AND color = 'blue' AND weight > 2500**
- **Country in ('UK', 'US', 'France')**

- **Use MessageListener**

This field is set to **true** by default, which means that the server does an asynchronous callback to the adapter. If this is set to **false**, then the adapter performs a synchronous blocking receive.

Note: This example shows a consume message operation. For a produce message operation, this window is different. See [Produce Message Procedure](#) to see how this part of the procedure differs.

After you enter the appropriate parameters, click **Next**.

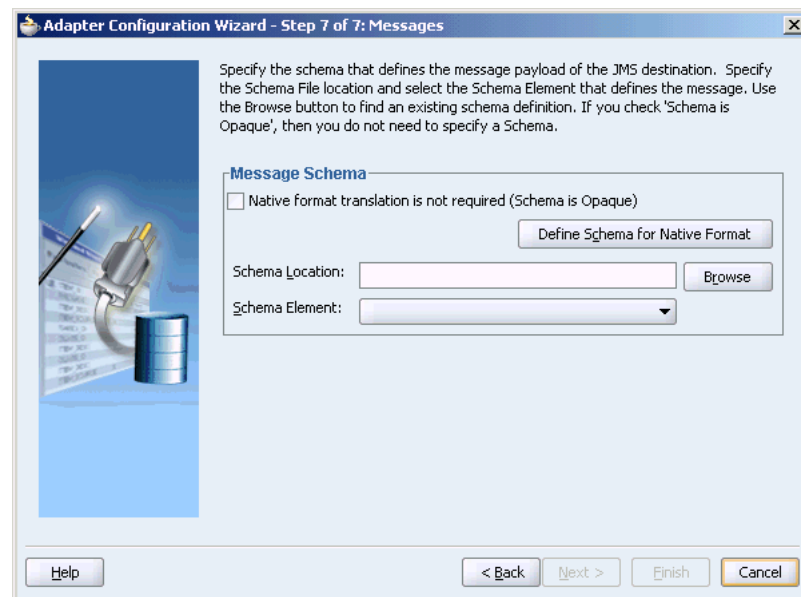
22. The Messages dialog box is displayed, as shown in [Figure 5–11](#). The settings in this dialog box define the correct schema for the message payload.

You can perform one of the following:

- Check **Native format translation is not required (Schema is Opaque)**, which disables the rest of the fields.
- Click **Define Schema for Native Format** to start the Native Format Builder Wizard, which guides you through the process of defining the native format.
- Enter the path for the schema file URL (or browse for the path).

The following steps demonstrate the last option: browsing for the schema file URL.

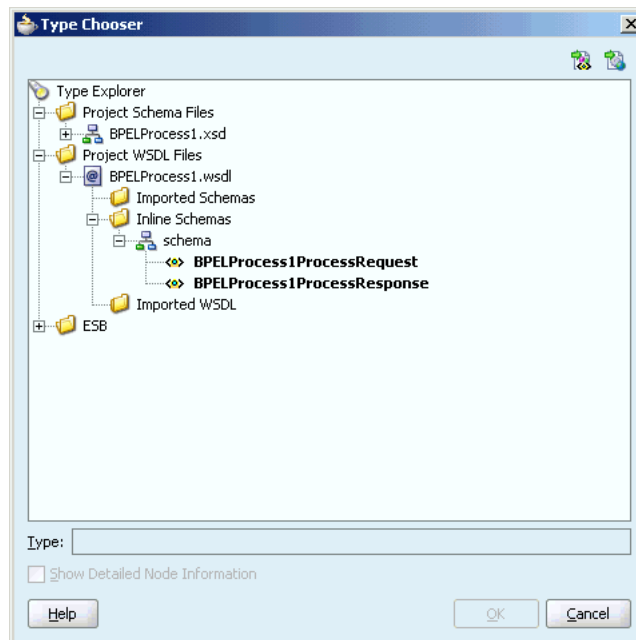
Figure 5–11 Messages Window



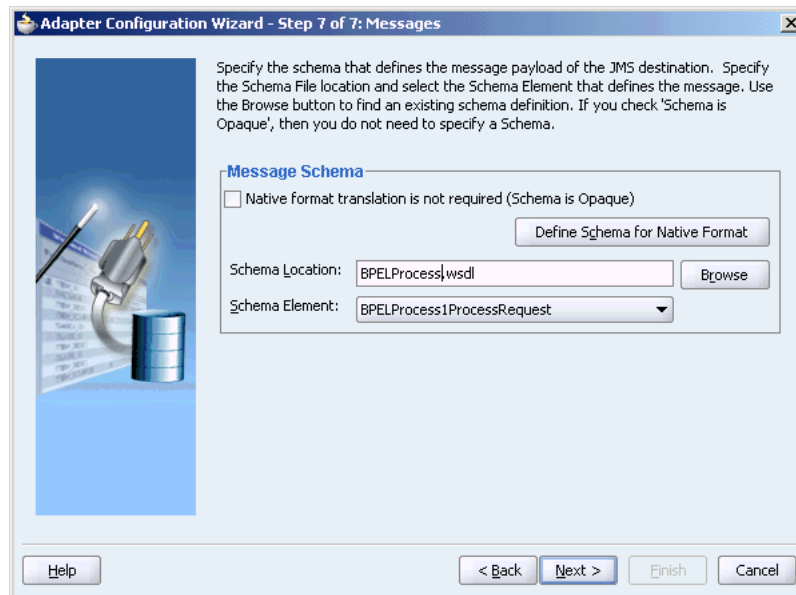
23. Click the **Browse** button.

The **Type Chooser** window is displayed, with the **Type Explorer** navigation tree, as shown in [Figure 5–12](#).

24. Browse the tree and select the appropriate schema type, and then click **OK**.

Figure 5–12 Selecting a Schema from the Type Chooser Window

The Messages window is displayed again, this time with the **Schema File URL** field and the **Schema Element** field filled up, as shown in Figure 5–13.

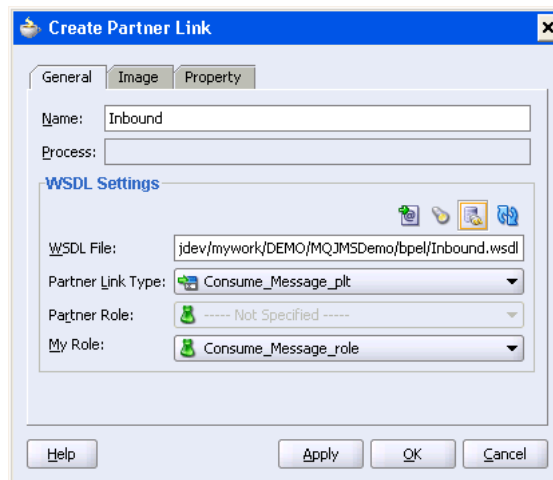
Figure 5–13 Completed Messages Window

25. Click Next.

The **Finish** window is displayed. This box shows the path and name of the adapter file that the wizard creates.

26. Click Finish.

The Create Partner Link window is displayed with the fields populated, as shown in Figure 5–14.

Figure 5–14 Completed Create Partner Link Window

27. Click OK.

5.2.1.3 Generated WSDL File

The following WSDL file is generated by the Adapter Configuration Wizard:

```
<definitions
  name="JMS_Example"
  targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/jms/JMS_Example/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:tns="http://xmlns.oracle.com/pcbpel/adapter/jms/JMS_Example/"
  xmlns:plt="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
  xmlns:jca="http://xmlns.oracle.com/pcbpel/wsdl/jca/"
  xmlns:opaque="http://xmlns.oracle.com/pcbpel/adapter/opaque/"
  xmlns:pc="http://xmlns.oracle.com/pcbpel/"
  xmlns:hdr="http://xmlns.oracle.com/pcbpel/adapter/jms/"
>
```

This code segment defines the name of the adapter, and the locations of various necessary schemas and other definition files.

```
<import namespace="http://xmlns.oracle.com/pcbpel/adapter/jms/"
location="jmsAdapterInboundHeader.wsdl"/>
```

This code segment imports the necessary namespace.

```
<types>
<schema targetNamespace="http://xmlns.oracle.com/pcbpel/adapter/opaque/"
  xmlns="http://www.w3.org/2001/XMLSchema" >
  <element name="opaqueElement" type="base64Binary" />
</schema>
</types>
<message name="Consume_Message_msg">
  <part name="opaque" element="opaque:opaqueElement" />
</message>
<portType name="Consume_Message_ptt">
  <operation name="Consume_Message">
    <input message="tns:Consume_Message_msg" />
  </operation>
</portType>
```

This code segment defines the message type, name, and the port type for the partner link.

```
<binding name="Consume_Message_binding" type="tns:Consume_Message_ptt">
<pc:inbound_binding />
  <operation name="Consume_Message">
    <jca:operation
      ActivationSpec="oracle.tip.adapter.jms.inbound.JmsConsumeActivationSpec"
      DestinationName="jms/DemoQue"
      UseMessageListener="true"
      PayloadType="TextMessage"
      OpaqueSchema="true" />
    </jca:operation>
  <input>
    <jca:header message="hdr:InboundHeader_msg" part="inboundHeader" />
  </input>
</operation>
</binding>
```

This code segment defines the necessary bindings for the consume message operation, the target queue, and identifies the message header.

```
<service name="JMS_Example2">
  <port name="Consume_Message_pt" binding="tns:Consume_Message_binding">
    <jca:address location="eis/Jms/topics.xml" />
  </port>
</service>
<plt:partnerLinkType name="Consume_Message_plt" >
  <plt:role name="Consume_Message_role" >
    <plt:portType name="tns:Consume_Message_ptt" />
  </plt:role>
</plt:partnerLinkType>
</definitions>
```

This last part defines the database connection, the connection factory (as defined in the `oc4j-ra.xml` file), and the name and role of the `partnerLinkType` and `portType`.

5.2.1.4 oc4j-ra.xml file

The `oc4j-ra.xml` file defines the endpoints for the JMS connection factories. The connection factories include configuration properties for each endpoint. Endpoints are added to accommodate different types of connections, as demonstrated in the following sections. The following example is from the generic `oc4j-ra.xml` file:

```
<?xml version="1.0"?>
<!DOCTYPE oc4j-connector-factories PUBLIC "-//Oracle//DTD Oracle Connector
  9.04//EN" "http://xmlns.oracle.com/ias/dtds/oc4j-connector-factories-9_04.dtd">
<oc4j-connector-factories>
  <connector-factory location="eis/MyJmsTopic1" connector-name="Jms Adapter">
    <config-property name="connectionFactoryLocation"
      value="jms/TopicConnectionFactory" />
    <config-property name="factoryProperties" value="" />
    <config-property name="acknowledgeMode" value="AUTO_ACKNOWLEDGE" />
    <config-property name="isTopic" value="true" />
    <config-property name="isTransacted" value="true" />
    <config-property name="username" value="admin" />
    <config-property name="password" value="welcome" />
  </connector-factory>
  <connector-factory location="eis/MyJmsTopic2" connector-name="Jms Adapter">
    <config-property name="connectionFactoryLocation"
```



```
...
</oc4j-connector-factories>
```

5.2.1.5 Produce Message Procedure

A produce message operation has certain differences in the definition procedure, particularly in Step 19 of [Using the Adapter Configuration Wizard to Configure a JMS Adapter](#). Instead of specifying consume operation parameters, you specify the following produce operation parameters. This enables the adapter to produce (send) outbound messages for a JMS destination. The Produce Operation Parameters window is shown in [Figure 5-15](#).

- **Destination Name:**

The JNDI name of the JMS queue or topic to which to deliver the message. The name to enter is based on the type of JMS provider you are using. See the following sections for detail:

- [Configuring for OJMS](#)
- [Configuring for OC4J JMS](#)
- [Configuring for TIBCO JMS](#)
- [Configuring for IBM Websphere JMS](#)

- **Message Body Type:**

The supported values are **TextMessage** or **BytesMessage**. The **StreamMessage** and **MapMessage** message types are not supported in this release.

- **Delivery Mode:**

The values are **Persistent** or **Nonpersistent**. A persistent delivery mode specifies a persistent JMS publisher; that is, a publisher that stores messages for later use by a durable subscriber. A durable subscriber is a consume message with a durable subscriber ID in the corresponding field in step 21 of [Using the Adapter Configuration Wizard to Configure a JMS Adapter](#). A nondurable subscriber loses any messages that are produced when the adapter is not active. A durable subscriber downloads messages that have been stored in the persistent publisher, and therefore does not have to remain active at all times to receive all the messages.

- **Priority:**

Select a priority value, with **9** representing the highest priority and **0** representing the lowest priority. The default is **4**.

- **TimeToLive:**

The amount of time before the message expires and is no longer available to be consumed.

Figure 5–15 Produce Operation Parameters Window

Adapter Configuration Wizard - Step 6 of 7: Produce Operation Parameters

Enter the parameters for the Produce Message operation.

Resource Provider:

Destination Name:

Message Body Type:

Delivery Mode:

Priority:

TimeToLive:

5.2.1.6 Configuring for OJMS

Configure the OJMS provider within the `resource-provider` element in the `global application.xml` file. You can configure the resource provider with a URL property. The following demonstrates a URL configuration:

```
<resource-provider class="oracle.jms.OjmsContext" name="ojmsdemo">
  <description>OJMS/AQ</description>
  <property name="url" value="jdbc:oracle:thin:@localhost:1521:my" />
  <property name="username" value="jmsuser" />
  <property name="password" value="jmsuser" />
</resource-provider>
```

In the `oc4j-ra.xml` file, add the following code segments:

```
<connector-factory location="eis/aqjms/Topic" connector-name="Jms Adapter">
  <config-property name="connectionFactoryLocation"
    value="java:comp/resource/ojmsdemo/TopicConnectionFactory/myTCF" />
  <config-property name="factoryProperties" value="" />
  <config-property name="acknowledgeMode" value="AUTO_ACKNOWLEDGE" />
  <config-property name="isTopic" value="true" />
  <config-property name="isTransacted" value="true" />
  <config-property name="username" value="jmsuser" />
  <config-property name="password" value="jmsuser" />
</connector-factory>
<connector-factory location="eis/aqjms/Queue" connector-name="Jms Adapter">
  <config-property name="connectionFactoryLocation" value="
    java:comp/resource/ojmsdemo/QueueConnectionFactory/myQCF" />
  <config-property name="factoryProperties" value="" />
  <config-property name="acknowledgeMode" value="AUTO_ACKNOWLEDGE" />
  <config-property name="isTopic" value="false" />
  <config-property name="isTransacted" value="true" />
  <config-property name="username" value="jmsuser" />
  <config-property name="password" value="jmsuser" />
</connector-factory>
```

In this case, correct JMS connection JNDI names for Step 7 of [Using the Adapter Configuration Wizard to Configure a JMS Adapter](#) are `eis/aqjms/Topic` or `eis/aqjms/Queue`.

Set the `isTransacted` value in the `oc4j-ra.xml` file to `true`. Setting it to `false` leads to exception errors.

Access the OJMS Resources

```
java:comp/resource + JMS_provider_name + TopicConnectionFactories + user_defined_name
```

```
java:comp/resource + JMS_provider_name + QueueConnectionFactories + user_defined_name
```

The OJMS syntax for the connection factory is as follows:

or

The `user_defined_name` can be anything and does not match any other configuration. The `ConnectionFactories` details what type of factory is being defined. For this example, the JMS provider name is defined in the `resource-provider` element as `ojmsdemo`.

For a queue connection factory: Because the JMS provider name is `ojmsdemo` and you decide to use a name of `myQCF`, the connection factory name is

```
java:comp/resource/ojmsdemo/QueueConnectionFactories/myQCF.
```

For a topic connection factory: Because the JMS provider name is `ojmsdemo` and you decide to use a name of `myTCF`, the connection factory name is

```
java:comp/resource/ojmsdemo/TopicConnectionFactories/myTCF.
```

The user-defined names of `myQCF` and `myTCF` are not used for anything else in your logic. Therefore, you can choose any name.

Destination

The OJMS syntax for any destination is as follows:

```
java:comp/resource + JMS_provider_name + Topics + Destination_name
```

or

```
java:comp/resource + JMS_provider_name + Queues + Destination_name
```

The topic or queue details which type of destination is being defined. The destination name is the actual queue or topic name defined in the database.

For this example, the JMS provider name is defined in the `resource-provider` element as `ojmsdemo`. In the database, the queue name is `aqQueue`.

For a queue: If the JMS provider name is `ojmsdemo` and the queue name is `aqQueue`, the JNDI name for the queue is

```
java:comp/resource/ojmsdemo/Queues/aqQueue.
```

For a topic: If the JMS provider name is `ojmsdemo` and the topic name is `aqTopic`, the JNDI name for the topic is

```
java:comp/resource/ojmsdemo/Topics/aqTopic.
```

OJMS and Remote Databases

To configure the adapter to use a remote database, the entries in the `application.xml` file must look as follows:

```
<resource-provider class="oracle.jms.OjmsContext" name="ojmsdemo">
<description>OJMS/AQ</description>
<property name="url"
value="jdbc:oracle:thin:@remote-host:remote-port:remote-sid" />
<property name="username" value="jmsuser" />
@ <property name="password" value="jmsuser" />
</resource-provider>
```

5.2.1.7 Configuring for OC4J JMS

If the OC4J JMS server is running on another remote host, you can configure the JMS adapter to talk to the server by using the following connector entry. Note that the only difference with this connector entry is in the factory properties. The factory properties can establish a JNDI context for the adapter. Substitute [hostname] with the hostname of the server on which the OC4J JMS server is running. If the RMI port of the remote OC4J instance is not the default value (23791), you must specify the RMI port in the provider URL (that is, `ormi://remotehost.domain.com:23795`).

```
<connector-factory location="eis/RemoteOC4JJMS/Queue"
connector-name="Jms Adapter">
    <config-property name="connectionFactoryLocation"
        value="jms/QueueConnectionFactory" />
    <config-property name="factoryProperties"
value="java.naming.factory.initial=com.evermind.server.ApplicationClientInitialCon
textFactory;java.naming.provider.url=ormi://[hostname];
java.naming.security.principal=admin;java.naming.security.credentials=welcome"/>
    <config-property name="acknowledgeMode" value="AUTO_ACKNOWLEDGE" />
    <config-property name="isTopic" value="false" />
    <config-property name="isTransacted" value="true" />
    <config-property name="username" value="admin" />
    <config-property name="password" value="welcome" />
</connector-factory>
```

In this case, the correct JMS connection JNDI name for Step 7 of [Using the Adapter Configuration Wizard to Configure a JMS Adapter](#) is `eis/RemoteOC4JJMS/Queue`.

In addition, you must have the file `META-INF/application-client.xml` in the classpath. The contents of the file can be the following:

```
<application-client/>
```

To put this file in the classpath, put the specified contents into the file at `Oracle_Home\bpel\system\classes\META-INF\application-client.xml` and restart Oracle BPEL Server.

If you set `isTransacted` to `true` in the `oc4j-ra.xml` file for an outbound connection, you receive an error. Do not set this value to `true` for outbound connections.

Destination Name

The destination name for OC4J JMS is either a JNDI location (for example, `jms/demoQueue` or `jms/demoTopic`) or the actual name of the destination as configured in `jms.xml` (for example, `Demo Queue` or `Demo Topic`).

5.2.1.8 Configuring for TIBCO JMS

The BPEL OC4J `application.xml` file should contain the following jar file, where Tibco EMS is installed in `C:\tibco\ems`. The JMS and JNDI jar files are already present in the classpath and do not need to be included.

```
<library path="C:\tibco\ems\clients\java\tibjms.jar" />
```

Create the appropriate endpoints for the JMS connection factories in `oc4j-ra.xml`. After this change is made, restart Oracle BPEL Server. Here are the appropriate code segments. You can modify the necessary parameters and use this for your purpose:

```
<connector-factory location="eis/tibjms/Topic" connector-name="Jms Adapter">
  <config-property name="connectionFactoryLocation"
    value="TopicConnectionFactory" />
  <config-property name="factoryProperties"
value="java.naming.factory.initial=com.tibco.tibjms.naming.TibjmsInitialContextFa
ctory;java.naming.provider.url=tibjmsnaming://localhost:7222;java.naming.security.
principal=admin;java.naming.security.credentials=password" />
    <config-property name="acknowledgeMode" value="AUTO_ACKNOWLEDGE" />
    <config-property name="isTopic" value="true" />
    <config-property name="isTransacted" value="true" />
    <config-property name="username" value="admin" />
    <config-property name="password" value="password" />
</connector-factory>
<connector-factory location="eis/tibjms/Queue" connector-name="Jms Adapter">
  <config-property name="connectionFactoryLocation"
    value="QueueConnectionFactory" />
  <config-property name="factoryProperties"
value="java.naming.factory.initial=com.tibco.tibjms.naming.TibjmsInitialContextFa
ctory;java.naming.provider.url=tibjmsnaming://localhost:7222;java.naming.security.
principal=admin;java.naming.security.credentials=password" />
    <config-property name="acknowledgeMode" value="AUTO_ACKNOWLEDGE" />
    <config-property name="isTopic" value="false" />
    <config-property name="isTransacted" value="true" />
    <config-property name="username" value="admin" />
    <config-property name="password" value="password" />
</connector-factory>
```

In this case, correct JMS connection JNDI names for Step 7 of [Using the Adapter Configuration Wizard to Configure a JMS Adapter](#) are `eis/tibjms/Topic` or `eis/tibjms/Queue`.

When using Tibco JMS, always set the `ClientID` property as follows in the `Oracle_Home\bpel\system\appserver\oc4j\j2ee\home\application-deployment\default\FtpAdapter\oc4j-ra.xml` file:

```
<config-property name="factoryProperties"
  value="ClientID=clientId{time}"/>
```

The substring `{time}` instructs the run time to replace it with the value of `Java System.currentTimeMillis()`. The other supported substitutions are:

- `{checksum}`
A checksum based on the values of the `oc4j-ra.xml` connection factory properties (referenced through `jca:address`).
- `{sequence}`
Next member of an increasing series of integers starting at zero.

These settings enable you to specify a fixed or a variable `ClientID` in `oc4j-ra.xml`.

5.2.1.8.1 Direct Connection

Direct connection can also be defined instead of the JNDI connection. A direct connection is necessary for the Solaris middle tier. Use the following direct connection entry in the `oc4j-ra.xml` file, instead of the JNDI entry.

5.2.1.9 Configuring for IBM Websphere JMS

The BPEL OC4J `application.xml` file should contain the following jar files, assuming MQ Series is installed in the `C:\mqseries` directory.

```
<library path="C:\mqseries\java\lib\com.ibm.mq.jar" />
<library path="C:\mqseries\java\lib\com.ibm.mqjms.jar" />
```

Create the appropriate endpoints for the JMS connection factories in `oc4j-ra.xml`. After this change is made, you must restart Oracle BPEL Server. Here are the appropriate code segments. You can modify the necessary parameters and use this for your purpose.

```
<connector-factory location="eis/mqseries/Queue" connector-name="Jms Adapter">
  <config-property name="connectionFactoryLocation"
    value="com.ibm.mq.jms.MQQueueConnectionFactory" />
  <config-property name="factoryProperties"
value="QueueManager=my.queue.manager;TransportType=1;HostName=myhost.com;Port=141
4;Channel=MYCHANNEL" />
  <config-property name="acknowledgeMode" value="AUTO_ACKNOWLEDGE" />
  <config-property name="isTopic" value="false" />
  <config-property name="isTransacted" value="true" />
  <config-property name="username" value="MUSR_MQADMIN" />
  <config-property name="password" value="password" />
</connector-factory>
```

In this case, the correct JMS connection JNDI name for Step 7 of [Using the Adapter Configuration Wizard to Configure a JMS Adapter](#) is `eis/mqseries/Queue`.

Note: WebSphere MQ uses the regular Windows username and group authorizations to protect WebSphere MQ applications and WebSphere MQ Administration.

To work with username and group authorizations:

- Open the **Control Panel**, then **User Accounts** or **Users** and then **Passwords**.
- On Windows NT, open the **User Manager**.

WebSphere MQ Installation automatically creates the local group `mqgm` and username `MUSR_MQADMIN`. Usernames which are members of this group have authority to work with and administer WebSphere MQ queue managers. You can change the password for the user `MUSR_MQADMIN`, by selecting it and then clicking on **Set password**. This is the user name and password that should reflect in the `oc4j-ra.xml` entry, as shown in the following example:

```
<config-property name="username" value="MUSR_MQADMIN"/>
<config-property name="password" value="password"/>
```

Destination Name

The destination name is the name of the topic or queue listed in your MQ Series configuration. For example, the name of the queue can be `queue:///MYQUEUE?targetClient=0`.

5.2.2 Case Two: MQSeries Queue Integration Through the OracleAS Adapter for JMS

The OracleAS Adapter for JMS is part of the Oracle BPEL Process Manager install and is a JCA 1.5 Resource Adapter. The Adapter Framework (AF) is used for the bidirectional integration of the JCA 1.5 resource adapters with BPEL Process Manager. Adapter FW is based on open standards and employs the Web Service Invocation Framework (WSIF) technology for exposing the underlying JCA Interactions as Web Services. This example showcases the following:

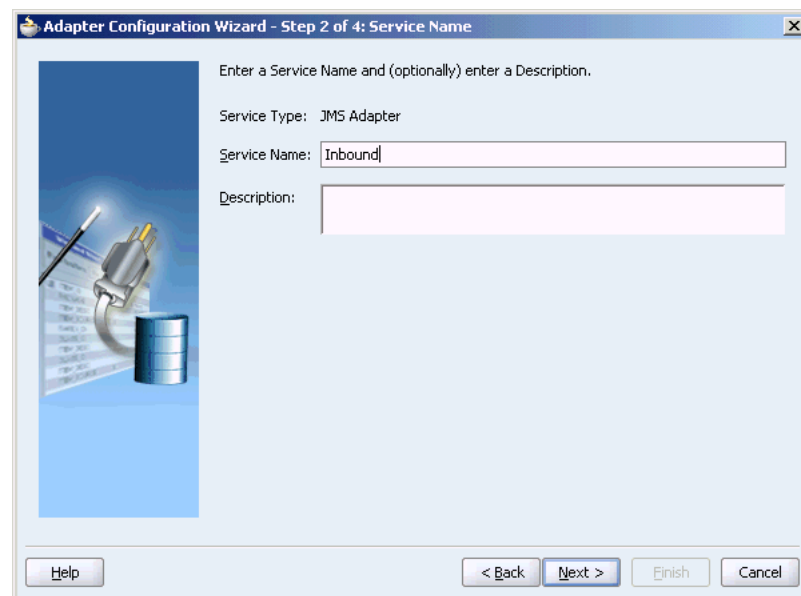
- A MQSeries Queue Consumer dequeuing the message from a MQSeries queue and triggering the BPEL process. The message payload is of type *Text* and is defined by a delimited schema using the Native Format Builder.
- A MQSeries Queue Producer enqueueing the message to a MQSeries Queue based on the invoke BPEL message.

5.2.2.1 Configuring the MQSeries Consumer Service

This section describes how to create an adapter service for a partner link.

1. Follow steps 1 through 3 under [Using the Adapter Configuration Wizard to Configure a JMS Adapter](#).
2. In Step 4 of [Using the Adapter Configuration Wizard to Configure a JMS Adapter](#), for service name, type *Inbound*, as shown in [Figure 5–16](#).

Figure 5–16 The service Name Dialog Box



3. Select **Third Party** as the operation type to integrate with a third party, and then click Next.

The JMS Connection dialog box is displayed.

4. Specify the following connection information in the JMS Connection dialog box.
 - a. **Factory Properties:**

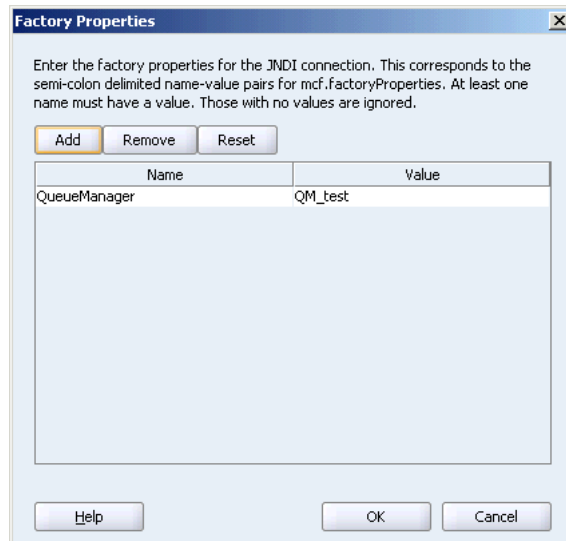
Connection factory location (`mcf.ConnectionFactoryLocation`) tells the JMS adapter where or how to instantiate either a Queue or a Topic Connection Factory in the underlying JMS provider. First the JMS adapter will try to locate the specified value through JNDI. If that fails, then it considers the value the

name of a Java class, which it will then try to instantiate. Only in this case it will also look in the Factory Properties value in `mcf.FactoryProperties` for any name or value pairs which suggest vendor specific bean methods on the Connection Factory.

This is a required field, and it is non-editable. Click **Edit** to enter the factory properties for the JNDI connection, and then click **OK**.

Figure 5–17 shows the Factory Properties dialog box with the factory properties filled up.

Figure 5–17 Specifying the Factory Properties



b. JMS Connection Factory:

JMS Connection Factory is used to create JMS Connections. In this example `MQQueueConnectionFactory` is used to create `QueueConnections`.

In this example, enter the following value in the JMS Connection Factory field:

```
com.ibm.mq.jms.MQQueueConnectionFactory
```

c. Transacted:

This is a boolean that determines whether the session is transacted or non-transacted.

In this example, retain the default value, `true`.

d. Destination Type:

This determines whether destination is a Queue or a Topic.

In this example, retain the default value, `Queue`.

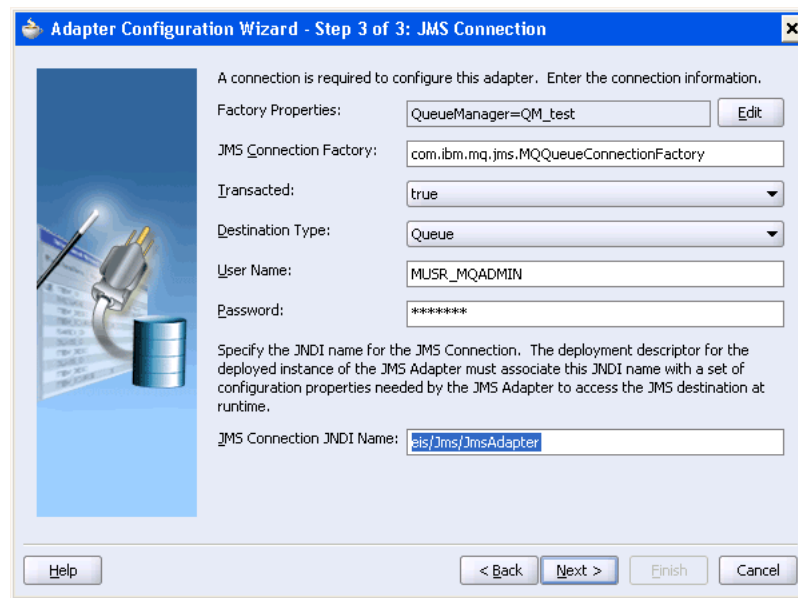
e. User Name:

Enter a user name.

f. Password:

Enter a password.

Figure 5–18 shows the JMS Connection dialog box with all the field populated.

Figure 5–18 The JMS Connection Dialog Box


Adapter Configuration Wizard - Step 3 of 3: JMS Connection

A connection is required to configure this adapter. Enter the connection information.

Factory Properties:

JMS Connection Factory:

Transacted:

Destination Type:

User Name:

Password:

Specify the JNDI name for the JMS Connection. The deployment descriptor for the deployed instance of the JMS Adapter must associate this JNDI name with a set of configuration properties needed by the JMS Adapter to access the JMS destination at runtime.

JMS Connection JNDI Name:

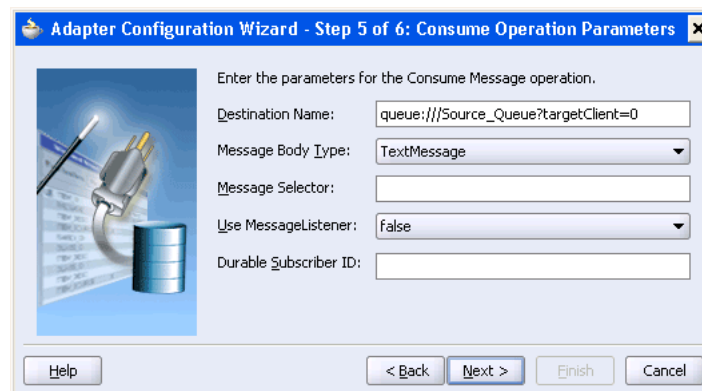
5. Click **Next**.

The Operation dialog box is displayed.

6. Select **Consume**, and then click **Next**.

7. Enter the values in the Consume Operation Parameters dialog box, as shown in [Figure 5–19](#), and then click **Next**.

The **Destination Name** field refers to the MQSeries queue name. The value `targetClient=0` in the Destination Name field is used indicates that the JMS Header will be part of the message. You have to specify the `targetClient` value of 1 if you want the JMS Headers to be truncated from the message. The JMS Adapter supports Text and Byte messages. This example uses the Text Message format.

Figure 5–19 The Consume Operation Parameters Dialog Box


Adapter Configuration Wizard - Step 5 of 6: Consume Operation Parameters

Enter the parameters for the Consume Message operation.

Destination Name:

Message Body Type:

Message Selector:

Use MessageListener:

Durable Subscriber ID:

8. In the Messages dialog box, click **Browse** to select a URL for the schema file. In this example, select `expense.xsd`, as shown in [Figure 5–20](#).

One of the key features of the JMS Adapter is the support for native format translation into XML and vice-versa. The Adapter payload (Text or Byte) can be

pointed to a delimited, fixed-positional, Cobol copy book data and is converted to XML and back by the Native Format Translator. This example points to a delimited schema for the incoming Text Message.

Figure 5–20 The Messages Dialog Box

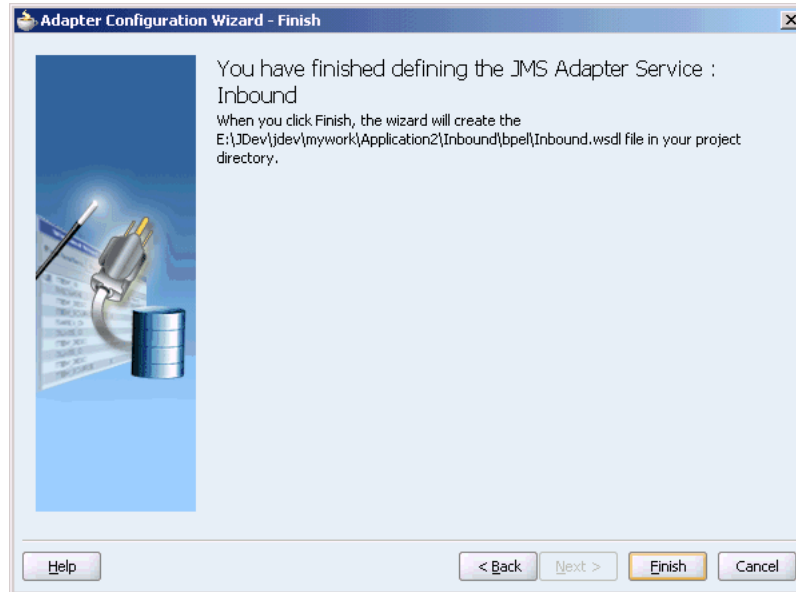


The Adapter Configuration Wizard-Finish screen is displayed.

9. Click **Next**, and then click **Finish** to complete configuring the MQSeries consumer service.

The Finish screen is displayed, as shown in [Figure 5–21](#).

Figure 5–21 Finish Screen of JMS Adapter Service: Inbound



10. Click **Ok** in the Create Partner Link dialog box to complete the creation of partner link.

5.2.2.2 Configuring the MQSeries Producer Service

The next step after configuring the MQSeries Consumer Service is to configure the MQSeries Produce Service.

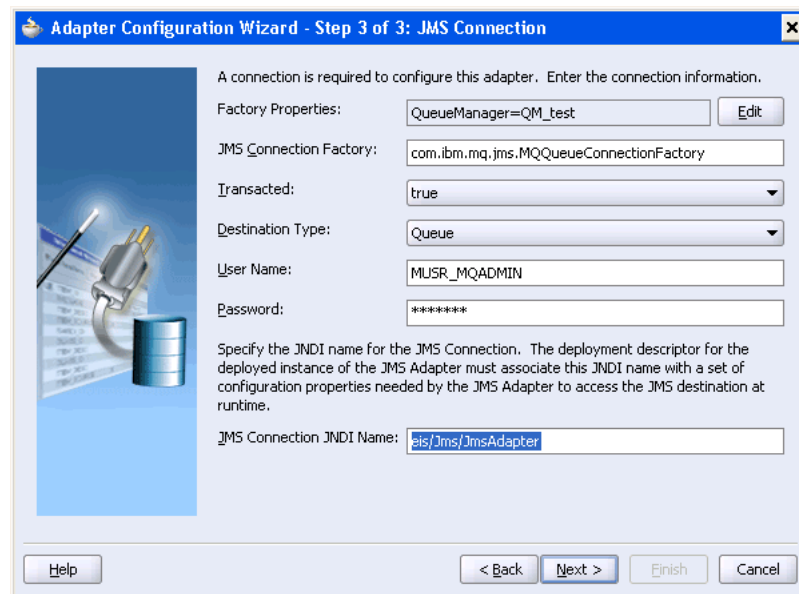
Use the following steps to configure the MQSeries Produce Service:

1. Follow steps 1 through 3 under [Using the Adapter Configuration Wizard to Configure a JMS Adapter](#).
2. In Step 4 of [Using the Adapter Configuration Wizard to Configure a JMS Adapter](#), for service name, type Outbound.
3. Select **Third Party** as the operation type to integrate with a third party, and then click **Next**.

The JMS Connection dialog box is displayed.

4. Specify the connection information in the JMS Connection dialog box, as shown in [Figure 5–22](#).

Figure 5–22 The JMS Connection Dialog Box



Adapter Configuration Wizard - Step 3 of 3: JMS Connection

A connection is required to configure this adapter. Enter the connection information.

Factory Properties:

JMS Connection Factory:

Transacted:

Destination Type:

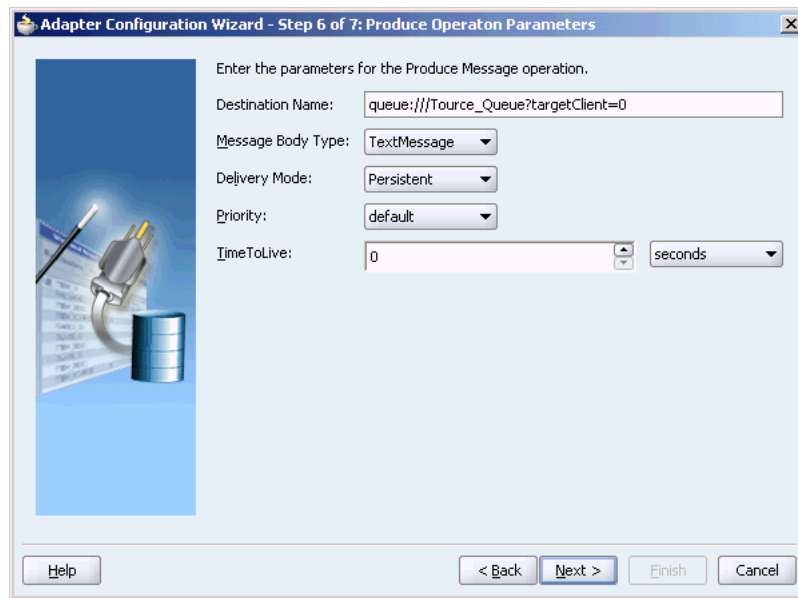
User Name:

Password:

Specify the JNDI name for the JMS Connection. The deployment descriptor for the deployed instance of the JMS Adapter must associate this JNDI name with a set of configuration properties needed by the JMS Adapter to access the JMS destination at runtime.

JMS Connection JNDI Name:

5. Click **Produce** in the Operation dialog box, and then click **Next**.
6. Select **Produce**, and then click **Next**.
7. Enter values in the Consume Operation Parameters dialog box, as shown in [Figure 5–23](#), and then click **Next**.

Figure 5–23 The Produce Operation Parameters Dialog Box


Adapter Configuration Wizard - Step 6 of 7: Produce Operation Parameters

Enter the parameters for the Produce Message operation.

Destination Name:

Message Body Type:

Delivery Mode:

Priority:

TimeToLive:

Buttons: Help, < Back, Next >, Finish, Cancel

8. In the Messages dialog box, click **Browse** to select a URL for the schema file. In this example, select `expense.xsd`, as shown in [Figure 5–24](#).

Figure 5–24 The Messages Dialog Box


Adapter Configuration Wizard - Step 6 of 6: Messages

Specify the schema that defines the message payload of the JMS destination. Specify the Schema File location and select the Schema Element that defines the message. Use the Browse button to find an existing schema definition. If you check 'Schema is Opaque', then you do not need to specify a Schema.

Message Schema

☐ Native format translation is not required (Schema is Opaque)

Define Schema for Native Format

Schema Location:

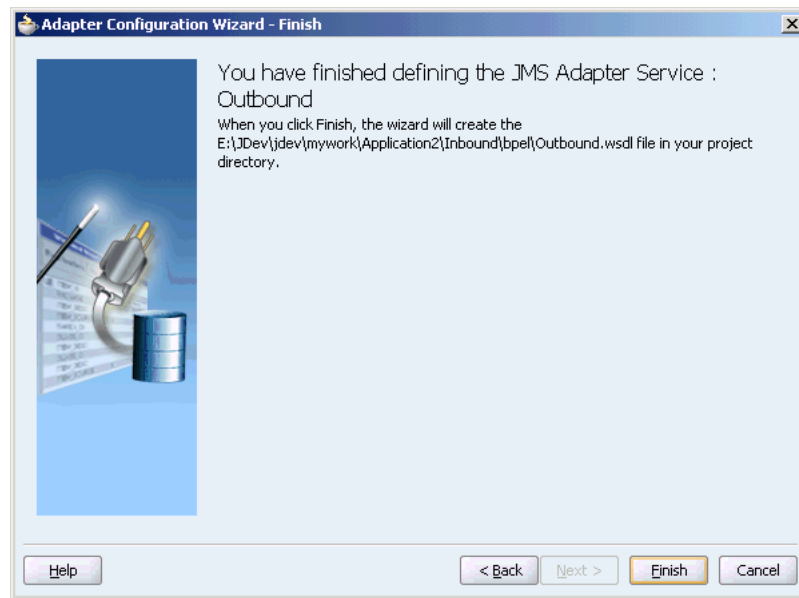
Schema Element:

Buttons: Help, < Back, Next >, Finish, Cancel

The Adapter Configuration Wizard-Finish screen is displayed.

9. Click **Next**, and then click **Finish** to complete configuring the MQSeries producer service.

The Finish screen is displayed, as shown in [Figure 5–25](#).

Figure 5–25 Finish Screen for JMS Adapter Service: Outbound

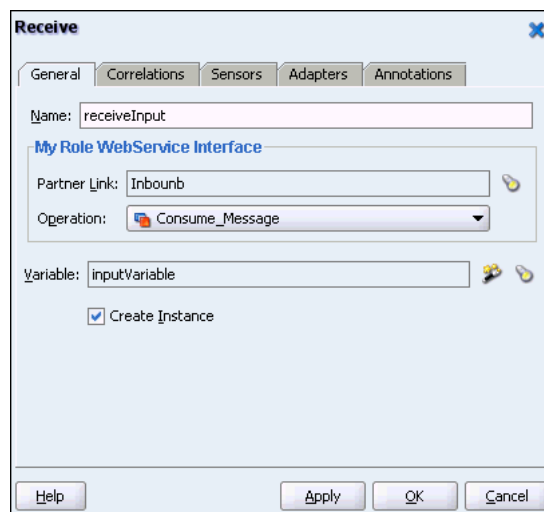
10. Click **Ok** in the Create Partner Link dialog box to complete the creation of partner link.

5.2.2.3 Configuring an End-to-End BPEL Process

The following are the steps to configure an end-to-end BPEL process:

1. Click **Receive** to configure a receive activity to point to the MQSeries Consumer Service.

Figure 5–26 shows the Receive activity.

Figure 5–26 Configuring a Receive Activity

2. Select the `create Instance` option, and create a global variable, `InputVariable`, as shown in Figure 5–26 to receive the incoming message from the MQSeries Consumer.
3. Click **Ok**.

4. Drag and drop an invoke activity to point to the MQSeries Producer Partner Link.

Figure 5–27 shows the invoke activity.

Figure 5–27 Configuring an Invoke Activity

The 'Invoke' dialog box is shown with the 'General' tab selected. It features a 'Name' field with the value 'Invoke_1'. Below this is a section titled 'Partner Role Web Service Interface' containing a 'Partner Link' dropdown set to 'Outbound' and an 'Operation' dropdown set to 'Produce_Message'. There are also fields for 'Input Variable' and 'Output Variable', each with a small icon to its right. At the bottom are buttons for 'Help', 'Apply', 'OK', and 'Cancel'. A warning icon and 'Errors: 3' are visible at the top left of the dialog.

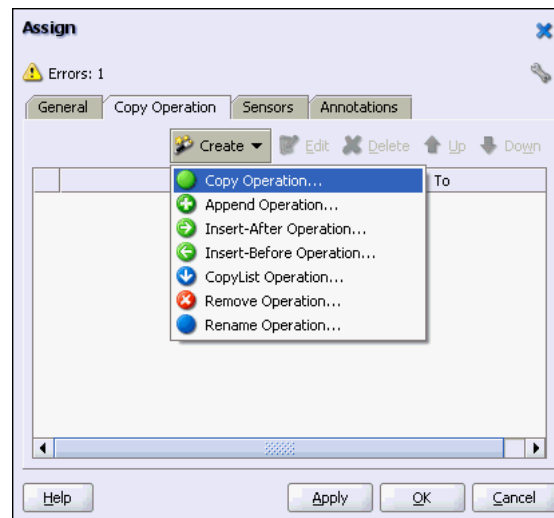
5. Drag and drop an assign activity to set the MQSeries Header and Payload message for the MQSeries Producer, and then click **OK**

Figure 5–28 shows the Assign activity.

Figure 5–28 Configuring an Assign Activity

The 'Assign' dialog box is shown with the 'General' tab selected. It has a 'Create' dropdown menu, and buttons for 'Edit', 'Delete', 'Up', and 'Down'. Below these is a table with two columns: 'From' and 'To'. The table is currently empty. At the bottom are buttons for 'Help', 'Apply', 'OK', and 'Cancel'. A warning icon and 'Errors: 1' are visible at the top left of the dialog.

6. Click the **Create** drop-down list, and then select **Copy Operation**, as shown in Figure 5–29.

Figure 5–29 Selecting the Copy Operation

The Create Copy Operation dialog box is displayed.

7. Perform copy operations, as shown in [Figure 5–30](#) and [Figure 5–31](#).

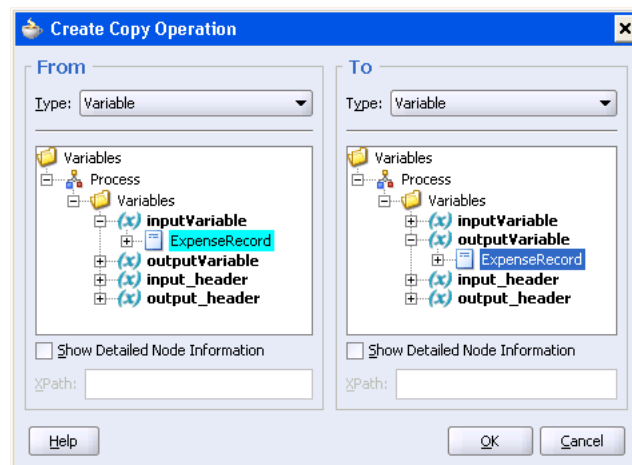
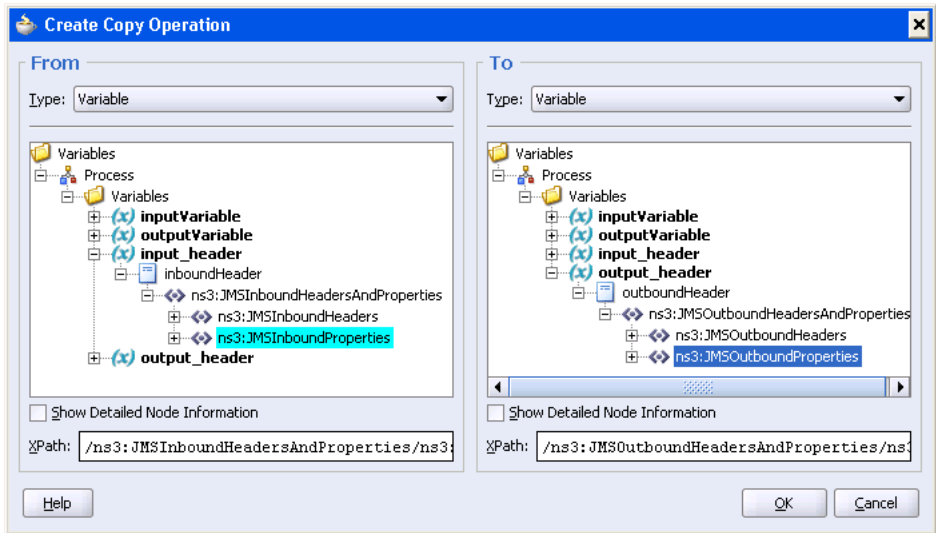
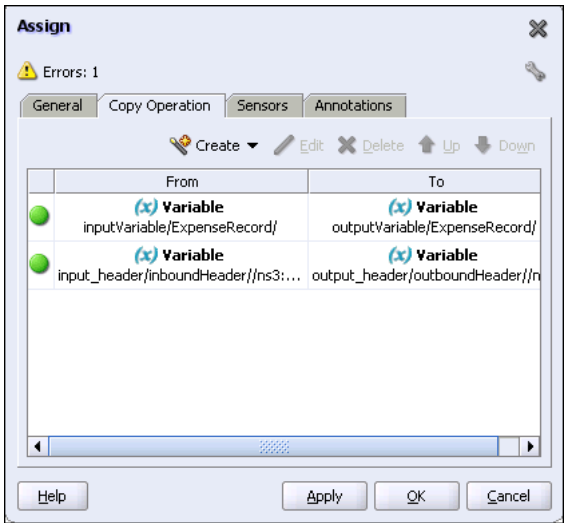
Figure 5–30 Copy Operation: Example 1

Figure 5–31 Copy Operation: Example 2

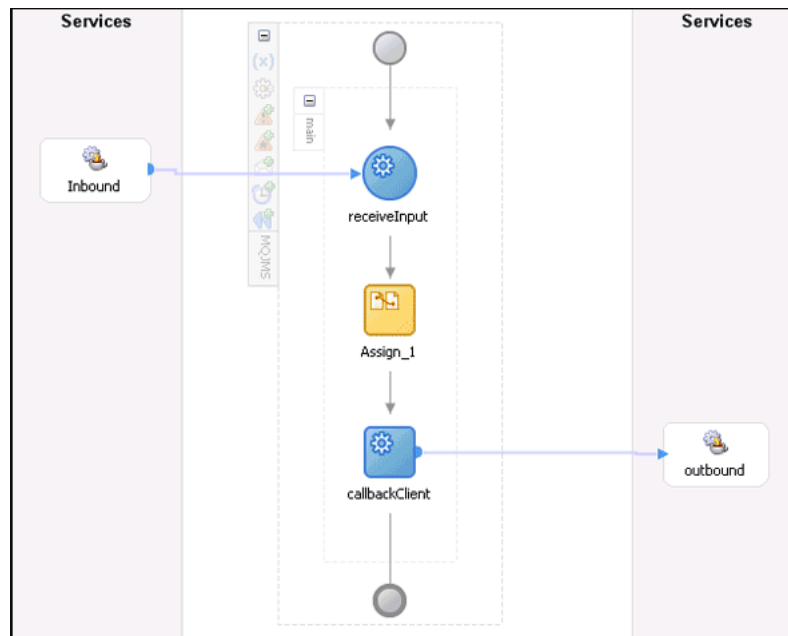


8. After performing the necessary copy operations, the Assign dialog will look like [Figure 5–32](#).

Figure 5–32 The Assign Dialog Box After Completing Copy Operation



9. The end-to- end BPEL process looks like [Figure 5–33](#).

Figure 5–33 BPEL Process

5.3 JMS Adapter Use Cases for Oracle Enterprise Service Bus

This section contains an example that showcases the following:

- An ESB service is listening on a JMS topic (OEMS database persistence) for new employee notifications.
- Upon receiving such messages it will invoke a stored procedure to insert the new employees in the company database.

The goals of the stated scenario are as follows

- Demonstrate how to configure the JMS adapter (inbound) against AQ
- Demonstrate how to configure the Database adapter to invoke a PL/SQL stored procedure
- Discuss promotion of projects from dev to prod (and related resource providers considerations)

This section comprises the following topics:

- [Meeting Prerequisites](#)
- [Preparing Database accounts](#)
- [Creating Stored Procedure](#)
- [Creating Destinations in AQ](#)
- [Creating a New ESB Project](#)
- [Creating Inbound JMS Adapter](#)
- [Creating Outbound Database Adapter Service](#)
- [Configuring Routing Service](#)
- [Promoting the Projects from Development to Production](#)
- [Configuring a JMS Adapter in Managed Mode](#)

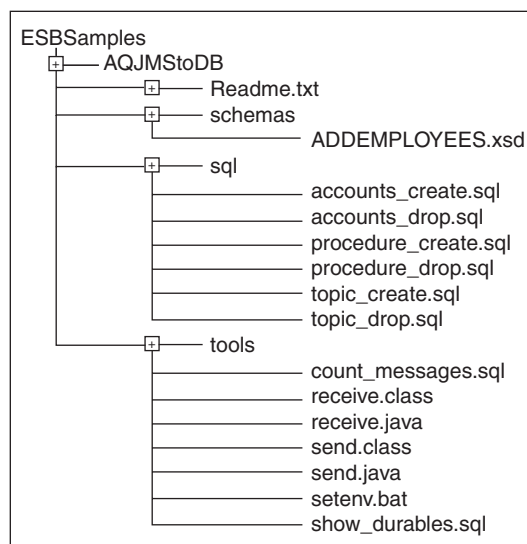
- [JMS Adapter: Configuring Database Resource Provider](#)
- [JMS Adapter: Configuring JMS Destinations in `oc4j-ra.xml`](#)
- [Database Adapter: Configuring Database Destinations in `oc4j-ra.xml`](#)
- [Database Adapter: Configuring `data-sources.xml`](#)
- [Restart Server](#)
- [Registering With ESB](#)
- [Checking the ESB Console](#)
- [Sending a JMS Message to Trigger the New Service](#)
- [Checking Execution in the ESB Control](#)

5.3.1 Meeting Prerequisites

Ensure that the following prerequisites are met, before you embark on creating this scenario:

1. Unzip `ESBSamples.zip` to `C:\ESBSamples` or any other location.
2. Make sure `ESBSamples\AQJMStoDB` is present
3. Ensure that the `ESBSamples` has the tree structure shown in the following [Figure 5–34](#).

Figure 5–34 The *ESBSamples* Folder



5.3.2 Preparing Database accounts

Perform the following steps to prepare the Database accounts for creating the scenario:

1. Open a DOS prompt and change the existing directory to the tutorial directory as shown in the following example:

```
cd \ESBSamples\AQJMStoDB
```

2. Set `ORACLE_HOME` to the database installation on your computer, as shown in the following example:

```
ex: set ORACLE_HOME=D:\ORADB_10gR2
```

3. Execute the following script to create an account:

```
sqlplus sys as sysdba @accounts_create.sql
```

5.3.3 Creating Stored Procedure

The following are the steps to create a stored procedure:

1. Change to the `sql` directory from the existing directory, as shown in the following example:

```
cd \ESBSamples\AQJMStoDB\sql
```

2. Run the following script to create the stored procedure:

```
sqlplus dbapp/dbapp @procedure_create.sql
```

5.3.4 Creating Destinations in AQ

Perform the following steps to create in AQ:

1. Change to the `sql` directory from the existing directory, as shown in the following example:

```
cd \ESBSamples\AQJMStoDB\sql
```

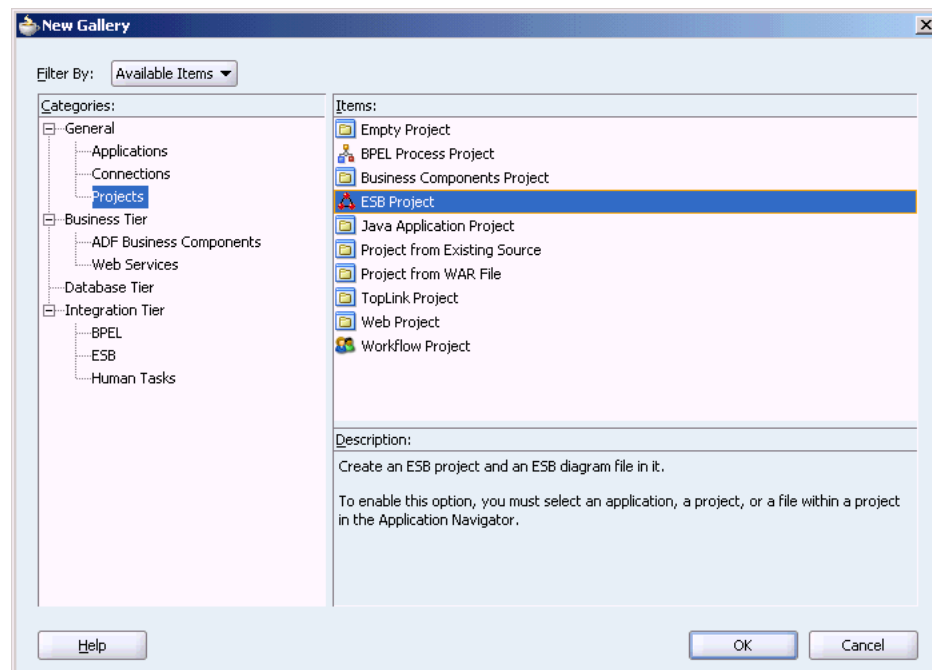
2. Using the following SQL script to create the necessary infrastructure for a JMS topic named `JMSDEMO_TOPIC`, in the database:

```
sqlplus jmsuser/jmsuser @topic_create.sql
```

5.3.5 Creating a New ESB Project

The following are the steps to create a new ESB project:

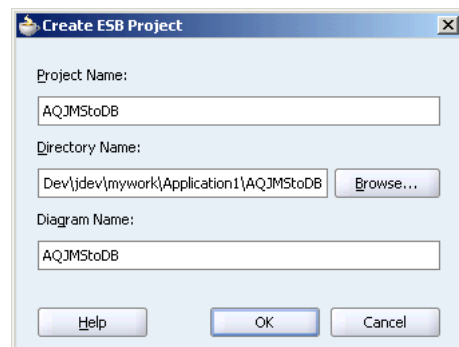
1. Open Oracle JDeveloper.
2. From the **File** menu, select **New**.
The New Gallery dialog box is displayed.
3. Select **All Technologies** from the **Filter By** box. This displays a list of available categories.
4. Expand the **General** node, and then select **Projects**.
5. Select **ESB Project** from the Items group, as shown in [Figure 5-35](#)

Figure 5–35 Creating a New ESB Project

6. Click **OK**.

The Create ESB Project dialog box is displayed.

7. In the **Project Name** field, enter a descriptive name. For example, AQJMStoDB, as shown in [Figure 5–36](#).

Figure 5–36 Entering the Name of the ESB Project

8. Click **OK**.

You have created a new ESB project, titled AQJMStoDB.

5.3.6 Creating Inbound JMS Adapter

The following are the steps to create an inbound JMS adapter:

1. Select **Adapter Services** from the Component Palette, and then drag and drop **JMS Adapter** into the AQJMStoDB.esb project.

The Create JMS Adapter Service dialog box is displayed, as shown in [Figure 5–37](#).

Figure 5–37 Creating JMS Adapter Service

2. Specify the following information in JMS Adapter Service dialog box:

- **Name:** Type a name for the service. In this example, type `ListenForNewEmployees`.
- **System/Group:** Retain the default value.

Figure 5–38 shows the JMS Adapter Service dialog box with the **Name** and **System/Group** fields, completed.

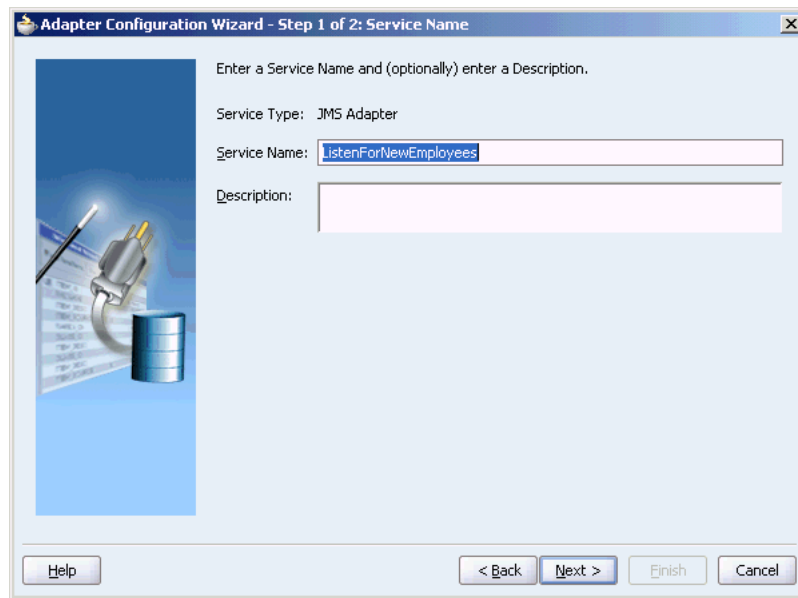
Figure 5–38 Defining the JMS Adapter Service

3. Under Adapter Service WSDL, click the **Configure adapter service wsdl** icon.

The Adapter Configuration wizard Welcome page is displayed.

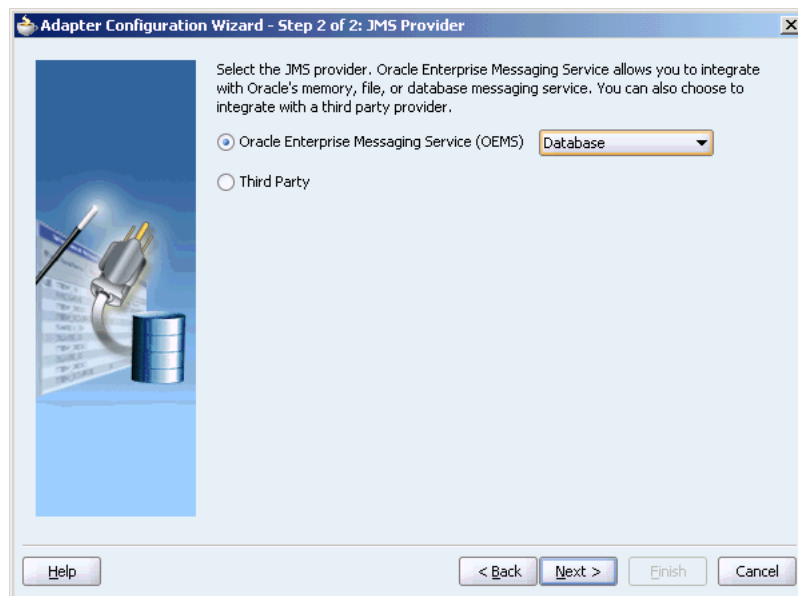
4. Click **Next**.

The Service Name dialog box is displayed with the Service Name field completed, as shown in Figure 5–39.

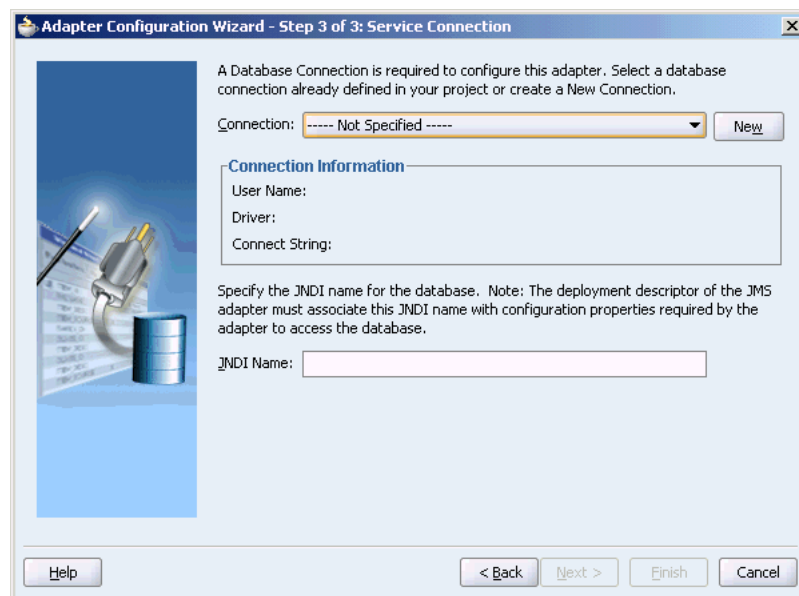
Figure 5–39 Entering a Service Name

5. Retain the service name, and click **Next**.
The JMS Provider dialog box is displayed.
6. Select **Oracle Enterprise Messaging Service (OEMS)** as a JMS Provider, click **Database** from the drop-down list.

Figure 5–40 shows the JMS Provider dialog box.

Figure 5–40 The JMS Provider Dialog Box

7. Click **Next**.
The Service Connection dialog box is displayed, as shown in Figure 5–41

Figure 5–41 The Service Connection Dialog Box

8. Click **New** to define a database connection.

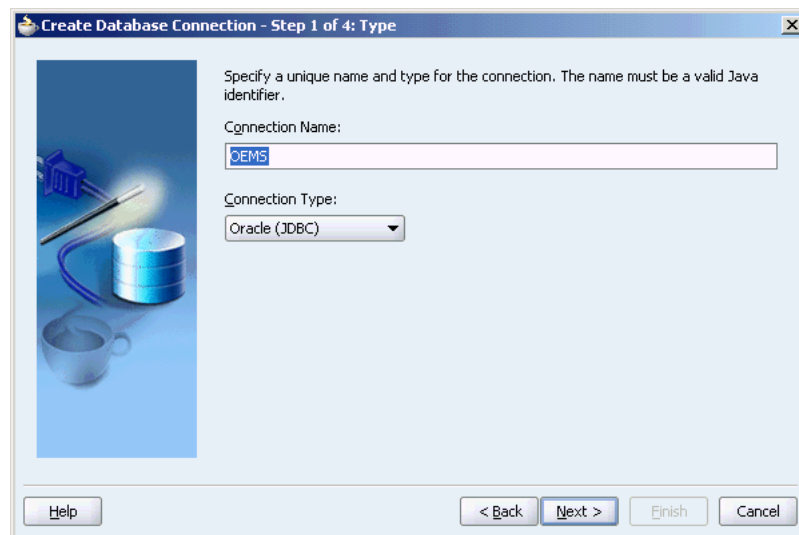
The Create Database Connection Wizard Welcome page is displayed.

9. Click **Next**.

The Type dialog box is displayed.

10. Enter the following information in the Type dialog box:
 - a. In the **Connection Name** field, specify a unique name for the database connection. In this example, type **OEMS**.
 - b. From the **Connection Type** box, select **Oracle (JDBC)**.

Figure 5–42 shows the Type dialog box.

Figure 5–42 Specifying the Connection Name and Type of Connection

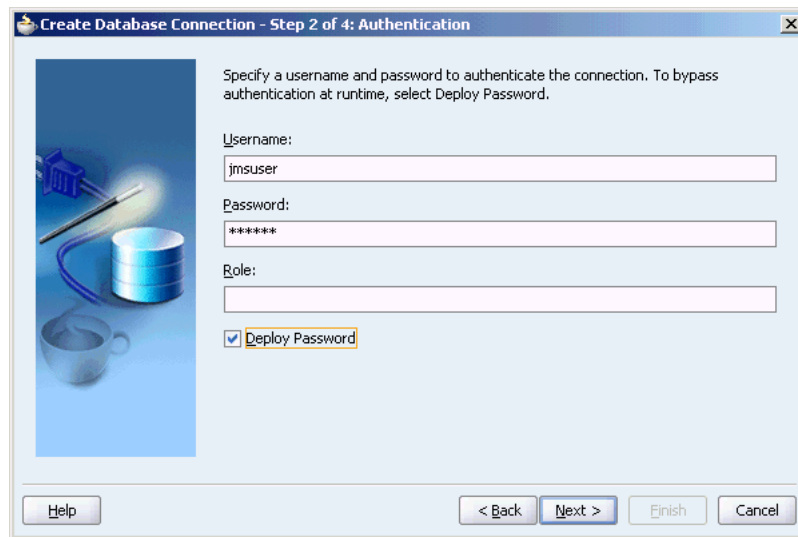
11. Click **Next**.

The Authentication dialog box is displayed.

12. Enter the authentication credentials in the following fields:
 - a. In the **UserName** field, specify a unique name for the database connection. In this example, type `jmsuser`.
 - b. In the **Password** field, specify a password for the database connection. In this example, type `jmsuser`.
 - c. Leave the **Role** field blank.
 - d. Select **Deploy Password**.

Figure 5–43 shows the Authentication dialog box with the credentials completed.

Figure 5–43 Specifying the Authentication Credentials



13. Click **Next**.

The Connection dialog box is displayed.

14. Enter information in the following fields:
 - a. In the **Driver** list, retain the default value, **Thin**.
 - b. In the **Host Name** field, retain the default value, **localhost**.
 - c. In the **JDBC Port** field, specify the port number for the database connection.
 - d. In the **SID** field, specify a unique SID value for the database connection.

Figure 5–44 shows the Connection dialog box.

Figure 5–44 Specifying the New Database Connection Information

Create Database Connection - Step 3 of 4: Connection

Specify connection details for the database machine. The database administrator should be able to provide you with this information.

Driver:

Host Name:

JDBC Port:

☒ SID:

☐ Service Name:

☐ Enter Custom JDBC URL:

Help < Back Next > Finish Cancel

15. Click Next.

The Test dialog box is displayed.

16. Click Test Connection to determine whether the specified information establishes a connection with the database.**17. Click Finish** to complete the process of creating a new database connection.

The Service Connection dialog box is displayed, providing a summary of the database connection.

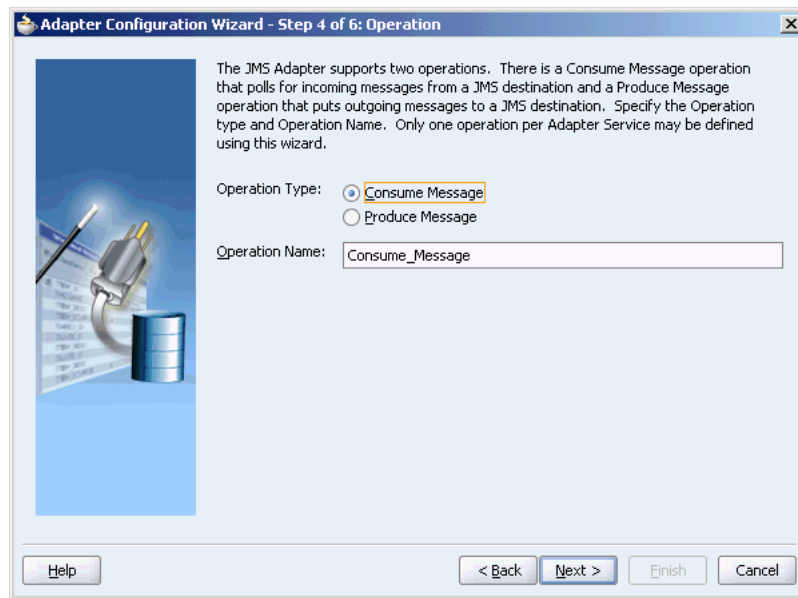
18. Click Next.

The Operation dialog box is displayed.

19. Select Consume Message.

The Operation Name field is filled, automatically.

[Figure 5–45](#) shows the Operation dialog box with **Consume Message**, selected.

Figure 5–45 Selecting an Operation for the JMS Adapter**20. Click Next.**

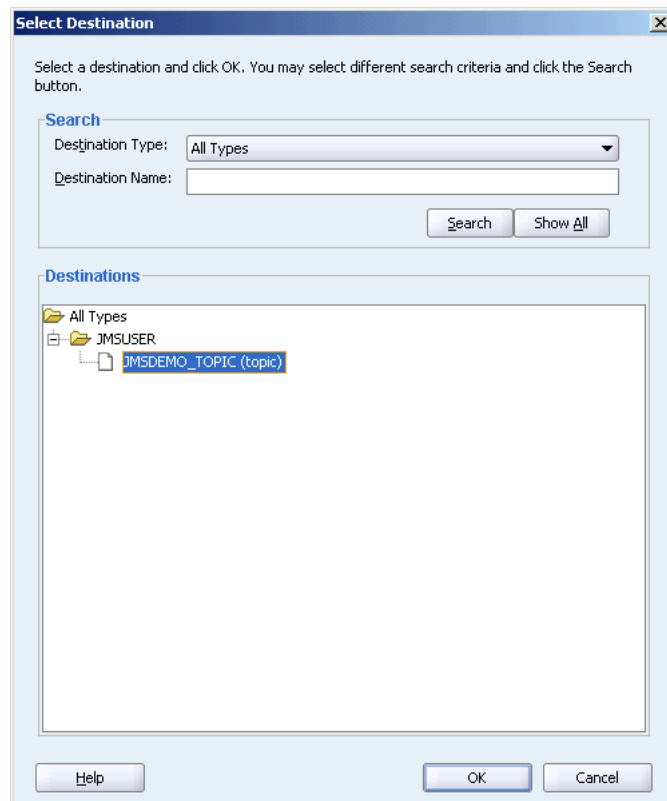
The Consume Operation Parameters dialog box is displayed.

21. Specify OEMS in the Resource Provider field, and click **Browse for the Destination Name.**

The Select Destination dialog box is displayed.

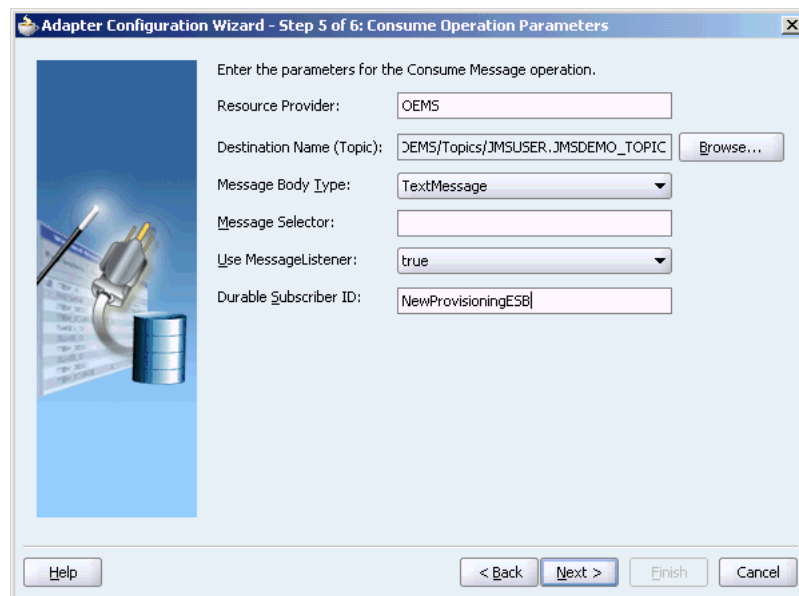
22. Under **Destinations, click **All Types**, **JMSUSER**, select **JMSDEMO_TOPIC (topic)**, and then click **OK**, as shown [Figure 5–46](#).**

The Consume Operation Parameters dialog box is displayed.

Figure 5–46 Selecting Destination

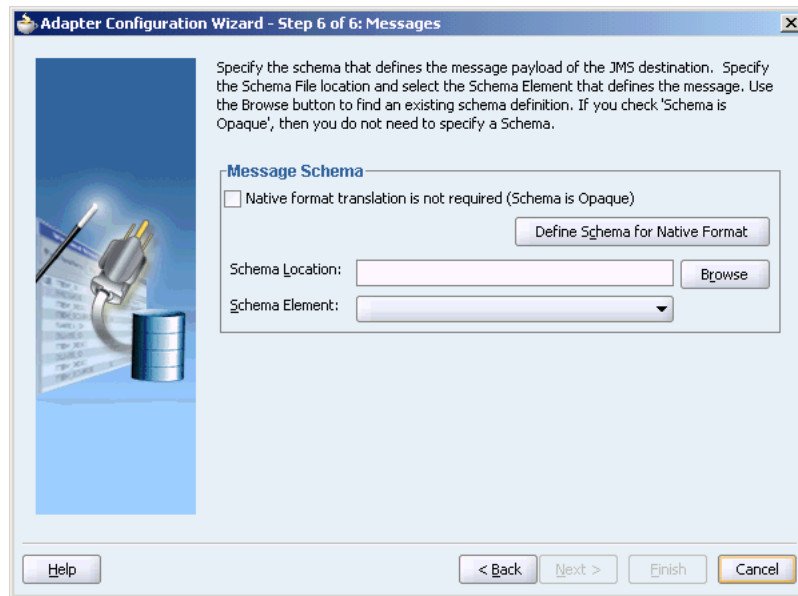
23. In the **Durable Subscriber ID** field, type `NewProvisioningESB`, and then click **Next**.

Figure 5–47 shows the Consume Operation Parameters dialog box with all the fields, completed.

Figure 5–47 The Consume Operation Parameters Dialog Box

The Messages dialog box is displayed, as shown in [Figure 5-48](#)

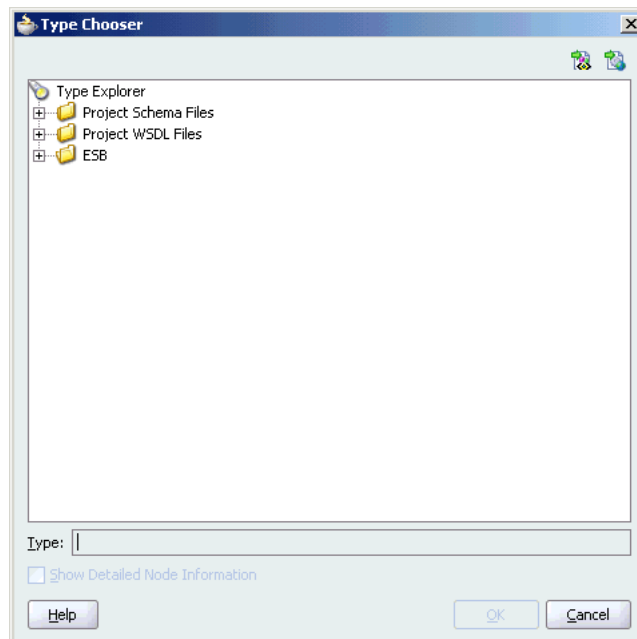
Figure 5-48 The Messages Dialog Box



24. Under **Message Schema**, next to **Schema Location**, click **Browse**.

The Type Chooser dialog box is displayed, as shown in [Figure 5-49](#).

Figure 5-49 The Type Chooser Dialog Box



25. Click on the **Import Schema File** icon (top right).

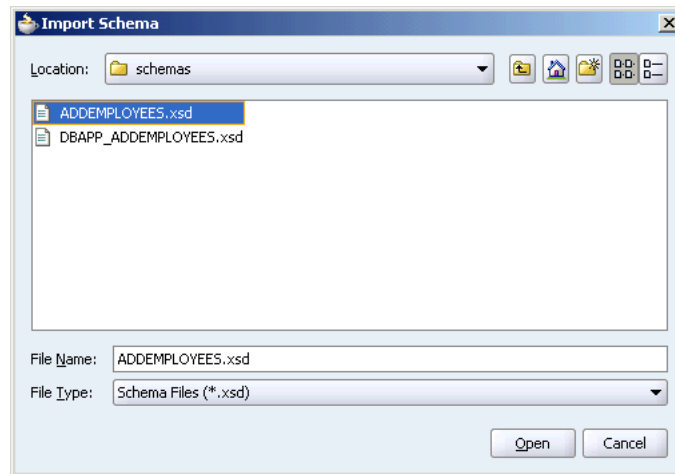
The Import Schema File dialog box is displayed.

26. Click the **Browse File System** icon.

The Import Schema dialog box is displayed.

27. Navigate to the directory where you placed this tutorial. Select the `ADDEMPLOYEES.xsd` schema, as shown in [Figure 5-50](#).

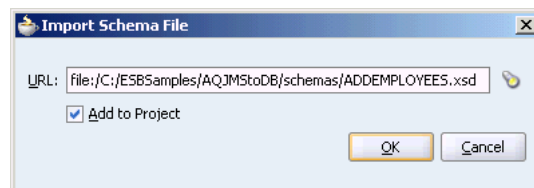
Figure 5-50 *Selecting the `ADDEMPLOYEES.xsd` Schema*



28. Click **Open**.

The Import Schema dialog box is displayed again with the **URL** field filled up, as shown in [Figure 5-51](#).

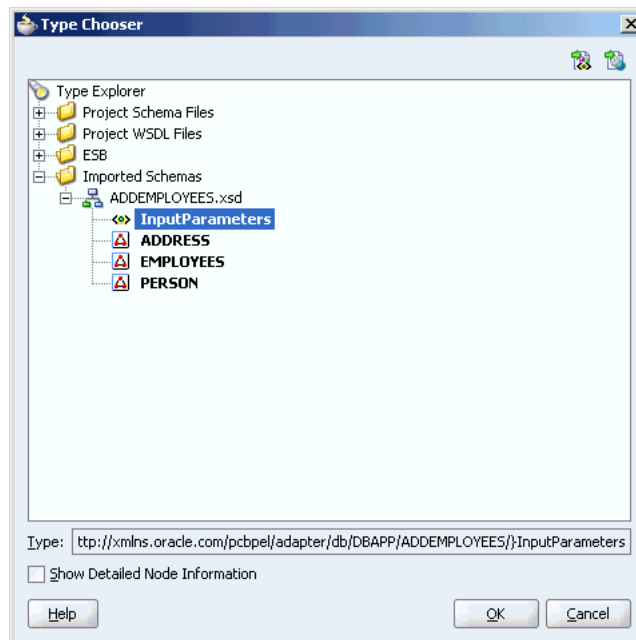
Figure 5-51 *The Import Schema Dialog Box*



29. Click **OK**. Ensure that the **Add to Project** checkbox is selected.

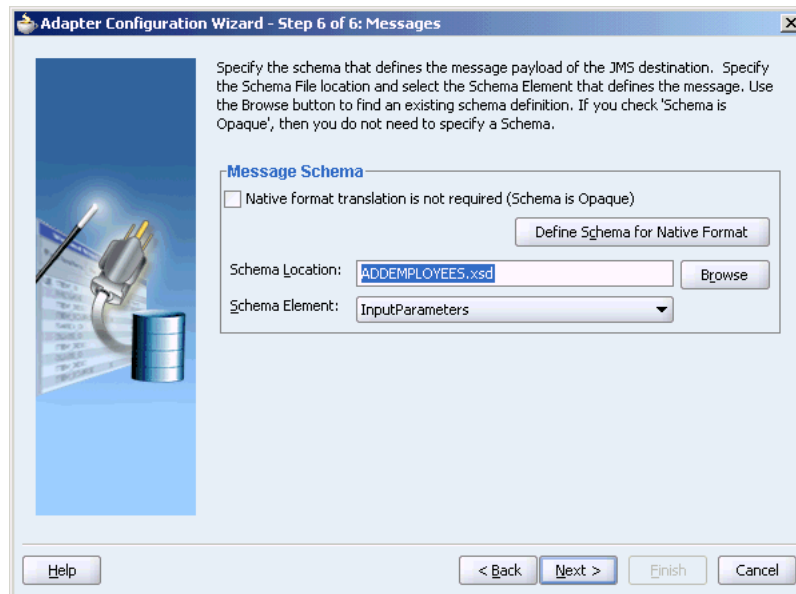
The Type Chooser dialog box, is displayed again.

30. In the Type Chooser dialog box, expand **Imported Schemas**, and then select **InputParameters** under **ADDEMPLOYEES.xsd**, as shown in [Figure 5-52](#).

Figure 5–52 Selecting the Schema in the Type Chooser Dialog Box

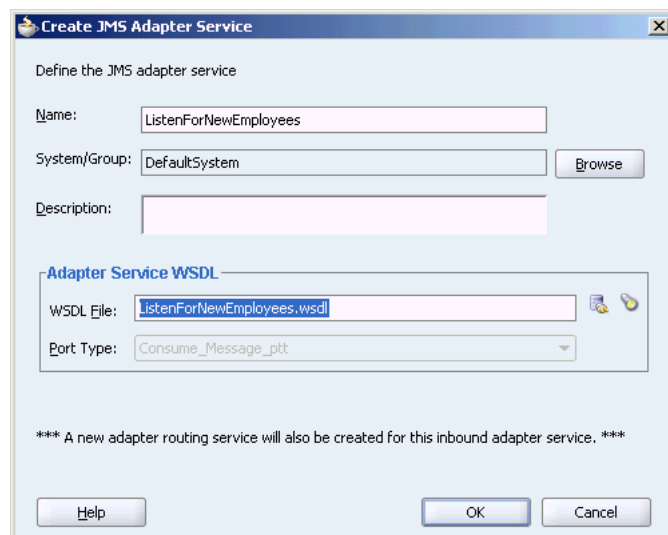
31. Click **OK**.

The Messages box is displayed with the Schema Location and Schema Element fields filled up, as shown in [Figure 5–53](#).

Figure 5–53 The Messages Dialog Box with Schema Details Filled Up

32. Click **Next**, and then click **Finish**.

The Create JMS Adapter Service dialog box is displayed, as shown in [Figure 5–54](#) with all the fields filled up.

Figure 5–54 The Create JMS Adapter Dialog Box with Definitions Filled Up**33. Click OK.**

You have created an inbound JMS Adapter service.

Tip: The timeout value in `transaction-manager.xml` must be configured to a level that is long enough for the `jta` transaction to complete. If you encounter an exception, increase the timeout value in `transaction-manager.xml`, as shown in the following example:

```
Caused by: javax.transaction.RollbackException: Transaction has
been marked
for rollback: Timed out

at
com.evermind.server.ApplicationServerTransaction.checkMarkedForRoll
back(ApplicationServerTransaction.java:612)

at
com.evermind.server.ApplicationServerTransaction.enlistResource(App
licationSer
verTransaction.java:108)

at
com.evermind.server.ApplicationServerTransaction.enlistResource(App
licationSer
verTransaction.java:87)

at
com.evermind.server.jms.EvermindSession.checkForCMT(EvermindSession
.java:1450)
```

5.3.7 Creating Outbound Database Adapter Service

Use the following steps to create an outbound DB Adapter service:

1. Select **Adapter Services** from the Component Palette, and then drag and drop **Database Adapter** into the `NewEmployee.esb` project.

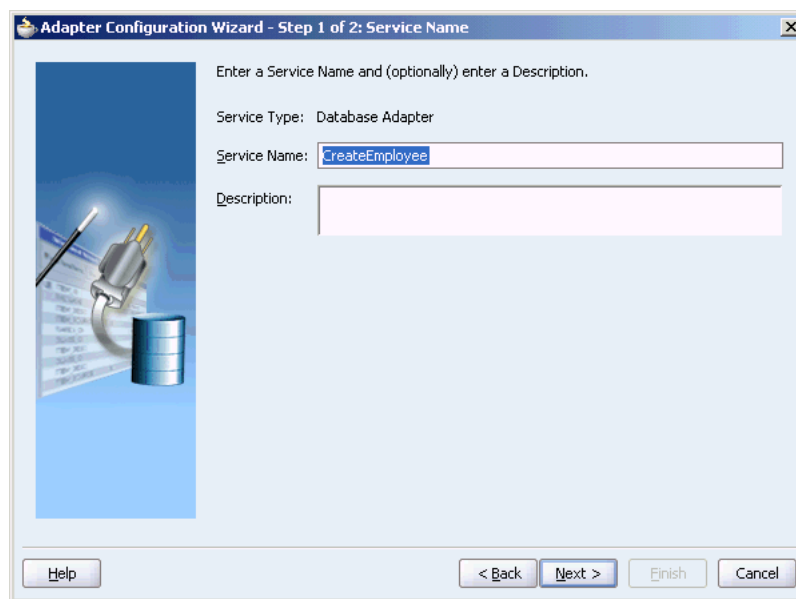
The Create Database Adapter Service dialog box is displayed.

2. Specify the following information in Database Adapter Service dialog box:
 - **Name:** Type a name for the service. In this example, type `CreateEmployee`.
 - **System/Group:** Retain the default value.
3. Under Adapter Service WSDL, click the **Configure adapter service wsdl** icon.
4. Click **Next**.

The Adapter Configuration wizard Welcome page is displayed.

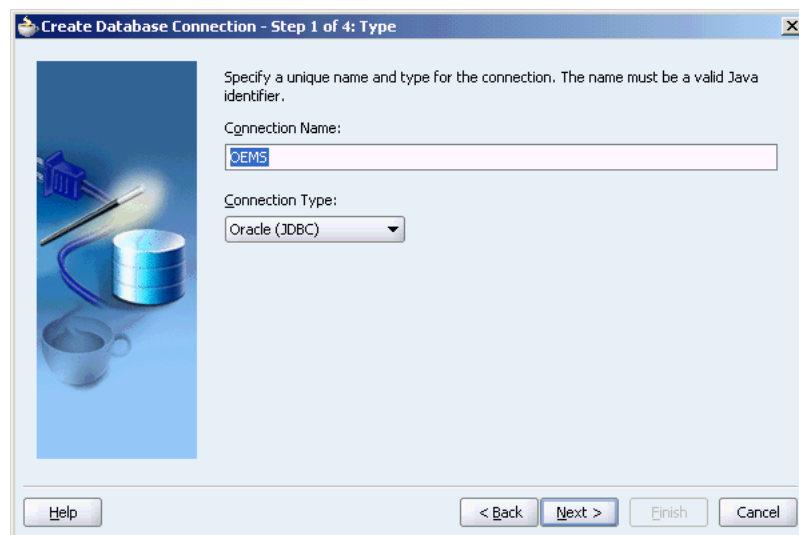
The Service Name dialog box is displayed with the Service Name field completed, as shown in [Figure 5–55](#).

Figure 5–55 Entering a Service Name



5. Retain the service name, and click **Next**.
6. Click **New** to define a database connection.
7. Click **Next**.
8. Enter the following information in the Type dialog box:
 - a. In the **Connection Name** field, specify a unique name for the database connection. In this example, type **EmployeeDB**.
 - b. From the **Connection Type** box, select **Oracle (JDBC)**.

[Figure 5–56](#) shows the Type dialog box.

Figure 5–56 Specifying the Connection Name and Type of Connection

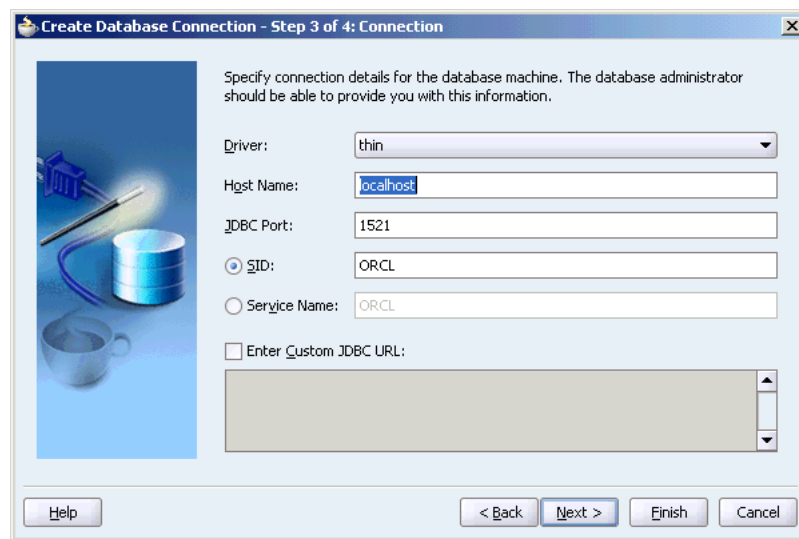
9. Click **Next**.

The Authentication dialog box is displayed.

10. Enter the authentication credentials in the following fields:

- a. In the **UserName** field, specify a unique name for the database connection. In this example, type dbapp.
- b. In the **Password** field, specify a password for the database connection. In this example, type dbapp.
- c. Leave the **Role** field blank.
- d. Select **Deploy Password**.

Figure 5–57 shows the Authentication dialog box with the credentials filled up.

Figure 5–57 Specifying the Authentication Credentials

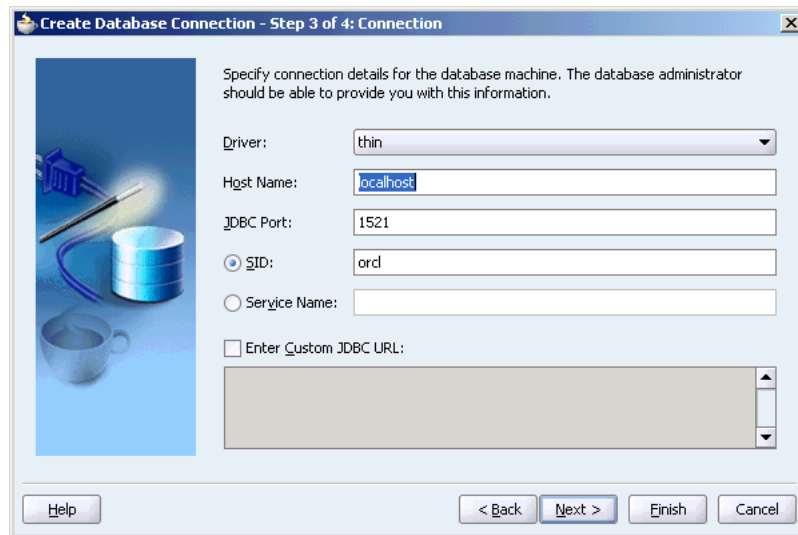
11. Click **Next**.

The Connection dialog box is displayed.

12. Enter information in the following fields:
 - a. In the **Driver** list, retain the default value, **Thin**.
 - b. In the **Host Name** field, retain the default value, **localhost**.
 - c. In the **JDBC Port** field, specify the port number for the database connection.
 - d. In the **SID** field, specify a unique SID value for the database connection.

Figure 5–58 shows the Connection dialog box.

Figure 5–58 Specifying the New Database Connection Information



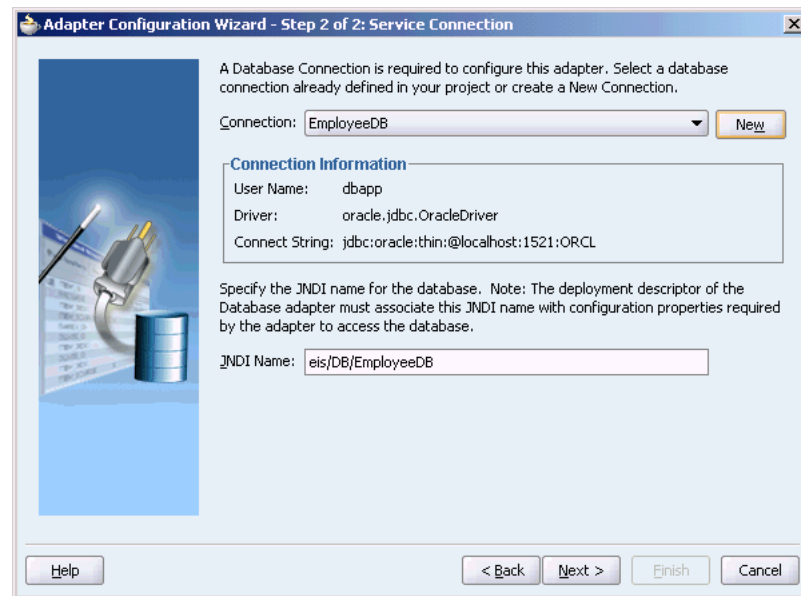
13. Click **Next**.

The Test dialog box is displayed.

14. Click **Test Connection** to determine whether the specified information establishes a connection with the database.
15. Click **Finish** to complete the process of creating a new database connection.

The Service Connection dialog box is displayed, providing a summary of the database connection, as shown in Figure 5–59.

Figure 5–59 The Service Connection Dialog Box with a Summary of the Database Connection



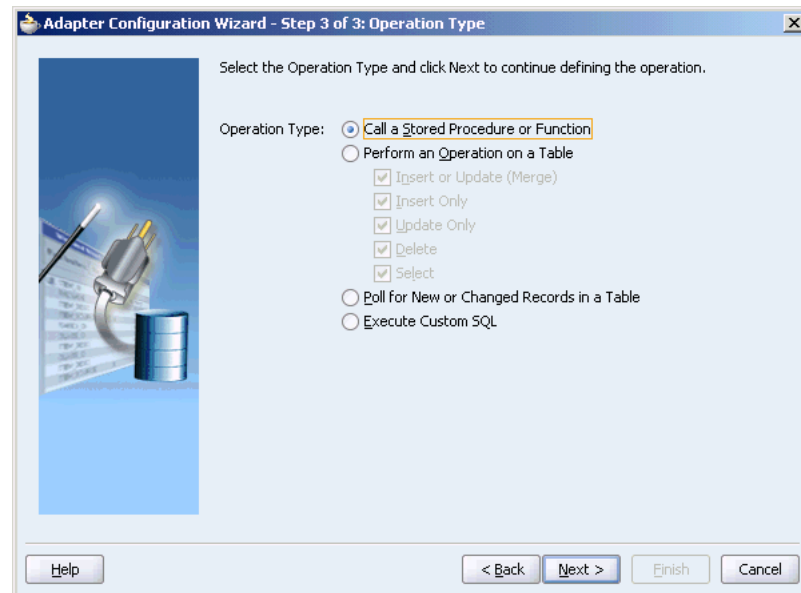
16. Click **Next**.

The Operation dialog box is displayed.

17. Select **Call a Stored Procedure or Function** as the operation type.

Figure 5–60 shows the Operation dialog box with the operation type, **Call a Stored Procedure or Function** selected.

Figure 5–60 Selecting an Operation for the Database Adapter



18. Click **Next**.

The Specify Stored Procedure dialog box is displayed.

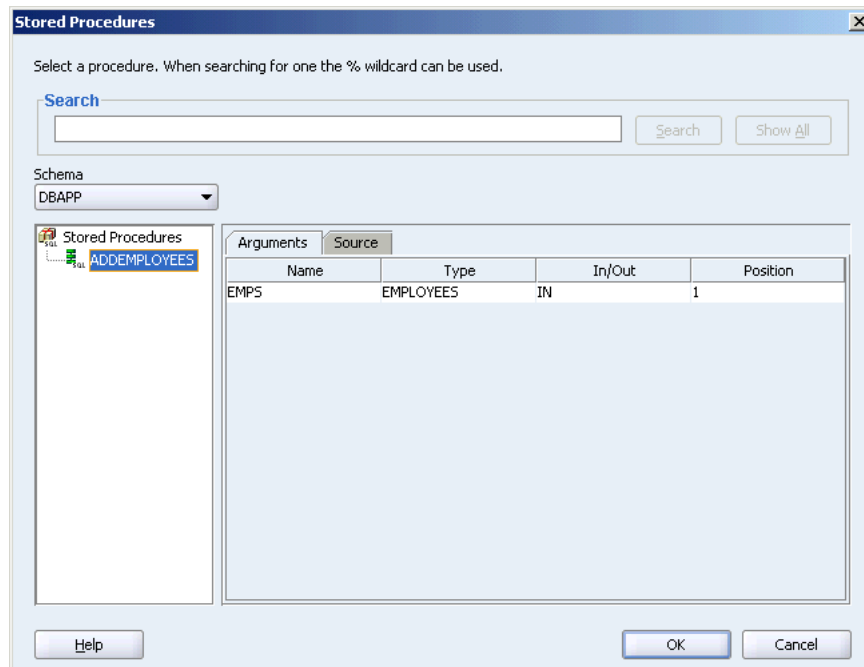
19. For Schema, select **DBAPP** from the drop-down list, and then click **Browse** to select a procedure.

The Stored Procedures dialog box is displayed.

20. Select **ADDEMPLOYEES** under Stored Procedures from DBAPP schema, and then click **OK**.

Figure 5–61 shows stored procedure **ADDEMPLOYEES** being selected in the Stored Procedures dialog box.

Figure 5–61 The Stored Procedures Dialog Box



21. Click **OK**.

The Specify Stored Procedure dialog box is displayed, as shown in Figure 5–62 with a summary of the schema and stored procedure that you specified.

Figure 5–62 Specify Stored Procedure Dialog Box

Enter a stored procedure, or a function. The procedure's package name can be included, for example, EMPLOYEE.GET_NAME, where the package name is EMPLOYEE and the procedure is GET_NAME. If the procedure does not belong in a package, enter the procedure's name. You can also browse and search for a procedure. The term 'procedure' is used to mean both stored procedures as well as functions.

Schema:

Procedure:

Arguments

Name	Type	In/Out	Position
EMPS	EMPLOYEES	IN	1

Buttons:

22. Click **Next**, and then click **Finish**.

The Create Database Adapter Service dialog box is displayed, as shown in [Figure 5–63](#) with all the field filled up.

Figure 5–63 The Create Database Adapter Service Dialog Box

Define the database adapter service

Name:

System/Group:

Description:

Adapter Service WSDL

WSDL File:

Port Type:

Buttons:

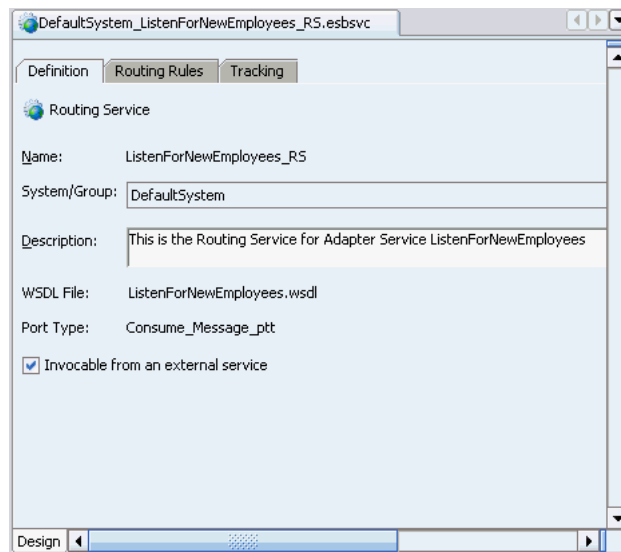
23. Click **OK**.

You have created an outbound RDBMS adapter.

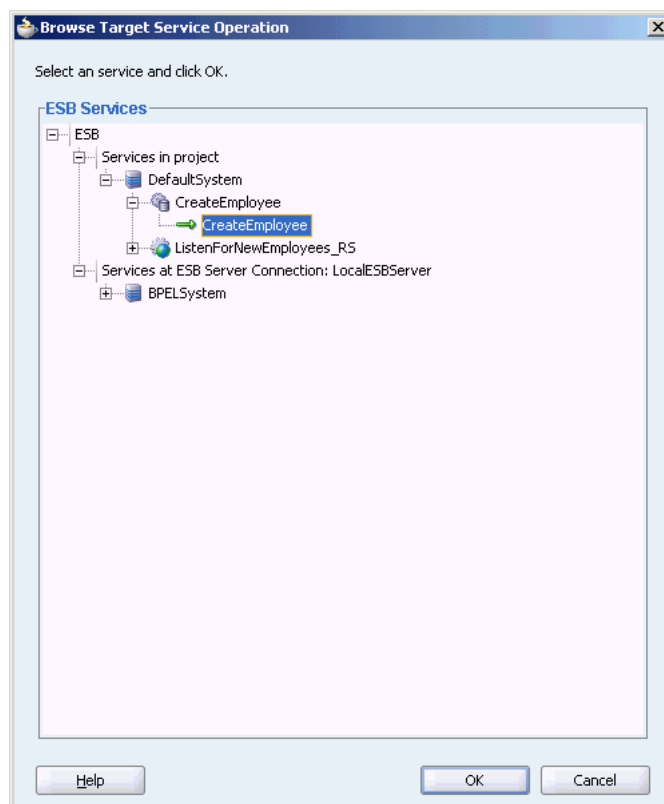
5.3.8 Configuring Routing Service

The following are the steps to configure the `ListenForNewEmployees` routing service:

1. Double-click `ListenForNewEmployees` routing service, as shown in [Figure 5–64](#)

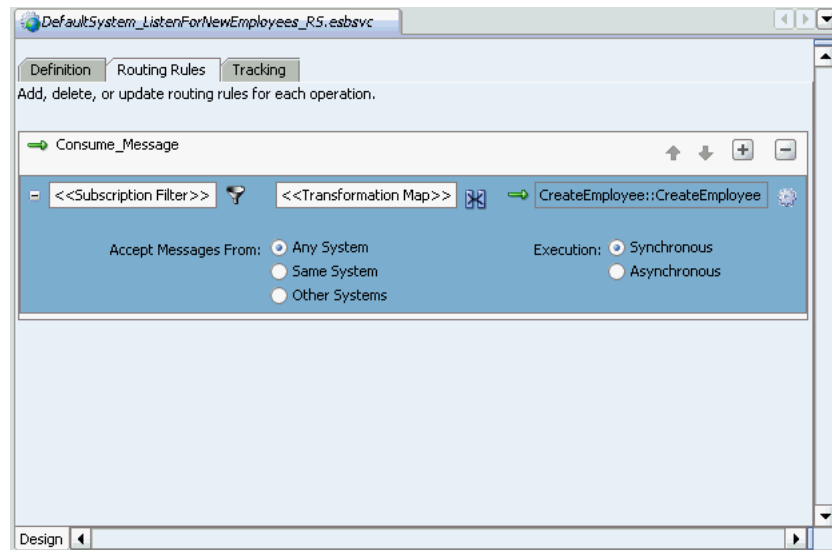
Figure 5–64 Selecting the Routing Service

2. Select the **Routing Rules** tab, and then click on the + icon to add a rule.
The Browse Target Service Operation dialog box is displayed.
3. Select the **CreateEmployee** service, as shown in [Figure 5–65](#).

Figure 5–65 The Browse Target Service Operation Dialog Box

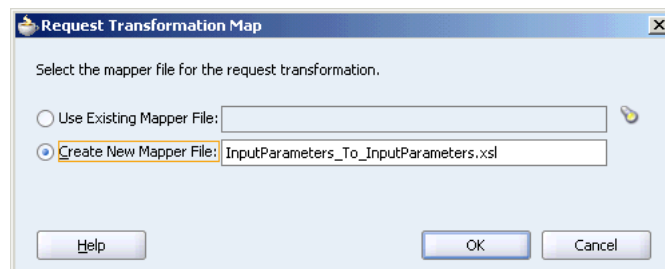
4. Click **Ok**.

The middle pane of the application window will resemble [Figure 5–66](#).

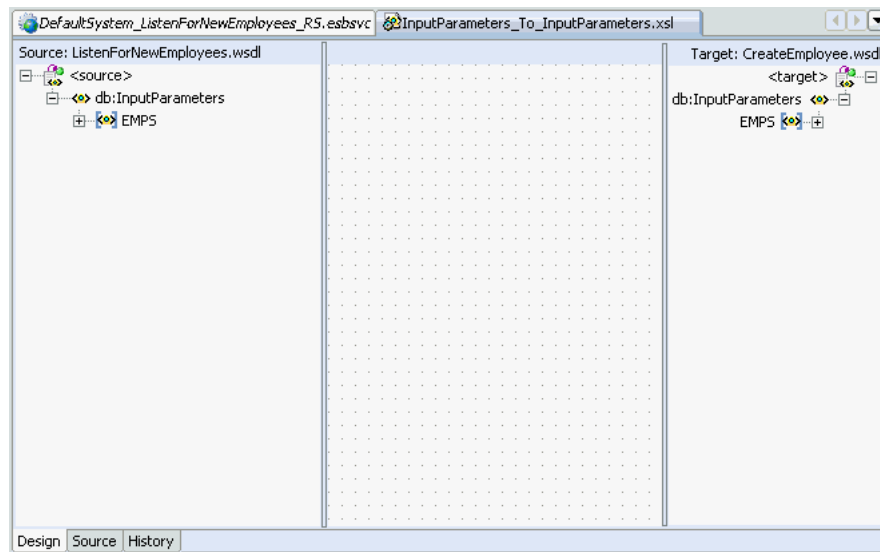
Figure 5–66 Selecting the Transformation Map

5. Double-click the **Transformation** icon, and then click the **Select Create New Mapper File** icon.

The Request Transformation Map dialog box is displayed, as shown in [Figure 5–67](#).

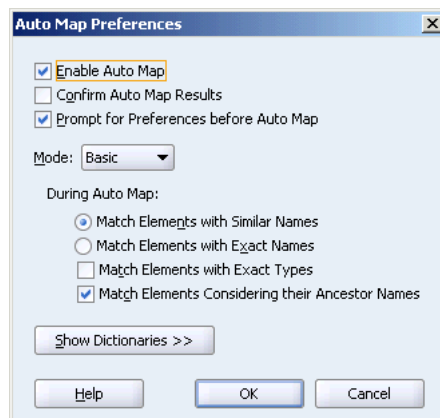
Figure 5–67 The Request Transformation Map Dialog Box

6. Select **Create New Mapper File**, accept the default name, and then click **OK**.
The mid pane of the application window will resemble [Figure 5–68](#).

Figure 5–68 Selecting the Mapper File

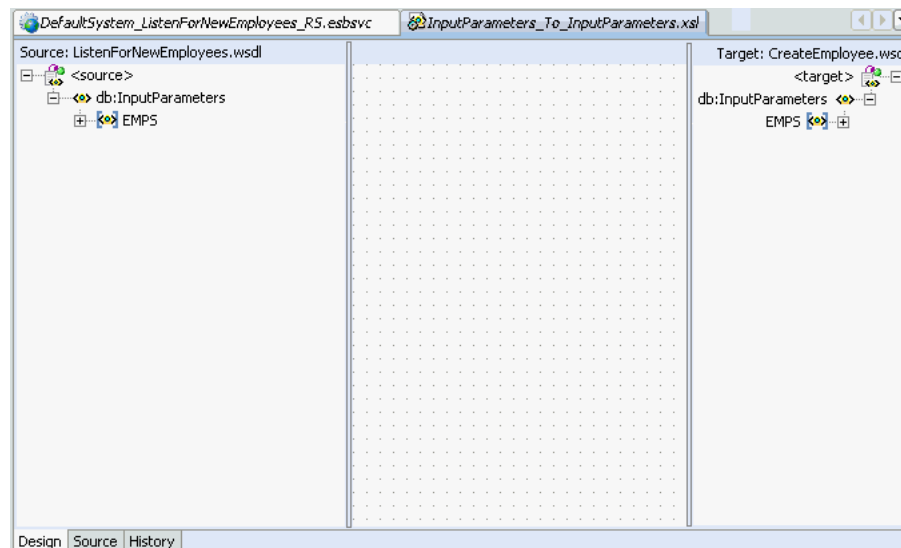
7. Select EMPS on the left-hand side of the mapper and drag it over to EMPS on the right-hand side.

The Auto Map Preferences dialog box is displayed, as shown in [Figure 5–69](#).

Figure 5–69 The Auto Map Preferences Dialog Box

8. Accept the defaults map preferences, and then click OK.

The middle pane of the application window will resemble [Figure 5–70](#).

Figure 5–70 Setting Map Preferences

9. Save and close the tab for the mapper.
10. Save and close the tab for the routing service.

5.3.9 Promoting the Projects from Development to Production

If you deploy your services, as it is, from development to production, you will end up with your adapters pointing to your development resources, for example, EmployeeDB. A non-managed mode like this has the following issues:

- Connection information is specified in the WSDL file
- Default out-of-the-box (wizard captures design time connection information assuming this as runtime connection)

In order to avoid this situation, administrators should create matching data sources on the ESB server, by using any of the following managed connections.

- **Managed mode (Toplink)**
 - Connection information is specified in `oc4j-ra.xml` through JNDI entry
 - This JNDI entry is referred to in the WSDL
- **Managed mode (OC4J)**

This is the most recommended approach.

 - Connection information is specified in `data-sources.xml`
 - The data-source is referred by JNDI entry in `oc4j-ra.xml`
 - The JNDI entry is referred to in the WSDL

5.3.10 Configuring a JMS Adapter in Managed Mode

You will require the following values from the project you created:

- **Resource Provider:** Use the value provided for resource provider in step 21 under [Creating Inbound JMS Adapter](#) in `application.xml` and `oc4j-ra.xml`.

- **JNDI name:** Use the JNDI name specified in this example, `eis/Jms/jmsuser` in `oc4j-ra.xml`.

5.3.11 JMS Adapter: Configuring Database Resource Provider

Use the following steps to configure database resource provider in `application.xml`:

1. Navigate to the location where ESB server is installed, that is, navigate to `%ESB_HOME%\j2ee\home\config`. For example:

```
cd D:\ORACLE\OracleESB_beta\j2ee\home\config
```

2. Open `application.xml`, and then add the following entry:

```
<resource-provider class="oracle.jms.OjmsContext" name="OEMS">
<description>Resource provider for the OEMS database</description>
<property name="url" value="jdbc:oracle:thin:cmsuser/cmsuser@localhost:1521:XE"
/>
</resource-provider>
```

Ensure that the database connection name matches that of the database name specified in this use case, which is, OEMS.

5.3.12 JMS Adapter: Configuring JMS Destinations in `oc4j-ra.xml`

Use the following steps to configure JMS Adapter destinations in `oc4j-ra.xml`:

1. Navigate to the JMS adapter deployment folder in the location where ESB server is installed. For example:

```
D:\ORACLE\OracleESB_beta\j2ee\home\application-deployments\default\JmsAdapter
```

2. Modify the JNDI name and the resource provider name in `oc4j-ra.xml`, as shown in the following example:

```
<connector-factory location="eis/Jms/OEMS" connector-name="Jms Adapter">
<config-property name="connectionFactoryLocation"
value="java:comp/resource/OEMS/TopicConnectionFactory/myTCF"/>
<config-property name="factoryProperties" value=""/>
<config-property name="acknowledgeMode" value="AUTO_ACKNOWLEDGE"/>
<config-property name="isTopic" value="true">
<config-property name="isTransacted" value="true"/>
<config-property name="username" value="cmsuser"/>
<config-property name="password" value="cmsuser"/>
<connection-pooling use="none">
</connection-pooling>
<security-config use="none">
</security-config>
</connector-factory>
```

Note that you have to modify the JNDI name and the resource provider name as mentioned in [Configuring a JMS Adapter in Managed Mode](#).

5.3.13 Database Adapter: Configuring Database Destinations in `oc4j-ra.xml`

Use the following steps to configure JMS Adapter destinations in `oc4j-ra.xml`:

1. Navigate to the Database adapter deployment folder in the location where ESB server is installed. For example:

```
D:\ORACLE\OracleESB_beta\j2ee\home\application-deployments\default\DBAdapter
```

2. Modify `oc4j-ra.xml`, as shown in the following example:

```
<connector-factory location="eis/DB/EmployeeDB" connector-name="Database
Adapter">
    <config-property name="xADataSourceName" value="jdbc/EmployeeDB-XA"/>
    <config-property name="dataSourceName" value="jdbc/EmployeeDB"/>
    <config-property name="platformClassName"
value="oracle.toplink.platform.database.Oracle9Platform"/>
    <config-property name="usesNativeSequencing" value="true"/>
    <config-property name="sequencePreallocationSize" value="50"/>
    <config-property name="defaultNChar" value="false"/>
    <config-property name="usesBatchWriting" value="true"/>
    <connection-pooling use="none">
    </connection-pooling>
    <security-config use="none">
    </security-config>
</connector-factory>
```

5.3.14 Database Adapter: Configuring `data-sources.xml`

Use the following steps to configure JMS Adapter destinations in `oc4j-ra.xml`:

- Navigate to the config folder in the location where ESB server is installed. For example:

```
D:\ORACLE\OracleESB_beta\j2ee\home\config
```

- Modify `data-sources.xml`, as shown in the following example:

```
<managed-data-source name="EmployeeDB-XA" connection-pool-name="EmployeePool"
jndi-name="jdbc/EmployeeDB-XA"/>
<managed-data-source name="EmployeeDB" connection-pool-name="EmployeePool"
jndi-name="jdbc/EmployeeDB"/>
<connection-pool name="EmployeePool">
    <connection-factory factory-class="oracle.jdbc.pool.OracleDataSource"
user="dbapp" password="dbapp" url="jdbc:oracle:thin:@//localhost:1521/XE">
    </connection-factory>
```

5.3.15 Restart Server

Use the following steps to restart your ESB server after configuring resource providers:

1. Click **Start** menu, point to **ESB**, and then select **Stop ESB**.
2. Click **Start** menu, point to **ESB**, and then select **Start ESB**.

5.3.16 Registering With ESB

This is the final step in the design-time. Use the following steps to register with ESB:

1. Save All
2. Right-click **AQJMStoDB** project, select **Register with ESB**, and then click **LocalIntegrationServer**.

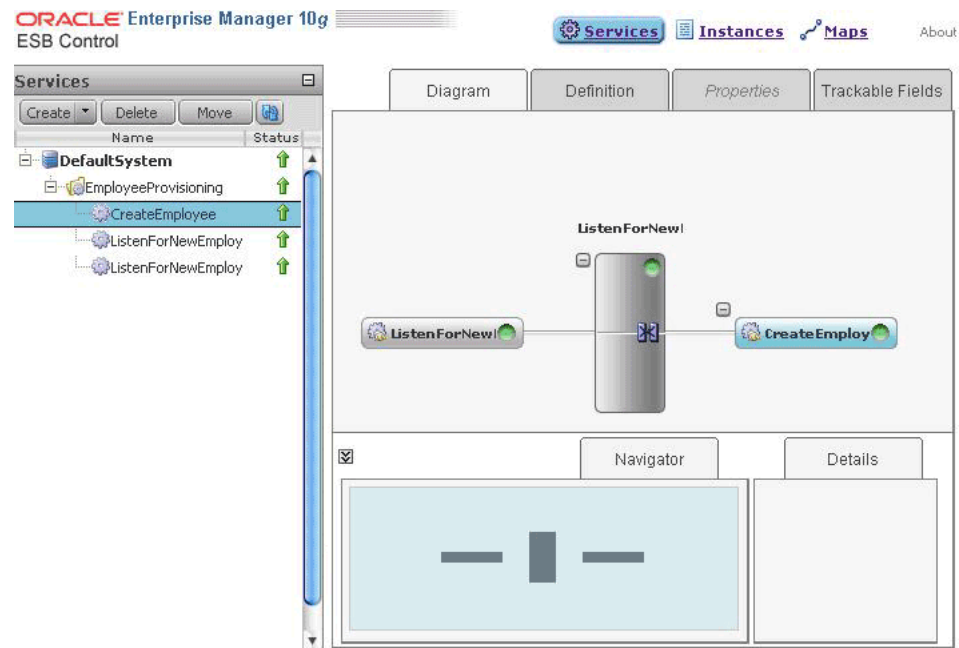
5.3.17 Checking the ESB Console

To check the ESB control, open the ESB Console. For example:

<http://localhost:8888/esb/esb/EsbConsole.html>

Now, your service window will resemble [Figure 5–71](#).

Figure 5–71 The ESB Console



5.3.18 Sending a JMS Message to Trigger the New Service

The following are the steps to send a JMS message to trigger the new service:

1. Open a DOS prompt and go to the tools directory, under tutorials, as shown in the following example:

```
cd C:\ESBSamples\AQJMStoDB\tools
```

2. Edit `setenv.bat` to reflect your environment (`classpath`) and OEMS properties to reflect your server settings (location, port, and so on). Then send a message as shown in the following example:

```
C:\ESBSamples\AQJMStoDB\tools>setenv
C:\ESBSamples\AQJMStoDB\tools>java send JMSDEMO_TOPIC

-----
OEMS.155 - simple JMS send / JMS 1.02 / Database AQ / no JNDI
-----
Connection factory = oracle.jms.AQjmsTopicConnectionFactory@ab95e6

destination: JMSUSER.JMSDEMO_TOPIC
message      :

<?xml version = '1.0' encoding = 'UTF-8'?>
<db:InputParameters
xmlns:db="http://xmlns.oracle.com/pcbpel/adapter/db/DBAPP/AD
EMPLOYEES/">
  <EMPS>
    <EMPS_ITEM>
      <FNAME>John</FNAME>
      <MIDDLE>W</MIDDLE>
      <LNAME>Doe</LNAME>
```

```

<ADDR>
  <STREET>100 Oracle Parkway</STREET>
  <CITY>Redwood Shores</CITY>
  <STATE>CA</STATE>
  <ZIP>94065</ZIP>
</ADDR>
</EMPS_ITEM>
</EMPS>
</db:InputParameters>

message was sent with ID=ID:F85494669F764BFAB51CC75367D07B35

```

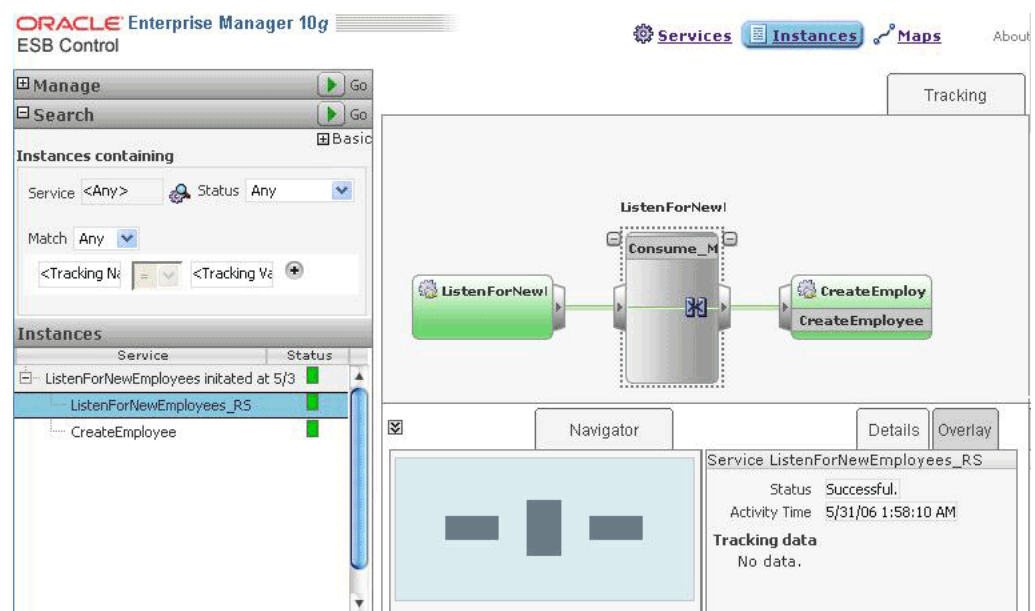
5.3.19 Checking Execution in the ESB Control

Use the following steps to check execution in the ESB control:

1. Open the ESB Console.
2. Click **Instances** on top-right corner.
3. Click the green arrow next to **Search**.

An instance resembling [Figure 5-72](#) is displayed.

Figure 5-72 ESB Control Instance



Oracle Application Server Adapter for Native MQSeries

This chapter describes how to use the Oracle Application Server Adapter for MQSeries (MQSeries adapter), which works in conjunction with Oracle BPEL Process Manager and Oracle Enterprise Service Bus as an external service. It contains the following sections:

- [Section 6.1, "MQSeries Message Queuing Concepts"](#)
- [Section 6.2, "Introduction to Native MQSeries Adapter"](#)
- [Section 6.3, "MQSeries Adapter Concepts"](#)
- [Section 6.4, "Configuring the MQSeries Adapter"](#)
- [Section 6.5, "MQSeries Adapter Use Cases for Oracle BPEL Process Manager"](#)

6.1 MQSeries Message Queuing Concepts

Message queuing is a technique for asynchronous program-to-program communication. It enables application integration by allowing independent applications on a distributed system to communicate with each other. One application sends messages to a queue owned by a queue manager, and another application retrieves the messages from the queue. The communication between applications is maintained even if the applications are running at different times or are temporarily unavailable.

The following list explains the basic concepts of message queuing:

- **Messaging**
Messaging is the mechanism that allows two entities to communicate by sending and receiving messages. Messaging can be of two types, synchronous and asynchronous. In *synchronous* messaging, sender of the message places a message on a message queue and then waits for a reply to its message before resuming its own processing. In *asynchronous* messaging, sender of the message proceeds with its own processing without waiting for a reply.
- **Message**
Messages are structured data sent by one program and intended for another program.
- **Message Queue**

Message queues are objects that store messages in an application. Applications can put messages to the queues and get messages from the queues. A queue is managed by a queue manager.

- Queue Manager

A queue manager provides the messaging and queuing services to applications through an application programming interface. It provides access to the queues and also transfers messages to another queue manager through message channels.

- Message Channel

A message channel provides a communication path between two queue managers. It connects the queue managers. A message channel can transmit messages in one direction only.

- Transmission Queue

A transmission queue is used to temporarily store messages that are destined for a remote queue manager.

- Message Segment

If a message is very large, it can be divided into multiple small messages called segments. Each segment has a group ID and an offset. All segments of a message have same group ID. The last segment of the message is marked with a flag.

- Message Group

A message group consists of a set of related messages with the same group ID. Each message in a message group has a message sequence number. The last message in a message group is marked with a flag.

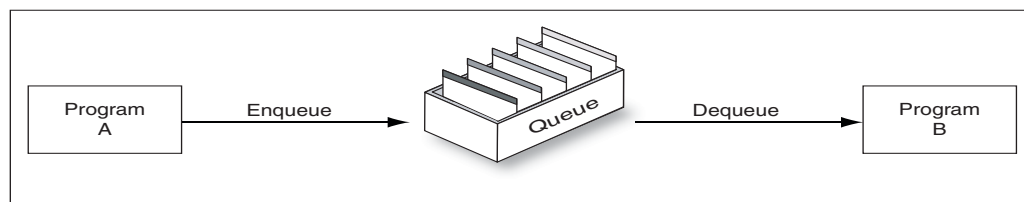
- Cluster

A cluster is a group of queue managers that are logically associated.

- Enqueue/Dequeue

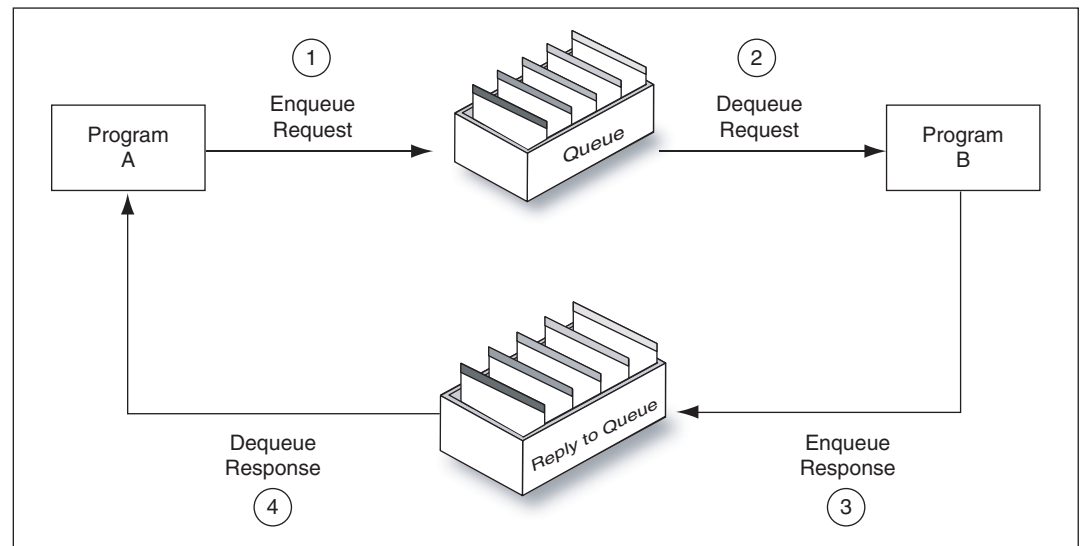
To enqueue is to put a message in a queue whereas to dequeue is to get a message from a queue as shown in [Figure 6-1](#).

Figure 6-1 Enqueue/Dequeue



- Request/Response

In request/reply interaction, a program sends a message to another program asking for a reply. The request message contains information about to where the reply should be sent. The receiving program sends a reply message in response to the request message. The request/reply interaction is shown in [Figure 6-2](#).

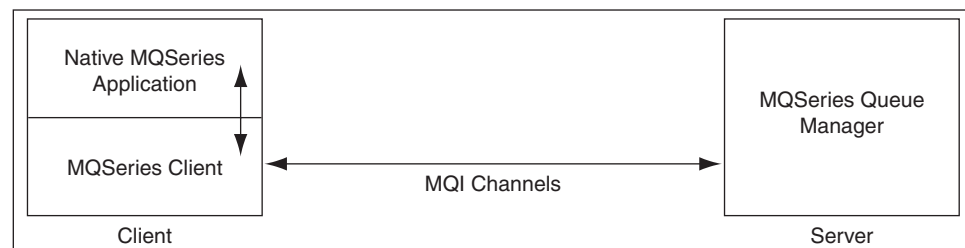
Figure 6–2 Request/Response Interaction

See Also: [Section 6.3.1.2, "Dequeue Message"](#) for information about the interactions scenarios supported by the MQSeries adapter.

6.1.1 MQSeries Concepts

Messaging and Queuing Series (MQSeries) is a set of products and standards developed by IBM. MQSeries provides a queuing infrastructure that provides guaranteed message delivery, security, and priority-based messaging.

The communication process between an MQSeries application and a MQSeries server is shown in [Figure 6–3](#). An MQSeries client enables an application to connect to a queue manager on a remote machine.

Figure 6–3 The MQSeries Communication Process

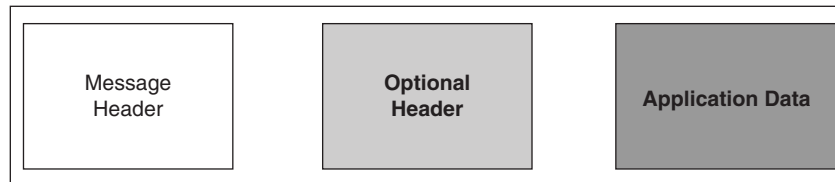
Every queue in MQSeries belongs to a queue manager. A queue manager has a unique name and provides messaging and queuing services to applications through a Message Queue Interface (MQI) channel. A queue manager also provides access to the queues created on it and transfers messages to other queue managers through message channels.

In MQSeries, data is sent in the form of messages. The sending application constructs a message and sends it to a queue by using the API calls. The message remains in the queue until the receiving application is ready to receive it. The receiving application gets the messages from the queue by using the API calls.

For sending messages to a remote queue, the remote queue definition must be defined locally. The remote queue definition consists of the destination queue name and the transmission queue name.

[Figure 6–4](#) displays message structure of an MQSeries message.

Figure 6–4 MQSeries Message



An MQSeries message consists of following parts, as shown in [Figure 6–4](#):

- **Message Header**
The message header contains information such as, unique message ID, message type, message priority, and routing information. Every MQSeries message must have a message header.
- **Optional Header**
The optional header is required for communication with specific applications, such as CICS application.

See Also: [Section 6.3.10, "Integration with CICS"](#)
- **Application Data**
The actual data. For example, a record from an indexed or flat file or a row or column from a DB2 table.

6.2 Introduction to Native MQSeries Adapter

Oracle BPEL Process Manager and Oracle Enterprise Service Bus include the MQSeries adapter. The MQSeries adapter enables applications to connect to MQSeries queue managers and place MQSeries messages on queues or to remove MQSeries messages from queues.

This section contains the following sections:

- [Section 6.2.1, "The Need for MQSeries Adapter"](#)
- [Section 6.2.2, "MQSeries Adapter Integration with Oracle BPEL Process Manager"](#)
- [Section 6.2.3, "MQSeries Adapter Integration with Oracle Enterprise Service Bus"](#)

6.2.1 The Need for MQSeries Adapter

The MQSeries adapter provides all native MQSeries functionalities. Although you can configure the Oracle Application Server Adapter for Java Message Service (JMS adapter) with MQSeries provider, it provides only the JMS functionalities provided by MQSeries and not the native MQSeries functionalities. The following list explains the advantages of MQSeries adapter over the JMS adapter for MQSeries:

- The MQSeries adapter supports Positive Action Notification (PAN)/Negative Action Notification (NAN).

- The MQSeries adapter supports report messages such as, confirmation on delivery, confirmation on arrival, exception report, and expiry report.
- The MQSeries adapter supports sending unwanted or corrupted message to a dead-letter queue.
- The MQSeries adapter provides advanced filter options, such as filtering message belonging to a group.
- The MQSeries adapter is faster and easier to use.

6.2.2 MQSeries Adapter **Integration with** Oracle BPEL Process Manager

The MQSeries adapter is automatically integrated with Oracle BPEL Process Manager. When you create a partner link or a MQ adapter service in Oracle JDeveloper, the Adapter Configuration Wizard is started.

This wizard enables you to select and configure the MQSeries adapter or other OracleAS Adapters, as shown in [Figure 1-3](#). The Adapter Configuration Wizard then prompts you to enter a service name, as shown in [Figure 1-4](#). When configuration is complete, a WSDL file of the same name is created in the Applications Navigator section of Oracle JDeveloper. This WSDL file contains the configuration information you specify with the Adapter Configuration Wizard.

The Operations window of the Adapter Configuration Wizard prompts you to select an operation to perform. Based on your selection, different Adapter Configuration Wizard windows appear and prompt you for configuration information.

[Table 6-1](#) lists the available operations and provides references to sections that describe the information about these operations.

Table 6-1 Supported Operations for Oracle BPEL Process Manager

Operation	See Section...
Enqueue Message	Section 6.3.1.1, "Enqueue Message"
Dequeue Message	Section 6.3.1.2, "Dequeue Message"
Request-Response	Section 6.3.1.3, "Request-Response (Oracle BPEL Process Manager as a Client)" Section 6.3.1.4, "Synchronous Request-Response (Oracle BPEL Process Manager as Server)" Section 6.3.1.5, "Asynchronous Request-Response (Oracle BPEL Process Manager as Server)"

6.2.3 MQSeries Adapter **Integration with** Oracle Enterprise Service Bus

The MQSeries adapter is automatically integrated with Oracle Enterprise Service Bus. When you create an MQ adapter service in JDeveloper ESB Designer, the Adapter Configuration Wizard is started as shown in [Figure 1-2](#).

This wizard enables you to select and configure the MQSeries adapter. When configuration is complete, a WSDL file of the same name is created in the Applications Navigator section of Oracle JDeveloper. This WSDL file contains the configuration information you specify with the Adapter Configuration Wizard.

The Operations window of the Adapter Configuration Wizard prompts you to select an operation to perform. Based on your selection, different Adapter Configuration Wizard windows appear and prompt you for configuration information. [Table 6-2](#) lists the available operations and provides references to sections that describe the configuration information you must provide.

Table 6–2 Supported Operations for Oracle Enterprise Service Bus

Operation	See Section...
Enqueue Message	Section 6.3.1.1, "Enqueue Message"
Dequeue Message	Section 6.3.1.2, "Dequeue Message"
Request-Response	Section 6.3.1.6, "Request-Response Synchronous (Oracle Enterprise Service Bus as Server)"

6.3 MQSeries Adapter Concepts

This section explains the following concepts of the MQSeries adapter:

- [Section 6.3.1, "Messaging Scenarios"](#)
- [Section 6.3.2, "Message Properties"](#)
- [Section 6.3.3, "Correlation Schemas"](#)
- [Section 6.3.4, "Distribution List Support"](#)
- [Section 6.3.5, "Report Messages"](#)
- [Section 6.3.6, "Filter-by Criteria"](#)
- [Section 6.3.7, "Message Delivery Failure Options"](#)
- [Section 6.3.8, "Message Segmentation"](#)
- [Section 6.3.9, "Message Grouping"](#)
- [Section 6.3.10, "Integration with CICS"](#)

6.3.1 Messaging Scenarios

The MQSeries adapter supports the following messaging scenarios:

- [Enqueue Message](#)
- [Dequeue Message](#)
- [Request-Response \(Oracle BPEL Process Manager as a Client\)](#)
- [Synchronous Request-Response \(Oracle BPEL Process Manager as Server\)](#)
- [Asynchronous Request-Response \(Oracle BPEL Process Manager as Server\)](#)
- [Request-Response Synchronous \(Oracle Enterprise Service Bus as Server\)](#)

Note: The MQSeries adapter does not support XA transactions.

6.3.1.1 Enqueue Message

In this scenario, the MQSeries adapter connects to a specific queue managed by a queue manager and then writes the message to the queue. For outbound messages sent from Oracle BPEL Process Manager or Oracle Enterprise Service Bus, the MQSeries adapter performs the following operations:

1. Receives message from Oracle BPEL Process Manager or Oracle Enterprise Service Bus.
2. Formats the XML content as specified at the design time.

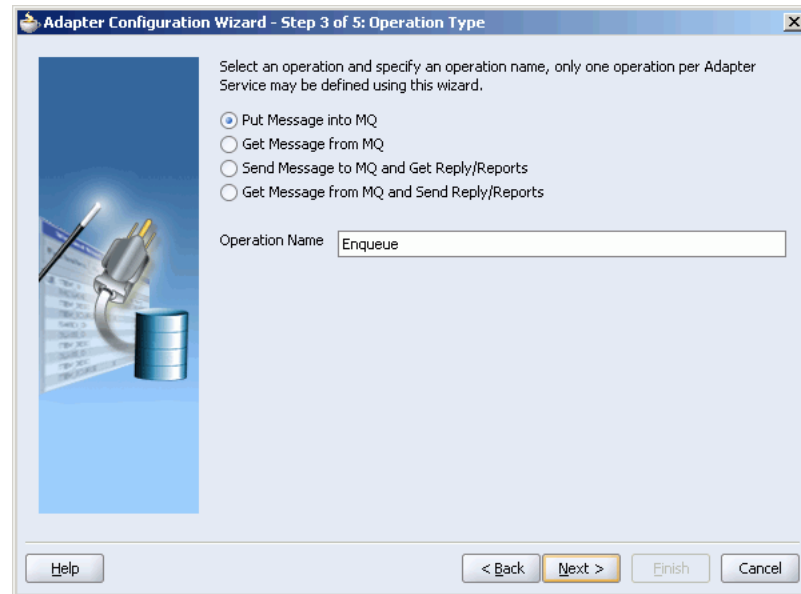
3. Sets the properties of the message such as, priority, expiry, message type, and persistence. These properties are based on the selections that you made in the Adapter Configuration wizard.

For information on message properties, see [Section 6.3.2.1, "Messages Types"](#).

4. Sends the message to the queue specified at design time in the Adapter Configuration wizard.

[Figure 6–5](#) displays the operation type that you need to select in the Adapter Configuration wizard window.

Figure 6–5 Adapter Configuration Wizard: Produce Message Selection



The window that appears after selecting the **Put Message into MQ** operation type is shown in [Figure 6–6](#).

Figure 6–6 Put Message Options

Adapter Configuration Wizard - Step 4 of 5: Put Message into MQ

Enter information for putting a normal message into MQ Series.

Queue Name:

Queue Manager (optional):

Message Format:

Priority:

Persistence:

Expiry:

☒ Never

☐ Expires in:

Help < Back Next > Finish Cancel

You can specify the following properties in this window:

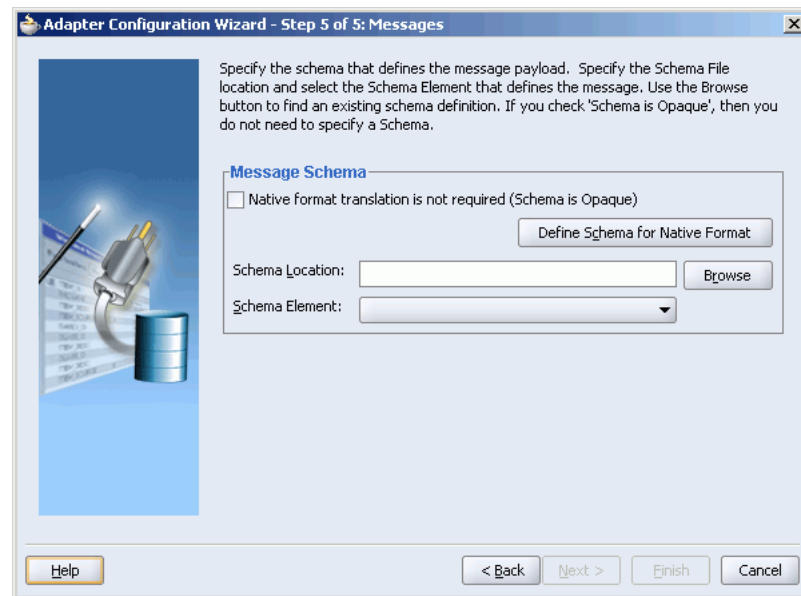
- Queue Name: The name of the queue on which the MQSeries adapter will enqueue the message. This is a mandatory field.
- Queue Manager: The name of the queue manager to which the queue belongs. This field is optional and should be used when enqueueing message to a remote queue.
- Message Format: The format of the message.

Note: When enqueueing a message, ensure that the various mandatory values, required for a specific format, are specified correctly.

- Priority: The priority of the message ranging from 0 (low) to 9 (high).
- Persistence: The persistence of the message. You can also specify the persistence of the message to be taken from the default priority attribute, as defined by the destination queue.
- Expiry: The expiry time of the message. The message is discarded after the expiry time has elapsed.

Refer to [Section 6.3.2, "Message Properties"](#) for detailed information about these properties.

The next Adapter Configuration Wizard window that appears is the Messages window shown in [Figure 6–7](#). This window enables you to select the XML Schema Definition (XSD) file for translation.

Figure 6–7 Messages Window

If native format translation is not required (for example, a JPG or GIF image is being processed), select the **Native format translation is not required** check box. The file is passed through in base-64 encoding.

XSD files are required for translation. If you want to define a new schema or convert an existing data type description (DTD) or COBOL Copybook, then select **Define Schema for Native Format**. This starts the Native Format Builder Wizard. This wizard guides you through the creation of a native schema file from file formats, such as comma-separated value (CSV), fixed-length, DTD, and COBOL Copybook. After the native schema file is created, you are returned to this Messages window with the **Schema File URL** and **Schema Element** fields filled in. See [Section 7.1, "Creating Native Schema Files with the Native Format Builder Wizard"](#) for more information.

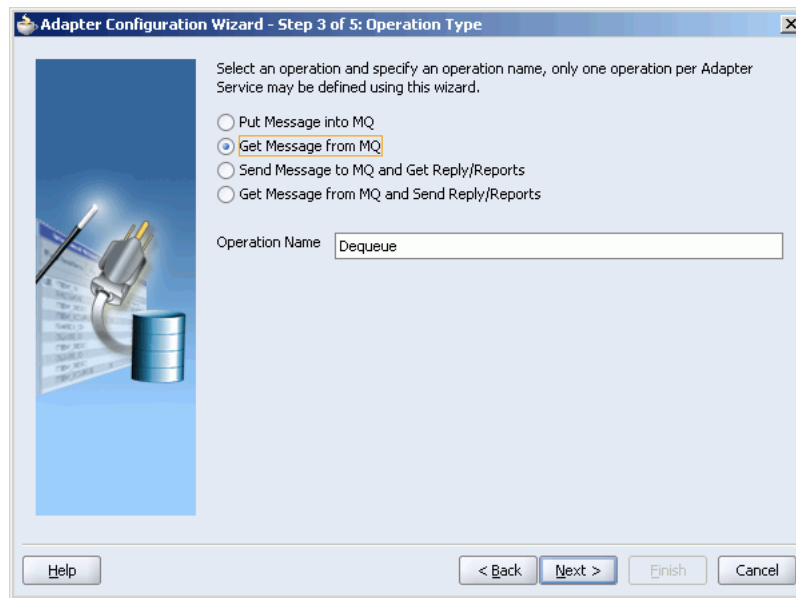
Note: Ensure that the schema you specify includes a namespace. If your schema does not have a namespace, an error message appears.

6.3.1.2 Dequeue Message

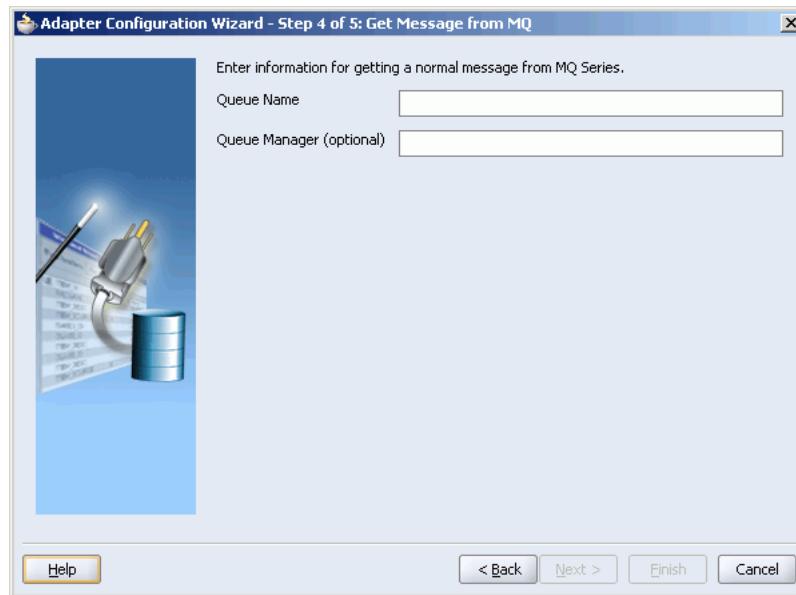
In this scenario, the MQSeries adapter connects to a specific queue managed by a queue manager and then removes the message from the queue. For inbound messages sent to Oracle BPEL Process Manager or Oracle Enterprise Service Bus, the MQSeries adapter performs the following operations:

1. Connects to the queue specified at the design time.
2. Dequeues the message from the queue when a message arrives.
3. Reads and translates the message based on the translation logic defined at design time.
4. Publishes the message as XML message to Oracle BPEL Process Manager or Oracle Enterprise Service Bus.

[Figure 6–8](#) displays the operation type that you need to select in the Adapter Configuration wizard.

Figure 6–8 Adapter Configuration Wizard: Consume Message Selection

The window that appears after selecting the **Get Message into MQ** operation type is shown in [Figure 6–6](#).

Figure 6–9 Get Message From MQ window

You can specify the following properties in this window:

- Queue Name: The name of the queue from which the MQSeries adapter will dequeue the message. This is a mandatory field.
- Queue Manager: The name of the queue manager to which the queue belongs. This field is optional.

The next Adapter Configuration Wizard window that appears is the Messages window shown in [Figure 6–7](#). This window enables you to select the XSD schema file for translation.

As with specifying the schema for the produce message operation, you can perform the following tasks in this window:

- Specify if native format translation is not required
- Select the XSD schema file for translation
- Start the Native Format Builder Wizard to create an XSD file from file formats such as CSV, fixed-length, DTD, and COBOL Copybook

See [Section 6.3.1.1, "Enqueue Message"](#) for more information about the Messages window.

6.3.1.3 Request-Response (Oracle BPEL Process Manager as a Client)

In this scenario, the Oracle BPEL Process Manager sends a request message and receives the corresponding response using a non-initiating receive activity. The invoke activity initiates the outbound invocation of the adapter to send the request. The MQSeries adapter performs the following operations:

1. Receives message from Oracle BPEL Process Manager.
2. Formats the XML content as specified at the design time in the Adapter Configuration wizard.
3. Sets properties and a correlation schema on the request message.
4. Sends the message to the queue specified at the design time. The third-party application receives the message, processes it, generates the response, and then enqueues the response message to the `replyTo` queue specified in the request message. The Correlation ID and Message ID of the response message is generated on the basis of the correlation schema specified in the request message.
5. The MQSeries adapter dequeues the message from the queue.
6. Sends the response to the non-initiating receive activity of the BPEL process. To ensure that response is sent to correct BPEL instance, correlation schemas are used.

[Figure 6–10](#) shows a sample BPEL process for this scenario.

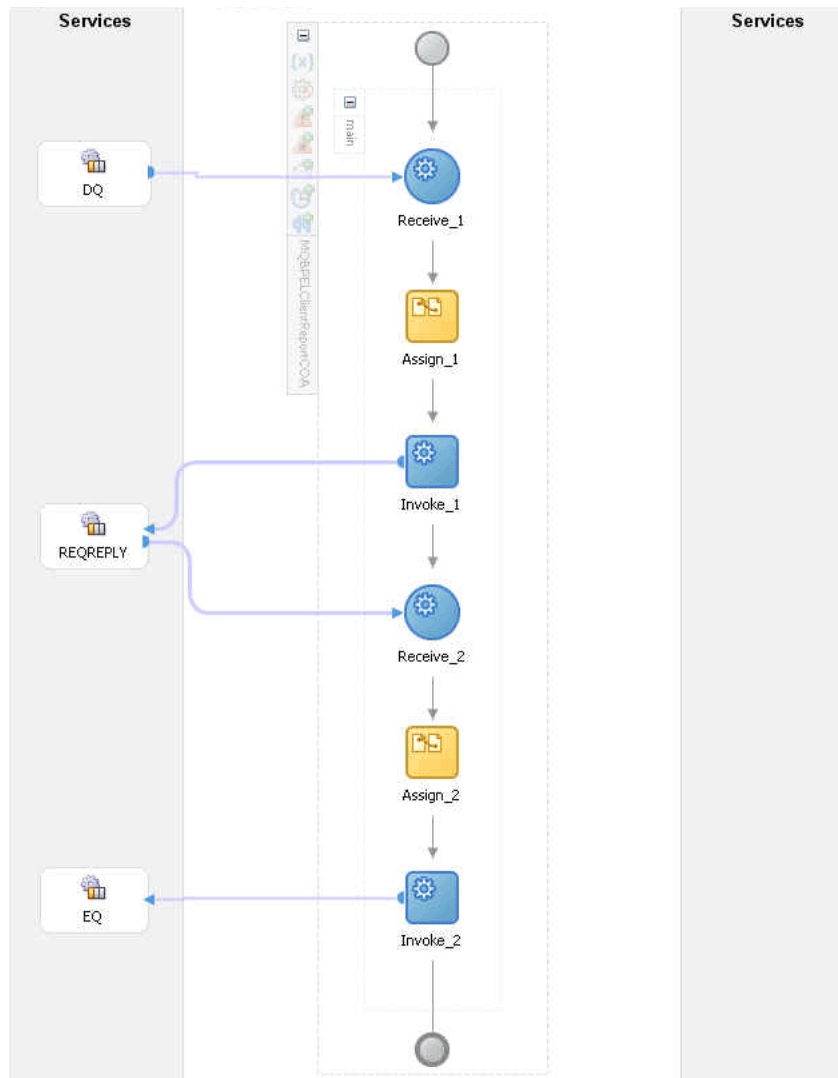
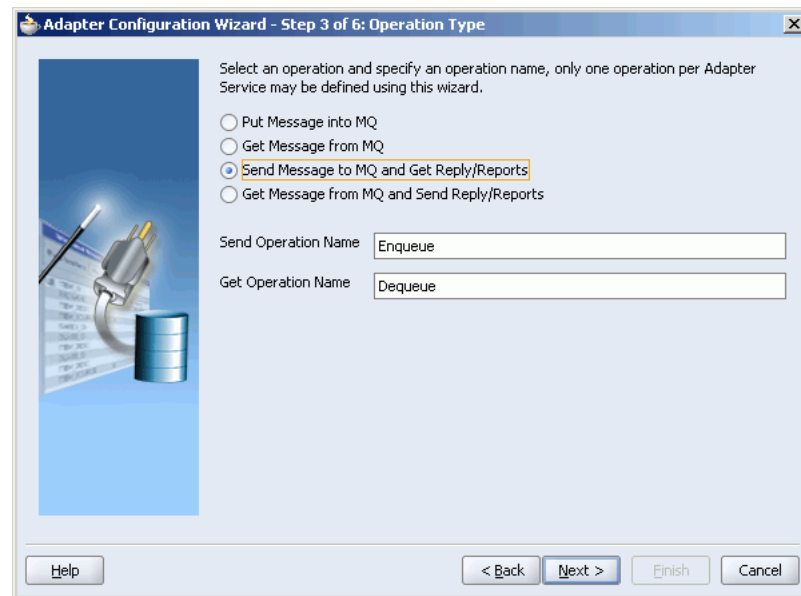
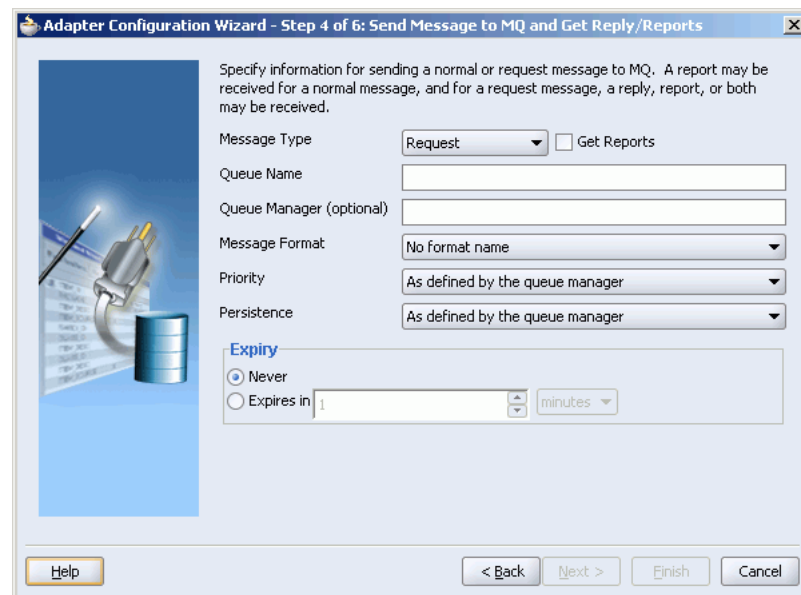
Figure 6–10 Oracle BPEL Process Manager as Client Sample

Figure 6–11 displays the operation type that you need to select in the Adapter Configuration wizard.

Figure 6–11 Selecting an Operation Type

The window that appears after selecting the **Send Message to MQ and Get Reply/Reports** operation type is shown in [Figure 6–12](#).

Figure 6–12 Send Message to MQ and Get Reply/Reports Window

You can specify the following properties in this window:

- **Message Type:** The type of the message. You can send a normal message or a request message.
- **Get Reports:** Select this option if you want any kind of report. You can specify the type of report in the next window shown in [Figure 6–13](#).
- **Queue Name:** The name of the queue to which the MQSeries adapter enqueues the message. This is a mandatory field.

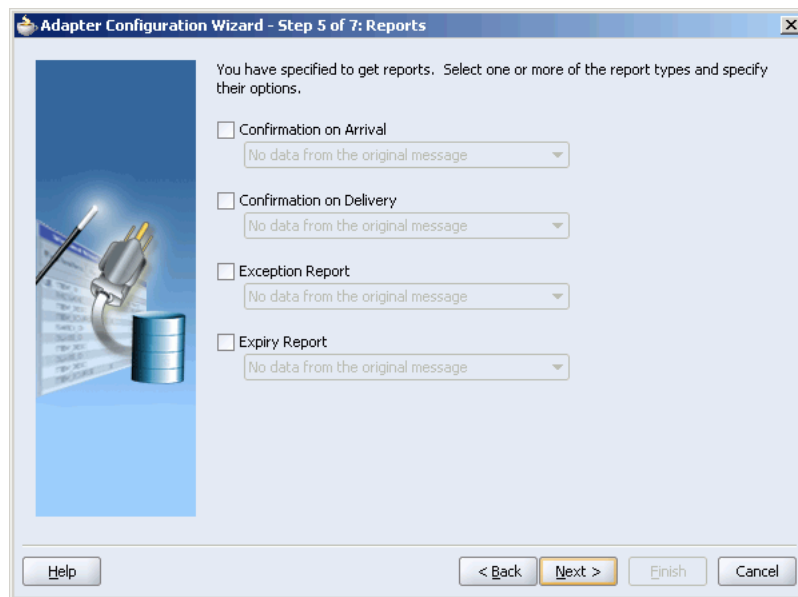
- Queue Manager: The name of the queue manager to which the queue belongs. This field is optional.
- Message Format: The format of the message.
- Priority: The priority of the message ranging from 0 (low) to 9 (high).
- Persistence: The persistence of the message.
- Expiry: The expiry time of the message. The message is discarded after the expiry time has elapsed.

Refer to [Section 6.3.2, "Message Properties"](#) and [Section 6.3.5, "Report Messages"](#) for more information about these properties.

The window that is displayed when you click **Next** in the Send Message to MQ and Get Reply/Reports window can be a Reports window (shown in [Figure 6-13](#)) or a Response window (shown in [Figure 6-13](#)).

The Reports window shown in [Figure 6-13](#) is displayed only if you have selected the **Get Reports** option in the Send Message to MQ and Get Reply/Reports window shown in [Figure 6-12](#).

Figure 6-13 Reports Window



You can select the following types of reports in this window:

- Confirmation on Arrival
- Confirmation on Delivery
- Exception Report
- Expiry Report

Refer to [Section 6.3.5, "Report Messages"](#) for information about these report types.

The Response window shown in [Figure 6-14](#) is displayed when you click **Next** in the Reports window.

Figure 6–14 Response Window

Adapter Configuration Wizard - Step 6 of 7: Response

Specify information about where the reply/report should be sent.

Message Type:

Queue Name:

Queue Manager (optional):

Correlation Scheme

Message ID:

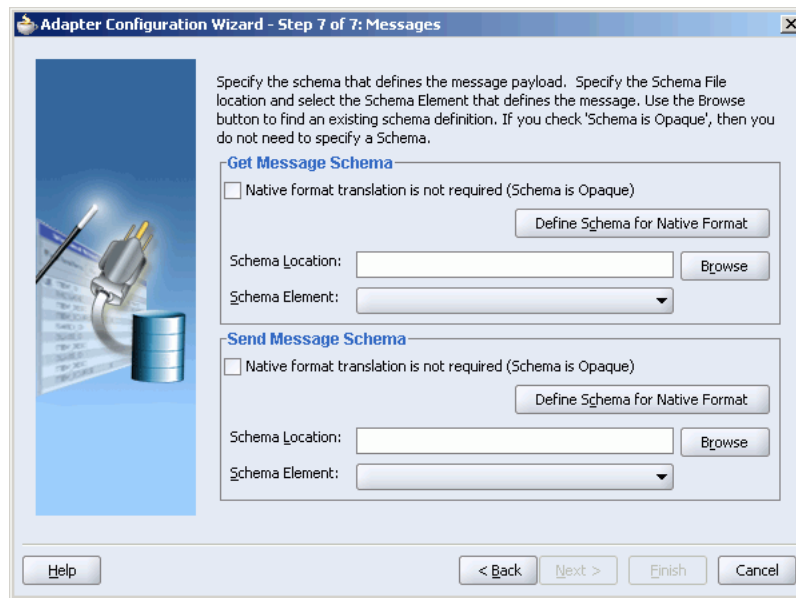
Correlation ID:

Help < Back Next > Finish Cancel

You can specify the following properties in the Response window:

- **Queue Name:** The name of the queue to which the response should be sent. This is a mandatory field.
- **Queue Manager:** The name of the queue manager to which the queue belongs. This field is optional.
- **Correlation Schema:** The correlation schema that should be used by the MQSeries adapter. For information about correlation schemas, refer to [Section 6.3.3, "Correlation Schemas"](#).

When you click **Next** in the Response window, a Messages window shown in [Figure 6–15](#) is displayed. This window enables you to select the XSD schema file for translation for request as well as response message.

Figure 6–15 Messages Window

You can perform the following tasks in this window:

- Specify if native format translation is not required
- Select the XSD schema file for translation
- Start the Native Format Builder Wizard to create an XSD file from file formats such as CSV, fixed-length, DTD, and COBOL Copybook

See [Section 6.3.1.1, "Enqueue Message"](#) for more information about the Messages window.

6.3.1.4 Synchronous Request-Response (Oracle BPEL Process Manager as Server)

In this scenario, the Oracle BPEL Process Manager receives a request, processes it, and sends the response synchronously by using a reply activity. The MQSeries adapter performs the following operations:

1. Dequeues the request message from the queue when the message arrives.
2. Reads and translates the message based on the translation logic defined at design time.
3. Publishes the message as XML message to Oracle BPEL Process Manager. The Oracle BPEL Process Manager processes the request and sends the response to the MQSeries adapter.
4. Receives the response message from the Oracle BPEL Process Manager.
5. Formats the XML content as specified at the design time.
6. Sets the properties of the message such as priority, expiry, message type, and persistence. These properties are based on the selections that you made in the Adapter Configuration wizard.
7. Sends the message to the queue specified at design time in the Adapter Configuration wizard.

[Figure 6–16](#) shows a sample BPEL process for this scenario.

Figure 6–16 Synchronous Request-Response Oracle BPEL Process Manager as Server sample

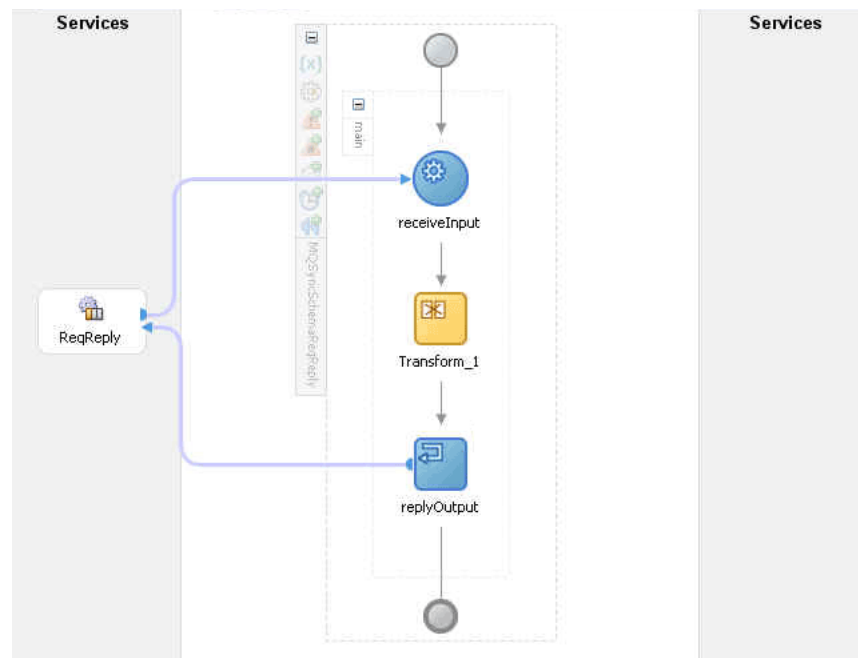
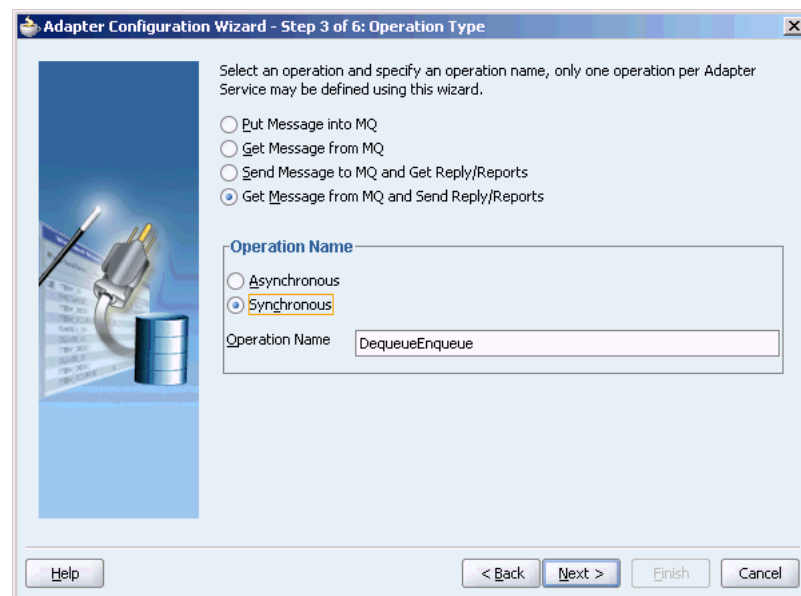


Figure 6–17 displays the operation type that you need to select in the Adapter Configuration wizard.

Figure 6–17 Operation Type Window Selection for Request-Response Synchronous Interaction



The window that appears after selecting the **Get Message from MQ and send Reply/Reports** operation type is shown in Figure 6–18. Specify the queue name from which the MQSeries adapter will dequeue the message in this window.

Figure 6–18 Get Message from MQ and Send Reply/Reports Window

Adapter Configuration Wizard - Step 4 of 6: Get Message from MQ and Send Reply/Reports

Specify information for getting a normal or request message from MQ, and sending a reply or report.

Message Type:

Queue Name:

Queue Manager (optional):

Buttons: Help, < Back, Next >, Finish, Cancel

When you click **Next** in the Get Message from MQ and send Reply /Reports window, the Response window shown in [Figure 6–19](#) is displayed.

Figure 6–19 Response Window for Synchronous Request-Response

Adapter Configuration Wizard - Step 5 of 6: Response

Specify information about the response message.

Message Type:

Response Fallback Queue Name:

Response Fallback Queue Manager:

Priority:

Persistence:

Expiry:

☒ Never

☐ Expires in

Buttons: Help, < Back, Next >, Finish, Cancel

You can specify the following properties in the Response window:

- **Response Fallback Queue Name:** The queue to which the response should be sent for a normal message.

Note: The Response fallback queue should be the primary queue of its queue manager.

- Response Fallback Queue Manager: The name of the queue manger to which the response fallback queue belongs.
- Priority: The priority of the message.
- Persistence: The persistence of the message.
- Expiry: The expiry time of the message.

Refer to [Section 6.3.2, "Message Properties"](#) for more information about these properties.

On clicking **Next** in the Response window, the Messages window shown in [Figure 6–15](#) is displayed. You can perform the following tasks in this window:

- Specify if native format translation is not required
- Select the XSD schema file for translation
- Start the Native Format Builder Wizard to create an XSD file from file formats such as CSV, fixed-length, DTD, and COBOL Copybook

See [Section 6.3.1.1, "Enqueue Message"](#) for more information about the Messages window.

6.3.1.5 Asynchronous Request-Response (Oracle BPEL Process Manager as Server)

In Oracle BPEL Process Manager initiated request-response interaction, a BPEL process receives a request as an inbound message, processes it, and then sends the response through an invoke activity. For asynchronous request-reply scenario, the MQSeries adapter performs the following operations:

1. Dequeues the message from the queue when a message arrives.
2. Reads and translates the message based on the translation logic defined at design time.
3. Publishes the message as XML message to Oracle BPEL Process Manager. The Oracle BPEL Process Manager processes the request and sends the response to the MQSeries adapter.
4. Receives messages from Oracle BPEL Process Manager.
5. Formats the XML content as specified at design time.
6. Sets the properties of the message, such as priority, expiry, message type, and persistence. These properties are based on the selections that you made in the Adapter Configuration wizard.
7. Sends the message to the queue specified at design time in the Adapter Configuration wizard.

[Figure 6–20](#) shows a sample BPEL process for this scenario.

Figure 6–20 Asynchronous Request-Response Oracle BPEL Process Manager as Server sample

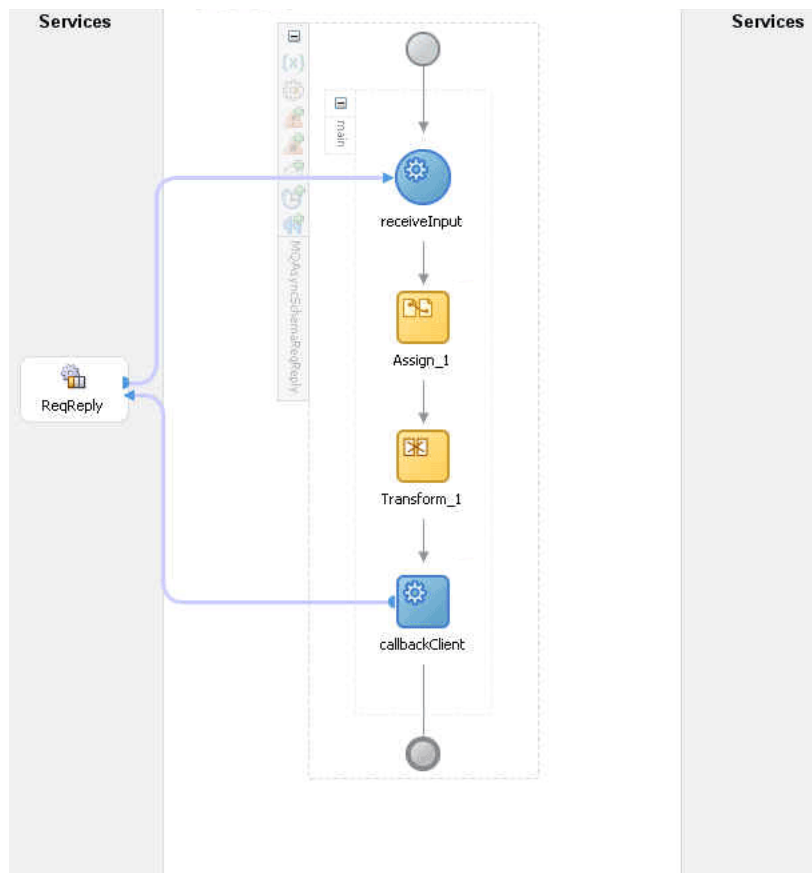
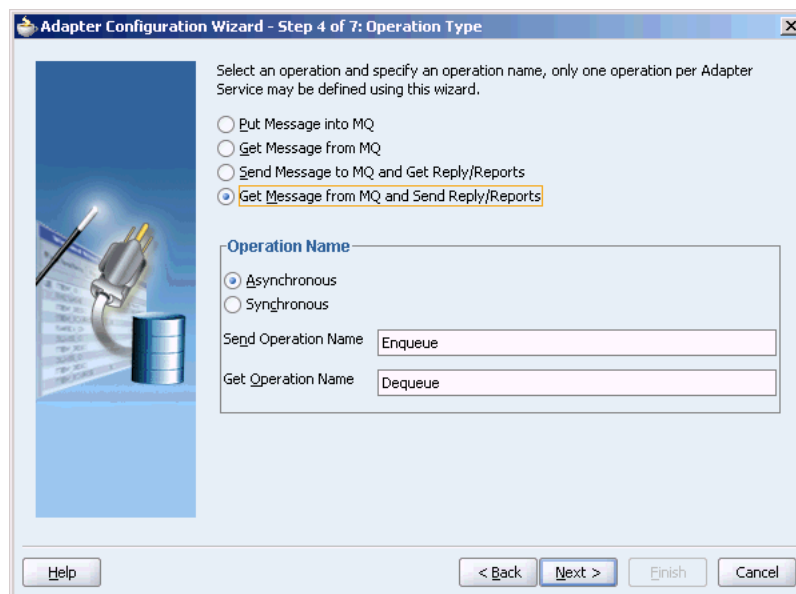


Figure 6–21 displays the operation type that you need to select in the Adapter Configuration wizard.

Figure 6–21 Operation Type Window Selection for Request-Response Asynchronous Interaction



The window that appears after selecting the **Get Message from MQ and send Reply/Reports** operation type is shown in [Figure 6–18](#). Specify the queue name from which the MQSeries adapter will dequeue the message in this window.

When you click **Next** in the Get Message from MQ and send Reply/Reports window, the Response window shown in [Figure 6–19](#) is displayed.

You can specify the following properties in the Response window:

- Response Fallback Queue Name: The queue to which the reply should be sent if the replyto queue is not specified in the request message.
- Response Fallback Queue Manager: The name of the queue manger to which the response fallback queue belongs.
- Priority: The priority of the message.
- Persistence: The persistence of the message.
- Expiry: The expiry time of the message.

Refer to [Section 6.3.2, "Message Properties"](#) for more information about these properties.

On clicking **Next** in the Response window, the Messages window shown in [Figure 6–15](#) is displayed. You can perform the following tasks in this window:

- Specify if native format translation is not required
- Select the XSD schema file for translation
- Start the Native Format Builder Wizard to create an XSD file from file formats such as CSV, fixed-length, DTD, and COBOL Copybook

See [Section 6.3.1.1, "Enqueue Message"](#) for more information about the Messages window.

In asynchronous request-reply interaction, you must map the following properties from the inbound message header to the outbound message header:

- MsgID
- CorrelID
- CorrelationScheme
- ReplyToQ

You can use the Assign activity to map these properties.

6.3.1.6 Request-Response Synchronous (Oracle Enterprise Service Bus as Server)

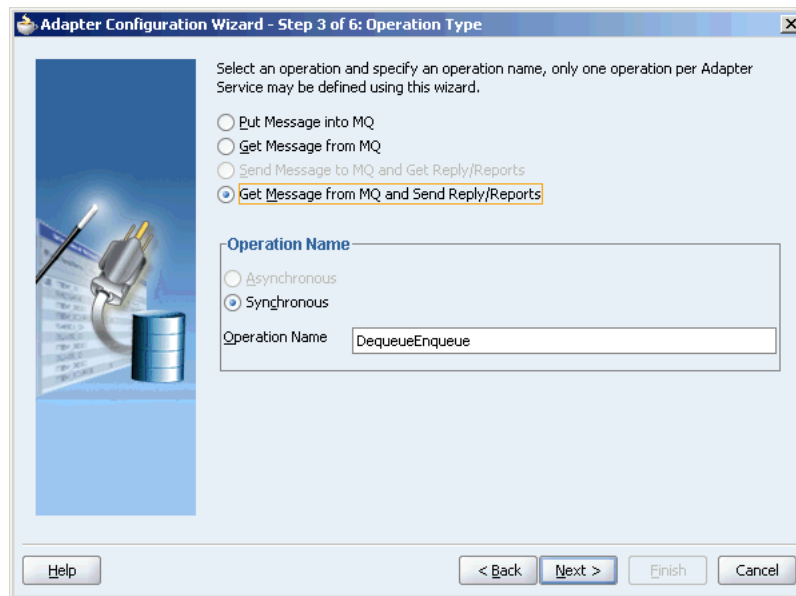
In this scenario, the Oracle Enterprise Service Bus receives a request, processes it, and sends the response synchronously. The MQSeries adapter performs the following operations:

1. Dequeues the request message from the queue when the message arrives.
2. Reads and translates the message based on the translation logic defined at design time.
3. Publishes the message as XML message to Oracle Enterprise Service Bus. The Oracle Enterprise Service Bus processes the request and sends the response to the MQSeries adapter.
4. Receives the response message from the Oracle Enterprise Service Bus.

5. Formats the XML content as specified at the design time.
6. Sets the properties of the message such as priority, expiry, message type, and persistence. These properties are based on the selections that you made in the Adapter Configuration wizard.
7. Sends the message to the queue specified at design time in the Adapter Configuration wizard.

Figure 6–22 displays the operation type that you must select in the Adapter Configuration wizard.

Figure 6–22 Oracle Enterprise Service Bus as Server-Request-Response Synchronous Interaction



From this window onwards, all the windows are similar to the windows explained in Section 6.3.1.4, "Synchronous Request-Response (Oracle BPEL Process Manager as Server)" on page 6-16.

6.3.2 Message Properties

The MQSeries adapter supports the following message properties:

- [Messages Types](#)
- [Message Format](#)
- [Message Expiry](#)
- [Message Priority](#)
- [Message Persistence](#)

6.3.2.1 Messages Types

The MQSeries adapter supports the following four types of messages:

- Normal Message

A normal message is sent by one program to another program without expecting any response.

- Request Message

A request message is sent by one program to another program requesting for a response.

- Reply Message

A reply message is sent by a program in response to a request message.

- Report Message

A report message is sent by receiving program to sending program as confirmation of successful or unsuccessful delivery of a message. A report message can be generated for any of the message types, normal message, request message, or reply message.

For information on acknowledgment messages supported by the MQSeries adapter, refer to [Section 6.3.5, "Report Messages"](#).

6.3.2.2 Message Format

You can specify the format of an for an outgoing message through Adapter Configuration wizard, as shown in [Figure 6-6](#). Following message formats are supported:

- No format name (Default)
- Command server request/reply message
- Type 1 command reply message
- Type 2 command reply message
- Dead-letter header
- Event message
- User-defined message in programmable command format
- Message consisting entirely of characters
- Trigger message
- Transmission queue header

6.3.2.3 Message Expiry

You can specify the expiry time for an outgoing message Adapter Configuration wizard shown in [Figure 6-6](#). The queue manager discards the message after the expiry time of a message has elapsed.

If a message has expiration notification set, then a notification is generated when the message is discarded. The notification is sent to the queue specified in the `replyToQueue` parameter. By default, `NEVER` is set for the expiry field.

6.3.2.4 Message Priority

You can specify the priority of an outgoing message through Adapter Configuration wizard, as shown in [Figure 6-6](#). A priority can be in the range of 0 (low) to 9 (high). You can also specify the priority of the message to be taken from the default priority attribute, as defined by the destination queue. By default, `AS_Q_DEF` is set as message priority.

6.3.2.5 Message Persistence

You can specify the persistence of an outgoing message through Adapter Configuration wizard shown in [Figure 6–6](#). If message persistence is not set, a message is lost when the queue manager restarts or there is a system failure. If you set persistence for a message to `true`, it means that the message will not be lost even if there is system failure or the queue manager is restarted. You can also specify the persistence of the message to be taken from the default priority attribute, as defined by the destination queue. Persistent messages are written to log files and queue data files. If a queue manager is restarted after a failure, it recovers these persistent messages from these files.

Note: You can specify all these message properties at run time through message headers. You can use the assign activity to assign values to these properties.

6.3.3 Correlation Schemas

Correlation is required for mapping a response to a request in a request-reply interaction. Each MQSeries request message contains a message ID and a correlation ID. When an application receives a request message from Oracle BPEL Process Manager, it checks for the correlation schema defined for the response message. Based on the correlation schema, the application generates the message ID and correlation ID of the response message.

The response window of the Adapter Configuration wizard shown in [Figure 6–23](#) enables you to specify the correlation schema for the response message.

Figure 6–23 *Selecting Correlation Schemas*

Adapter Configuration Wizard - Step 6 of 7: Response

Specify information about where the reply/report should be sent.

Message Type:

Queue Name:

Queue Manager (optional):

Correlation Scheme

Message ID:

Correlation ID:

Help < Back Next > Finish Cancel

The Message ID box shown in [Figure 6–23](#) provides the following options for the message ID of the response message:

- Generate a new message ID for the response message
- Use the message ID of the request message

Similarly, the Correlation ID box shown in [Figure 6-23](#) provides the following options for the correlation ID of the response message:

- Use the message ID of the request message
- Use the correlation ID of the request message

6.3.4 Distribution List Support

The MQSeries adapter enables you to enqueue a message to multiple queues. You can use the `DistributionList` parameter of the adapter WSDL file for this. For example:

```
DistributionList="Queue1Name,QueueManger1Name; Queue2Name,QueueManger2Name"
```

The queue name and queue manager name value pairs should be separated by semi colon (;). The queue manager name is optional.

6.3.5 Report Messages

The MQSeries adapter enables you to set various types of acknowledgement messages on an outgoing message. These acknowledgement messages are known as report messages. A report message is generated only if the criteria for generating that report message is met. When enqueueing a message on a queue, you can request for more than one type of report message. When you request for a report message, you must specify the queue name to which the report message will be sent. This queue is known as `replyTo` queue. A report message can be generated by a queue manager, a message channel, or an application.

The MQSeries adapter supports the following message reports:

- Confirmation on Arrival

The Confirmation on Arrival (COA) message indicates that the message has been delivered to the target queue manager. A COA message is generated by the queue manager. This message report can be selected in the Reports window of the Adapter Configuration window shown in [Figure 6-13](#).

- Confirmation on Delivery

A Confirmation on Delivery (COD) message indicates that the message has been retrieved by the receiving application. A COD message is generated by the queue manager. This message report can be selected in the Reports window shown in [Figure 6-13](#).

- Exception Report

An exception report is generated when a message cannot be delivered to the specified destination queue. Exception reports are generated by the message channel. This message report can be selected in the Reports window of the Adapter Configuration window shown in [Figure 6-13](#).

- Expiry Report

An expiry report indicates that the message was discarded because the expiry time specified for the message elapsed before the message could be delivered. An expiry report is generated by a queue manager. This message report can be selected in the Reports window of the Adapter Configuration window shown in [Figure 6-13](#).

- Positive Action Notification

A Positive Action Notification (PAN) indicates that a request has been successfully processed. It means that the action requested in the message has been performed successfully. This type of report is generated by the application.

- Negative Action Notification

A Negative Action Notification (NAN) indicates that a request has not been successfully serviced. It means that the action requested in the message has not been performed successfully. This type of report is generated by the application.

You can specify whether all these report messages except PAN and NAN should contain the complete original message, a part of the original message, or no part of the original message. You can select one of the following options in the Adapter Configuration wizard:

- No data from the original message
- The first 100 bytes of data in original message
- The entire original message

6.3.6 Filter-by Criteria

The MQSeries adapter enables you to filter the messages by priority in the inbound interaction. When you specify a priority in the `FilterByPriority` parameter, the MQSeries adapter checks the queue for the messages that have same priority, as specified in the `FilterByPriority` parameter. The following example shows how to set the `FilterByPriority` parameter:

```
FilterByPriority= "2"
```

6.3.7 Message Delivery Failure Options

The MQSeries adapter enables you to specify the action that should be taken in case a message could not be delivered to the destination queue. You can specify one of the following actions:

- Place message on a dead-letter queue

This is the default action. Message is placed on a dead-letter queue if it cannot be delivered to the destination queue. A report message is generated if requested by the sender. Specify `DeadLetterQueue` as value of the `OnDeliverFailure` parameter in the adapter WSDL file as shown in the following example:

```
OnDeliverFailure="DeadLetterQueue"
```

- Discard message

This indicates that the message should be discarded if it cannot be delivered to the destination queue. A report message is generated, if requested by the sender. Specify `Discard` as value of the `OnDeliverFailure` parameter in the adapter WSDL file as shown in the following example:

```
OnDeliverFailure="Discard"
```

6.3.8 Message Segmentation

The MQSeries adapter supports message segmentation for both inbound and outbound interactions. Segmentation is required when size of a message is greater than the message size specified in the queue manager. A physical message is divided

into two or more logical messages. All logical messages have same group ID and a sequence number, and an offset.

In the inbound interaction, the segmentation is inherently supported by the MQSeries adapter. The MQSeries adapter dequeues all logical messages in the order of sequence number and then publishes the single message as XML to Oracle BPEL Process Manager or Oracle Enterprise Service Bus.

In the outbound interaction, you can set the `SegmentIfRequired` parameter to segment the outgoing message, if required. For example:

```
SegmentIfRequired="true"
```

The message will be segmented based on constraints such as, if the size of the message is larger than the maximum limit set on the queue.

6.3.9 Message Grouping

The MQSeries adapter supports message grouping for both inbound and outbound interactions. Messages belonging to a group can be complete messages or logical segments of a physical message.

In the inbound interaction, grouping is inherently supported by the MQSeries adapter. The MQSeries adapter dequeues all the messages that belong to same group or that have same group ID in the order of sequence number and publishes the single message as XML to Oracle BPEL Process Manager or Oracle Enterprise Service Bus.

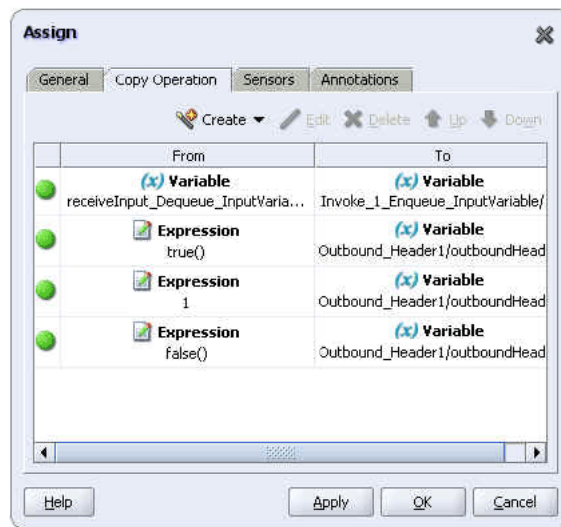
Note: Message grouping is not supported when the inbound messages have an opaque schema.

In the outbound interaction, grouping is supported through headers. You have to generate same group ID for all messages and also specify the sequence number. The message in the group should be marked with `last-msg-in-group` flag. In Oracle BPEL Process Manager, you can use the assign activity to set these properties. For example, you want to send three messages as part of the same group:

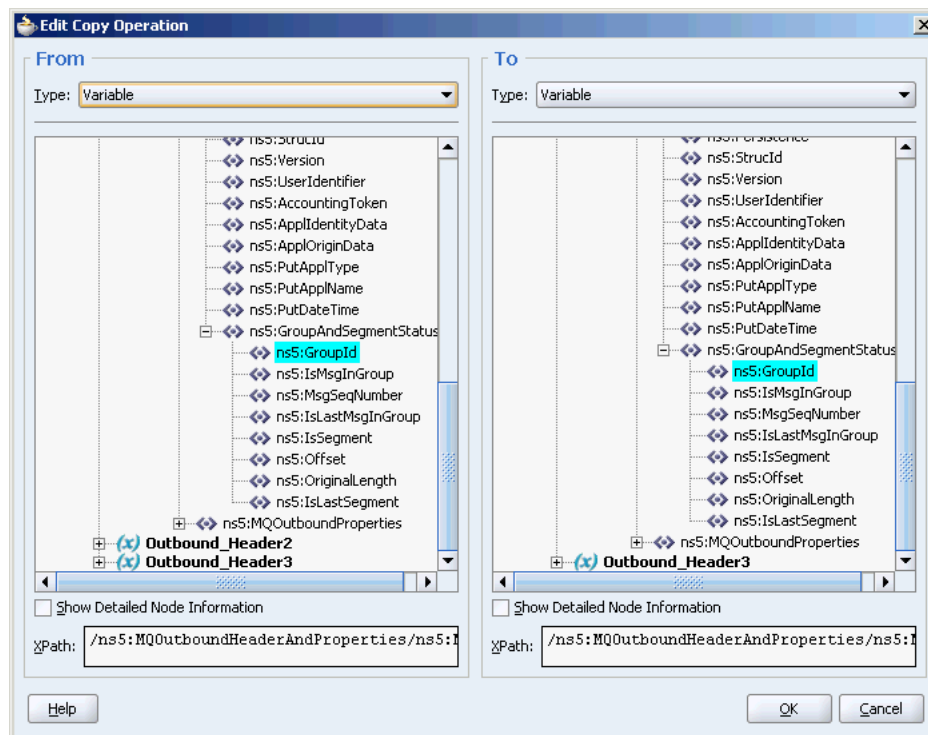
For the first message, set the following properties through Assign activity:

- `IsMsgInGroup`: This property should be set to `true`.
- `GroupId`: A group id can be generated by the queue manager or you can specify your own group id. If you want to the queue manager to generate the group id, set the `GroupId` as `GENERATE_GROUP_ID`.
- `MsgSeqNumber`: This property should be set to 1.
- `IsLastMsgInGroup`: This property should be set to `false`.

Figure 6–24 shows the assign activity dialog box:

Figure 6–24 Assign Activity Box

For the second message, first copy the `GroupId` of the first message from the message header to the second message header, as shown in [Figure 6–25](#).

Figure 6–25 Copy Message GroupIDs

Next, set the following properties through Assign activity:

- `IsMsgInGroup`: This property should be set to `true`.
- `MsgSeqNumber`: This property should be set to 2.
- `IsLastMsgInGroup`: This property should be set to `false`.

For the last message, you again need to assign the Groupid similar to the first and second message and then specify the following properties:

- IsMsgInGroup: This property should be set to true.
- MsgSeqNumber: This property should be set to 3.
- IsLastMsgInGroup: This property should be set to true.

6.3.10 Integration with CICS

The MQSeries adapter provides support for sending and receiving messages from CICS server. In the inbound direction, an inbound message from CICS server is dequeued in same way as a normal message. In outbound direction, the message should be in the CICS format. A sample schema file for outbound CICS message format is shown in the following example:

```
<?xml version="1.0" ?>

<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://xmlns.oracle.com/pcbpel/nxsd/cics_mqcih"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"

  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  nxsd:version="NXSD"

  nxsd:encoding="UTF8"
  nxsd:stream="bytes"
  nxsd:byteOrder="bigEndian"

  xmlns:nxsd_extn="http://xmlns.oracle.com/pcbpel/nxsd/extensions"

  <element name="MSGForMQCICSBridge">
    <complexType>
      <sequence>
        <element name="MQCIH">
          <complexType>
            <sequence>
              <!--
                MQCHAR4    StrucId;
                Structure identifier
              -->
              <element name="StrucId" type="string"
                nxsd:style="fixedLength" nxsd:length="4" nxsd:padStyle="tail"/>

              <!--
                MQLONG     Version;
                Structure version number 1 or 2
              -->
              <element name="Version" type="string"
                nxsd:style="integer" nxsd_extn:octet="4"
                nxsd_extn:align="0" nxsd_extn:sign="unticked" />
              <!--
                MQLONG     StrucLength;
                Length of MQCIH structure V1=164 V2=180
              -->
              <element name="StrucLength" type="string"
                nxsd:style="integer" nxsd_extn:octet="4"
                nxsd_extn:align="0" nxsd_extn:sign="unticked" />
            </sequence>
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>
```

```
<!--
MQLONG    Encoding;
Reserved
-->
<element name="Encoding" type="string"
  nxsd:style="integer" nxsd_extn:octet="4"
  nxsd_extn:align="0" nxsd_extn:sign="unticked" />

<!--
MQLONG    CodedCharSetId;
Reserved
-->
<element name="CodedCharSetId" type="string"
  nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
  nxsd_extn:sign="unticked" />

<!--
MQCHAR8    Format;
MQ Format name
-->
<element name="Format" type="string"
  nxsd:style="fixedLength" nxsd:length="8" />

<!--
MQLONG    Flags;
Reserved
-->
<element name="Flags" type="string"
  nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"

  nxsd_extn:sign="unticked" />

<!--
MQLONG    ReturnCode;
Return code from bridge
-->
<element name="ReturnCode" type="string"
  nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
  nxsd_extn:sign="unticked" />

<!--
MQLONG    CompCode;
MQ completion code or CICS EIBRESP
-->
<element name="CompCode" type="string"
  nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
  nxsd_extn:sign="unticked" />

<!--
MQLONG    Reason;
MQ reason or feedback code, or CICS EIBRESP2
-->
<element name="Reason" type="string"
  nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
  nxsd_extn:sign="unticked" />

<!--
MQLONG    UOWControl;
Unit-of-work control
-->
```

```

<element name="UOWControl" type="string"
  nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
  nxsd_extn:sign="unticked" />

<!--
MQLONG   GetWaitInterval;
Wait interval for MQGET call issued by bridge
-->
<element name="GetWaitInterval" type="string"
  nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
  nxsd_extn:sign="ticked" />

<!--
MQLONG   LinkType;
Link type
-->
<element name="LinkType" type="string"
  nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
  nxsd_extn:sign="unticked" />

<!--
MQLONG   OutputDataLength;
Output commarea data length
-->
<element name="OutputDataLength" type="string"
  nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
  nxsd_extn:sign="ticked" />

<!--
MQLONG   FacilityKeepTime;
Bridge facility release time
-->
<element name="FacilityKeepTime" type="string"
  nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
  nxsd_extn:sign="unticked" />

<!--
MQLONG   ADSDescriptor;
Send/receive ADS descriptor
-->
<element name="ADSDescriptor" type="string"
  nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
  nxsd_extn:sign="unticked" />

<!--
MQLONG   ConversationalTask;
Whether task can be conversational
-->
<element name="ConversationalTask" type="string"
  nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
  nxsd_extn:sign="unticked" />

<!--
MQLONG   TaskEndStatus;
Status at end of task
-->
<element name="TaskEndStatus" type="string"
  nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
  nxsd_extn:sign="unticked" />

```

```
<!--
MQBYTE Facility[8];
BVT token value. Initialise as required.
-->
<element name="Facility" type="string"
  nxsd:style="integer" nxsd_extn:octet="8" nxsd_extn:align="0"
  nxsd_extn:sign="unticked" />

<!--
MQCHAR4 Function;
MQ call name or CICS EIBFN function name
-->
<element name="Function" type="string"
  nxsd:style="fixedLength" nxsd:length="4" />

<!--
MQCHAR4 AbendCode;
Abend code
-->
<element name="AbendCode" type="string"
  nxsd:style="fixedLength" nxsd:length="4" />

<!--
MQCHAR8 Authenticator;
Password or passticket
-->
<element name="Authenticator" type="string"
  nxsd:style="fixedLength" nxsd:length="8" />

<!--
MQCHAR8 Reserved1;
Reserved
-->
<element name="Reserved1" type="string"
  nxsd:style="fixedLength" nxsd:length="8" />

<!--
MQCHAR8 ReplyToFormat;
MQ format name of reply message
-->
<element name="ReplyToFormat" type="string"
  nxsd:style="fixedLength" nxsd:length="8" />

<!--
MQCHAR4 RemoteSysId;
Remote sysid to use
-->
<element name="RemoteSysId" type="string"
  nxsd:style="fixedLength" nxsd:length="4" />

<!--
MQCHAR4 RemoteTransId;
Remote transid to attach
-->
<element name="RemoteTransId" type="string"
  nxsd:style="fixedLength" nxsd:length="4" />

<!--
MQCHAR4 TransactionId;
```

```

Transaction to attach
-->
<element name="TransactionId" type="string"
  nxsd:style="fixedLength" nxsd:length="4" />

<!--
MQCHAR4 FacilityLike;
Terminal emulated attributes
-->
<element name="FacilityLike" type="string"
  nxsd:style="fixedLength" nxsd:length="4" />

<!--
MQCHAR4 AttentionId;
AID key
-->
<element name="AttentionId" type="string"
  nxsd:style="fixedLength" nxsd:length="4" />

<!--
MQCHAR4 StartCode;
Transaction start code
-->
<element name="StartCode" type="string"
  nxsd:style="fixedLength" nxsd:length="4" />

<!--
MQCHAR4 CancelCode;
Abend transaction code
-->
<element name="CancelCode" type="string"
  nxsd:style="fixedLength" nxsd:length="4" />

<!--
MQCHAR4 NextTransactionId;
Next transaction to attach
-->
<element name="NextTransactionId" type="string"
  nxsd:style="fixedLength" nxsd:length="4" />

<!--
MQCHAR8 Reserved2;
Reserved
-->
<element name="Reserved2" type="string"
  nxsd:style="fixedLength" nxsd:length="8" />
<!--
MQCHAR8 Reserved3;
Reserved
-->
<element name="Reserved3" type="string"
  nxsd:style="fixedLength" nxsd:length="8" />

<!--
MQLONG CursorPosition;
Cursor position
-->
<element name="CursorPosition" type="string"
  nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
  nxsd_extn:sign="unticked" />

```

```

        <!--
        MQLONG    ErrorOffset;
        Error offset
        -->
        <element name="ErrorOffset" type="string"
            nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
            nxsd_extn:sign="unticked" />

        <!--
        MQLONG    InputItem;
        Input item
        -->
        <element name="InputItem" type="string"
            nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
            nxsd_extn:sign="unticked" />

        <!--
        MQLONG    Reserved4;
        Reserved
        -->
        <element name="Reserved4" type="string"
            nxsd:style="integer" nxsd_extn:octet="4" nxsd_extn:align="0"
            nxsd_extn:sign="unticked" />
    </sequence>
</complexType>
</element>

    <!--
    Application data
    -->
    <element name="ApplicationData" type="string"
        fixed="Nothing" />

</sequence>
</complexType>
</element>
</schema>

```

6.4 Configuring the MQSeries Adapter

The setup prerequisite for using the MQSeries adapter are:

- IBM Websphere MQ server should be installed and running.
- A queue manager, queue, and channel should be created.

To configure the MQSeries adapter, perform the following:

- [Add the com.ibm.mq.jar to the MQSeries Adapter Classpath](#)
- [Modify the oc4j-ra.xml File](#)

6.4.1 Add the com.ibm.mq.jar to the MQSeries Adapter Classpath

The steps in this section should be performed only once, before using the MQSeries adapter. Perform the following steps to add the `com.ibm.mq.jar` to the classpath for the MQSeries adapter:

1. Copy the `com.ibm.mq.jar` file to the any folder.

2. Create a new shared library in the `server.xml` file by specifying the path of the `com.ibm.mq.jar` as shown in the following example:

```
<shared-library name="oracle.mqseries" version="10.1.3">
  <code-source path="C:\ORAHOME\bpel/lib/com.ibm.mq.jar"/>
</shared-library>
```

The location of the `server.xml` file depends on your installation type. [Table 6–3](#) lists the installation type and corresponding `server.xml` file location.

Table 6–3 *server.xml File Location*

Installation Type	File Location
SOA Basic Installation	<i>Oracle_Home\j2ee\home\config</i>
Oracle Enterprise Service Bus on Oracle Application Server Middle tier	<i>Oracle_Home\j2ee\homemid\config</i>
Oracle BPEL Process Manager on Oracle Application Server Middle tier	<i>Oracle_Home\j2ee\homemid\config</i>

3. Modify the `oc4j-ra.xml` file to include the new shared library as shown in the following example.

```
<imported-shared-libraries>
  <import-shared-library name="oracle.bpel.common"/>
  <import-shared-library name="oracle.xml"/>
  <import-shared-library name="oracle.mqseries"/>
</imported-shared-libraries>
```

The location of the `oc4j-ra.xml` file depends on your installation type. [Table 6–4](#) lists the installation type and corresponding `oc4j-ra.xml` file location.

Table 6–4 *oc4j-ra.xml File Location*

Installation Type	File Location
SOA Basic Installation	<i>Oracle_Home\j2ee\home\application-deployments\default\MQSeriesAdapter</i>
Oracle Enterprise Service Bus on Oracle Application Server Middle tier	<i>Oracle_Home\j2ee\homemid\application-deployments\default\MQSeriesAdapter</i>
Oracle BPEL Process Manager on Oracle Application Server Middle tier	<i>Oracle_Home\j2ee\homemid\application-deployments\default\MQSeriesAdapter</i>

4. Restart the server.

6.4.2 Modify the `oc4j-ra.xml` File

Specify the value of the following parameters in the `oc4j-ra.xml` file:

- `hostName`: The name of the computer on which the IBM Websphere MQ server is running.
- `portNumber`: The port number for connecting to the IBM Websphere MQ server.

- `queueManagerName`: The name of the primary queue manager.
- `channelName`: The name of channel.
- `userID`: The user ID if connecting to IBM Websphere MQ server. running on a remote location.
- `password`: The password corresponding to the user ID.
- `clientEncoding`: This parameter is required if you are encoding the message header.
- `hostOSType`: This parameter should be specified if the Websphere server is running on a zSeries/Operating System (z/OS). The value should be `zos`.

Note: Properties of the MQSeries queues should be set as specified in the following list:

- The shareability property of MQSeries queues should be set to `Shareable`.
 - Index type of the MQSeries queues should be set to `MQIT_MSG_ID`.
-

A sample `oc4j-ra.xml` file is shown in the following example:

```
<connector-factory location="eis/MQ/MQSeriesAdapter"
connector-name="MQSeriesAdapter">
  <config-property name="hostName" value="localhost"/>
  <config-property name="portNumber" value="1414"/>
  <config-property name="queueManagerName" value="QM"/>
  <config-property name="channelName" value="MYCHANNEL"/>
  <config-property name="userID" value=""/>
  <config-property name="password" value=""/>
  <config-property name="clientEncoding" value=""/>
  <config-property name="hostOSType" value=""/>
  <connection-pooling use="private">
    <property name="waitTimeout" value="300" />
    <property name="scheme" value="fixed_wait" />
    <property name="maxConnections" value="50" />
    <property name="minConnections" value="0" />
  </connection-pooling>
  <security-config use="none">
  </security-config>
</connector-factory>
```

6.5 MQSeries Adapter **Use Cases** for Oracle BPEL Process Manager

This section contains the following use cases:

- [Section 6.5.1, "Message Enqueue/Dequeue"](#)
- [Section 6.5.2, "Synchronous Request-Response"](#)
- [Section 6.5.3, "Demonstrations and Samples"](#)

6.5.1 Message Enqueue/Dequeue

In this use case, the inbound MQSeries adapter dequeues the message from MQSeries queue `test_in` and publishes it to the BPEL process. A BPEL receive activity consumes the incoming MQSeries message and the same message is sent to

the MQSeries queue `test_out` through a BPEL activity. This use case consists of following sections:

- [Section 6.5.1.1, "Prerequisites"](#)
- [Section 6.5.1.2, "Creating the Inbound Adapter Service"](#)
- [Section 6.5.1.3, "Creating the Outbound Adapter Service"](#)
- [Section 6.5.1.4, "Creating the Receive Activity"](#)
- [Section 6.5.1.5, "Creating the Invoke Activity"](#)
- [Section 6.5.1.6, "Creating the Assign Activity"](#)
- [Section 6.5.1.7, "Run-Time"](#)

6.5.1.1 Prerequisites

This example assumes that you are familiar with basic BPEL constructs, such as activities and partner links, and Oracle JDeveloper environment for creating and deploying BPEL Process.

The MQSeries adapter must be configured as specified in [Section 6.4, "Configuring the MQSeries Adapter"](#) and two queue `test_in` and `test_out` should be created.

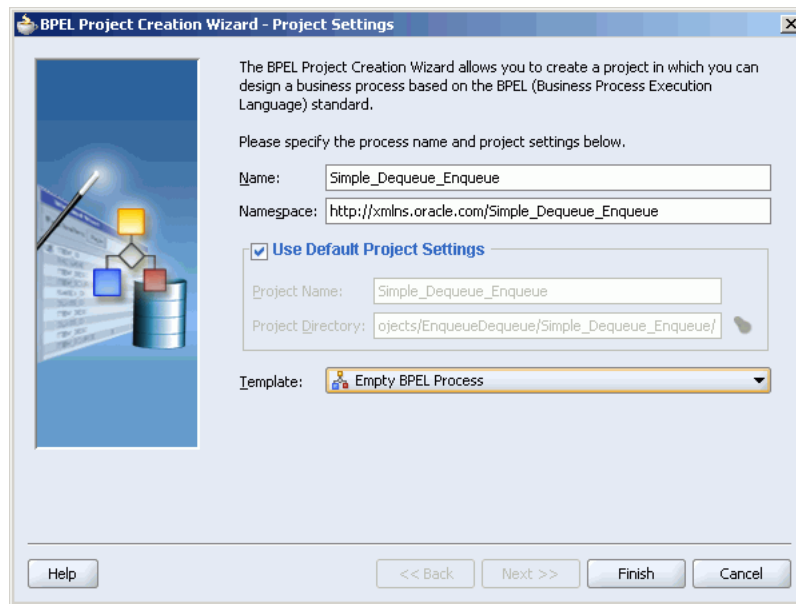
To define the schema of the messages, you require the `address-csv.xsd` file. This file can be copied from the following location:

```
ORAHOME\bpel\samples\tutorials\121.FileAdapter\FlatStructure
```

6.5.1.2 Creating the Inbound Adapter Service

Perform the following steps in Oracle JDeveloper to create an adapter service that will enqueue the message to a queue.

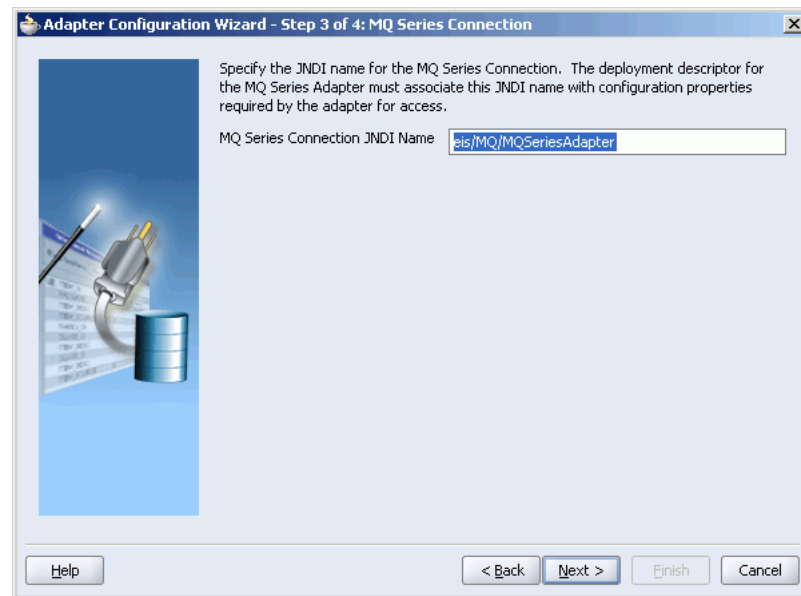
1. In the Application Navigator, right-click **Applications** and select **New Application**. The Create Application dialog box is displayed.
2. Enter `EnqueueDequeue` in the **Application Name** field and click **OK**. The Create Project dialog box is displayed.
3. Click **Cancel**.
4. In the Application Navigator, right-click **EnqueueDequeue** and select **New Project**. The New Gallery dialog box is displayed.
5. From Categories, select **General**, and then **Projects**.
6. From Items, select **BPEL Process Project** and click **OK**. The BPEL Process Creation Wizard-Project Settings dialog box is displayed.
7. Perform the following as shown in [Figure 6-26](#).
 - a. Enter `Simple_Dequeue_Enqueue` in the **Name** field.
 - b. Select **Empty BPEL Process** from the Templatelist.
 - c. Click **Finish**.

Figure 6–26 BPEL Process Creation Wizard-Project Settings Dialog Box

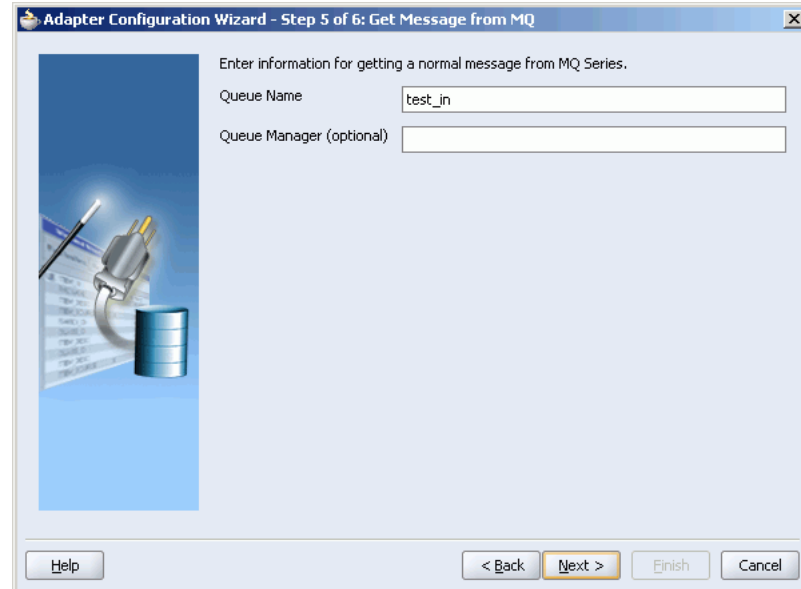
8. Create a Schema folder and copy the address-csv.xsd file to this folder.
9. Move the Schema folder to bpe1 sub directory of Simple_Dequeue_Enqueue project directory.
10. Drag a **Partner Link** activity from the Component Palette to the design area. The Create Partner Link dialog box is displayed.
11. Enter Inbound in the **Name** field and click the Define Adapter Service icon shown in [Figure 6–27](#).

Figure 6–27 Adapter Configuration icon

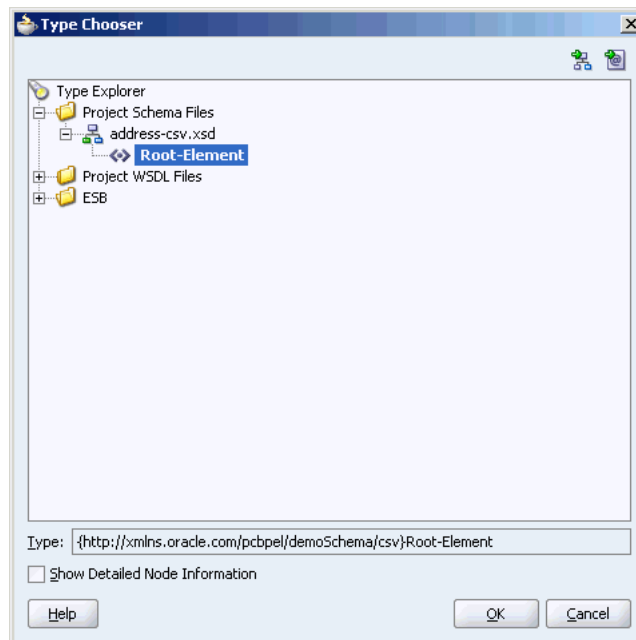
12. Click **Next** in the Welcome window of the Adapter Configuration Wizard. The Adapter Type window is displayed.
13. Select **MQ Adapter** and click **Next**. The Service Name window is displayed.
14. Enter inboundService in the Service Name field and click **Next**. The MQ Series Connection window is displayed.
15. Specify the JNDI name for the run-time connection, as shown in [Figure 6–28](#) and click **Next**.

Figure 6–28 MQ Series Connection Window

16. Select **Get Message from MQ** in the Operation Type window and click **Next**.
17. Enter `test_in` in the **Queue Name** field as shown in [Figure 6–29](#) and click **Next**.
The MQSeries adapter will dequeue the message from this queue.

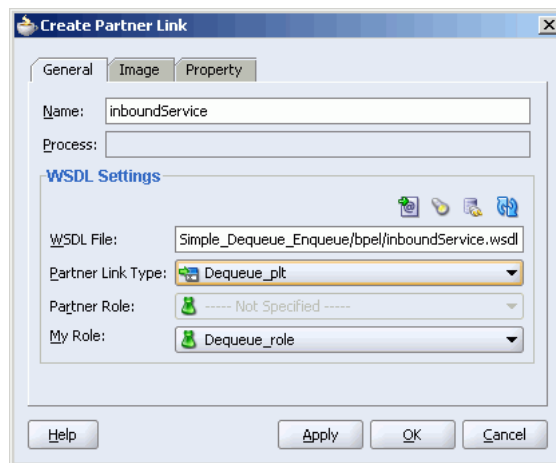
Figure 6–29 Get Message from MQ window

18. Click **Browse** in the Messages window. The Type Chooser dialog box is displayed.
19. Select **Project Schema Files**, `address-csv.xsd` and then **Root-Element**, as shown in [Figure 6–30](#) and click **OK**.

Figure 6–30 Type Chooser Dialog Box

20. Click **Next** in the Messages window.

21. Click **Finish**. The Create Partner Link dialog box appears, as shown in [Figure 6–31](#).

Figure 6–31 inboundService Create Partner Link dialog box

22. Click **OK**.

6.5.1.3 Creating the Outbound Adapter Service

Perform the following steps to create an adapter service that will dequeue the message from a queue.

1. Drag a **Partner Link** activity from the Component Palette to the design area. The Create Partner Link dialog box is displayed.
2. Enter **Outbound** in the **Name** field and click the Define Adapter Service icon shown in [Figure 6–27](#).

3. Click **Next** in the Welcome window of the Adapter Configuration Wizard. the Adapter Type window is displayed.
4. Select **MQ Adapter** and click **Next**. The Service Name window is displayed.
5. Enter `outboundService` in the **Service Name** field and click **Next**. The MQ Series Connection window is displayed.
6. Specify the JNDI name for the run-time connection as shown in [Figure 6-28](#) and click **Next**. The Operation Type window is displayed.
7. Select **Put Message into MQ** and click **Next**.
8. Enter `test_out` in the **Queue Name** field. The MQSeries adapter will enqueue the message to this queue. Leave the other fields to default, as shown in [Figure 6-32](#).

Figure 6-32 Get Message from MQ window

Adapter Configuration Wizard - Step 5 of 6: Put Message into MQ

Enter information for putting a normal message into MQ Series.

Queue Name:

Queue Manager (optional):

Message Format:

Priority:

Persistence:

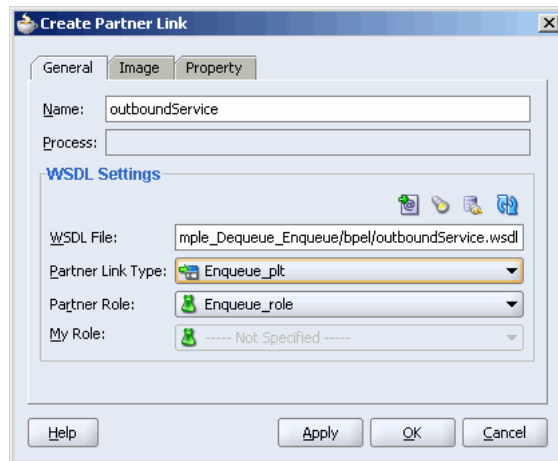
Expiry:

☒ Never

☐ Expires in:

Help < Back Next > Finish Cancel

9. Click **Next**. The Messages window is displayed.
10. Click **Browse**. The Type Chooser dialog box is displayed.
11. Select **Project Schema Files**, `address-csv.xsd` and then **Root-Element** as shown in [Figure 6-30](#) and click **OK**.
12. Click **Next**.
13. Click **Finish**. The Create Partner Link dialog box appears, as shown in [Figure 6-33](#).

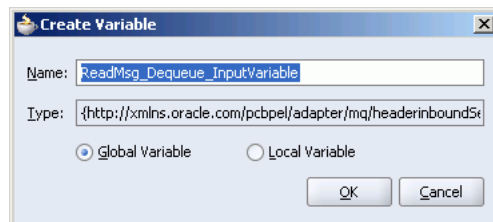
Figure 6–33 OutboundService Create Partner Link dialog box

14. Click **OK**.

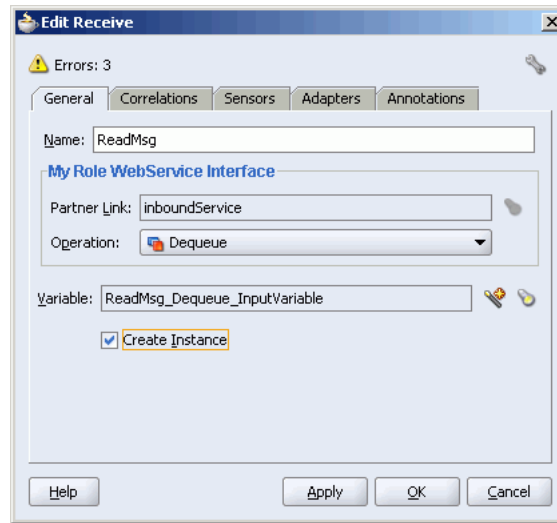
6.5.1.4 Creating the Receive Activity

Perform the following steps to create a Receive activity to instantiate a BPEL process instance when it receives a message from the MQSeries adapter:

1. Drag the **Receive** activity from the Components palette to the Drop Activity Here section.
2. Join the Receive activity to the inboundService partner link. The Edit Receive dialog box is displayed.
3. Enter ReadMsg in the **Name** field.
4. Click the Auto-Create Variable icon next to the **Variable** field. The Create variable dialog box is displayed, as shown in [Figure 6–34](#).

Figure 6–34 Create Variable dialog box

5. Click **OK**. The ReadMsg_Dequeue_InputVariable variable is created.
6. Select **Create Instance**. The Edit Receive dialog box appears as shown in [Figure 6–35](#).

Figure 6–35 Edit Receive Dialog Box

7. Click **OK**.

6.5.1.5 Creating the Invoke Activity

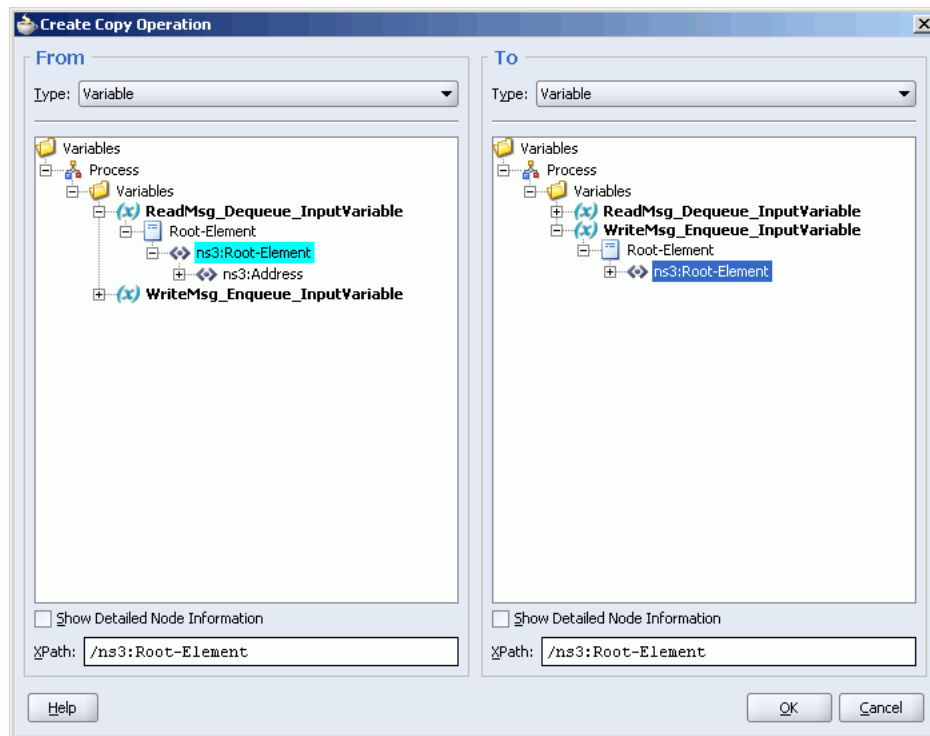
Perform the following steps to create an Invoke activity that will send the message to partner link, which enqueues the message to the outbound queue:

1. Drag an **Invoke** activity below the receive activity from the Component palette .
2. Join the Invoke activity to the `outboundService` partner link. The Edit Invoke dialog box is displayed.
3. Enter `WriteMsg` in the **Name** field.
4. Click the Auto-Create Variable icon next to the Variable field. The Create variable dialog box is displayed.
5. Click **OK**. The `WriteMsg_Enqueue_InputVariable` variable is created.
6. Click **OK**.

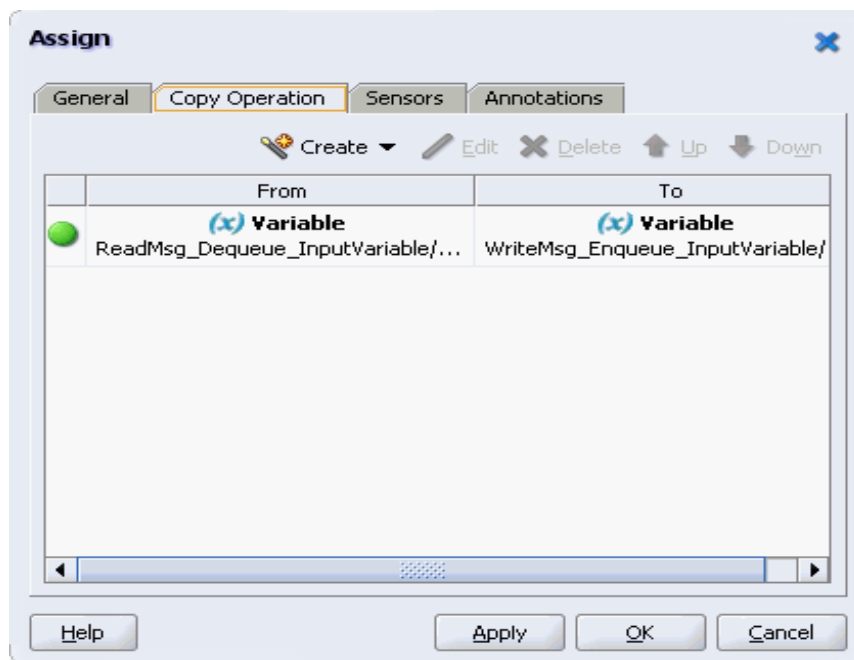
6.5.1.6 Creating the Assign Activity

Perform the following steps to create an Assign activity:

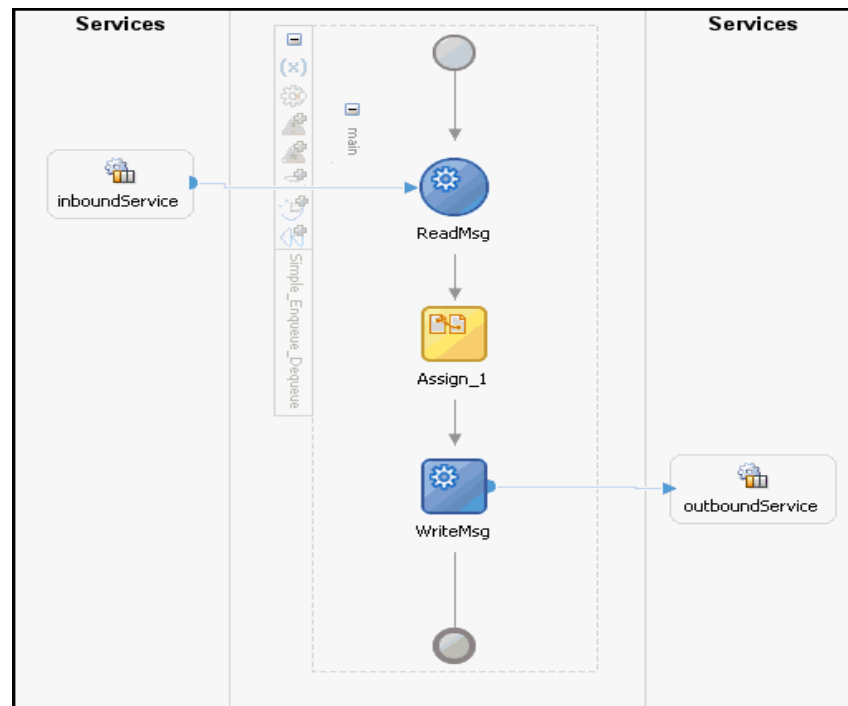
1. Drag an **Assign** activity from the Component Palette between the `ReadMsg` and the `WriteMsg` activity.
2. Double-click the Assign activity.
3. In the Copy operation tab, click **Create** and then select **Copy Operation**. The Create Copy Operation dialog box is displayed.
4. In the **From** group, select **Variable, Process, Variables, ReadMsg_Dequeue_InputVariable, Root-Element, ns3:Root-Element**, as shown in [Figure 6–36](#).
5. In the **To** group, select **Variable, Process, Variables, WriteMsg_Enqueue_InputVariable, Root-Element, ns3:Root-Element**, as shown in [Figure 6–36](#).

Figure 6–36 Create Copy Operation Dialog Box

6. Click **OK**. The Assign activity dialog box appears, as shown in [Figure 6–37](#).

Figure 6–37 Assign Activity Dialog Box

7. Click **OK**. The Oracle JDeveloper appears, as shown in [Figure 6–38](#).

Figure 6–38 Simple_Enqueue_Dequeue Use Case Design View

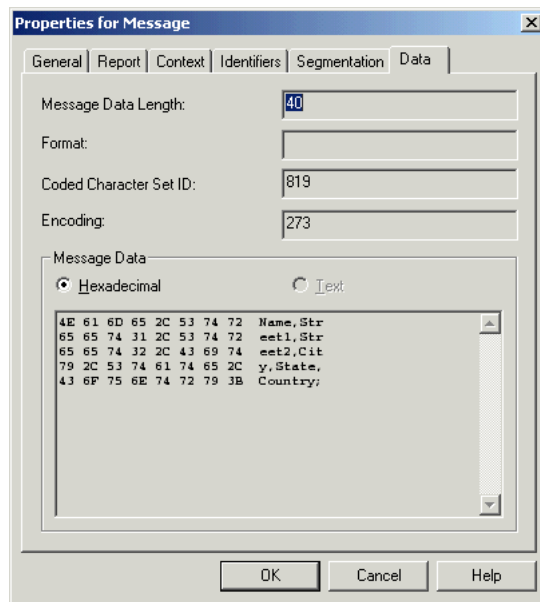
6.5.1.7 Run-Time

Perform the following steps to test the run-time of this scenario:

1. In the Applications window, click **Applications** and then right-click **Simple_Dequeue_Enqueue**.
2. Deploy the application.
3. Put the following text message to the `test_in` queue:

```
Name,Street1,Street2,City,State,Country;
```

The same text will be written to the `test_out` queue. [Figure 6–39](#) displays the sample `test_out` queue to which the data has been written.

Figure 6–39 Sample test_out Queue

6.5.2 Synchronous Request-Response

In this use case, the inbound MQSeries adapter dequeues the request message from MQSeries inbound queue `test_in` and publishes it to the BPEL process. The MQSeries adapter waits for the response from the BPEL process. When the MQSeries adapter receives the response, it enqueues the response message to the MQSeries queue specified in the `replyTo` queue of the request message. This use case consists of following sections:

- [Section 6.5.2.1, "Prerequisites"](#)
- [Section 6.5.2.2, "Creating the Synchronous Request-Response Service"](#)
- [Section 6.5.2.3, "Creating the Receive Activity"](#)
- [Section 6.5.2.4, "Creating the Reply Activity"](#)
- [Section 6.5.2.5, "Creating the Transform Activity"](#)
- [Section 6.5.2.6, "Run-Time"](#)

6.5.2.1 Prerequisites

This example assumes that you are familiar with basic BPEL constructs, such as activities and partner links, and Oracle JDeveloper environment for creating and deploying BPEL Process.

The MQSeries adapter must be configured as specified in [Section 6.4, "Configuring the MQSeries Adapter"](#) and two queue `test_in` and `test_out` should be created.

To define the schema of the messages, you require the `address-csv.xsd` and the `address-fixedLength.xsd` files. This files can be copied from the following location:

```
ORAHOME\bpel\samples\tutorials\121.FileAdapter\FlatStructure
```

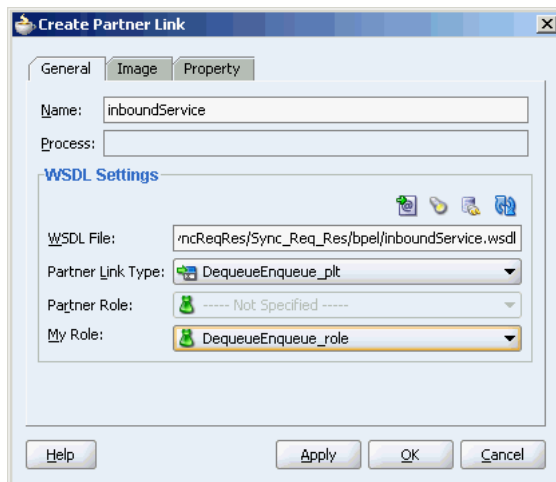
6.5.2.2 Creating the Synchronous Request-Response Service

Perform the following steps in Oracle JDeveloper to create an partner link that dequeues the request message from the inbound queue and enqueues the response message to the outbound queue:

1. In the Application Navigator, right-click **Applications** and select **New Application**. The Create Application dialog box is displayed.
2. Enter `SyncReqRes` in the **Application Name** field and click **OK**. The Create Project dialog box is displayed.
3. Click **Cancel**.
4. In the Applications Navigator, right-click **SyncReqRes** and select **New Project**. The New Gallery dialog box is displayed.
5. From Categories, select **General** and then **Projects**.
6. From Items, select **BPEL Process Project** and click **OK**. The BPEL Process Creation Wizard-Project Settings dialog box is displayed.
7. Perform the following:
 - a. Enter `Sync_Req_Res` in the **Name** field.
 - b. Select **Empty BPEL Process** from Template box.
 - c. Click **Finish**.
8. Create a Schema folder and copy the `address-csv.xsd` file to this folder.
9. Move the Schema folder to `bpel` sub directory of `Sync_Req_Res` project directory.
10. Drag a **Partner Link** activity from the Component Palette to the design area. The Create Partner Link dialog box is displayed.
11. Enter `inbound_req_res` in the **Name** field and click the Define Adapter Service icon shown in [Figure 6-27](#). The Welcome window of the Adapter Configuration Wizard is displayed.
12. Click **Next**. The Adapter Type window is displayed.
13. Select **MQ Adapter** and click **Next**. The Service Name window is displayed.
14. Enter `inboundService` in the **Service Name** field and click **Next**. The MQ Series Connection window is displayed.
15. Specify the JNDI name for the run-time connection in the **MQ Series Connection JNDI Name** field as shown in [Figure 6-28](#) and click **Next**. The Operation Type window is displayed.
16. Select **Get message from MQ and Send Reply/Reports**.
17. Select **Synchronous** option as shown in [Figure 6-17](#) and click **Next**. The Get message from MQ and Send Reply/Reports window is displayed.
18. Enter `test_in` in the **Queue Name** field and click **Next**. The Response window shown in [Figure 6-19](#) is displayed.
19. Specify the name of the response fallback queue in the **Response Fallback Queue Name** field and click **Next**. The Messages window shown in [Figure 6-15](#) is displayed.
20. In the Get Messages Schema group, click **Browse**. The Type Chooser dialog box is displayed.

21. Select **Project Schema Files**, **address-csv.xsd** and then **Root-Element** and click **OK**.
22. In the Send Messages Schema group, click **Browse**. The Type Chooser dialog box is displayed.
23. Click **Browse** in the Messages window. The Type Chooser dialog box is displayed.
24. Select **Project Schema Files**, **address-fixedLength.xsd** and then **Root-Element** and click **OK**.
25. Click **Next** in the Messages window.
26. Click **Finish**.
27. In the Create Partner Link dialog box, select **Dequeue_role** from the My Role box as shown in [Figure 6–40](#).

Figure 6–40 Synchronous Request-Response Use Case Create Partner Link Dialog Box

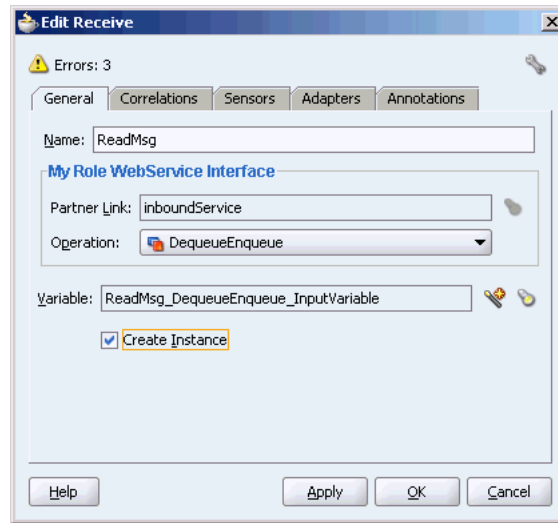


28. Click **OK**.

6.5.2.3 Creating the Receive Activity

Perform the following steps to create a Receive activity that instantiates a BPEL Process instance when it receives a message from the MQSeries adapter:

1. Drag the **Receive** activity from the Components palette to the Drop Activity Here section.
2. Join the receive activity to the `inboundService` partner link. The Edit Receive dialog box is displayed.
3. Enter `ReadMsg` in the **Name** field.
4. Click the Auto-Create Variable icon next to the **Variable** field. The Create variable dialog box is displayed.
5. Click **OK**. A `ReadMsg_DequeueEnqueue_InputVariable` variable is created.
6. Select **Create Instance**. The Edit Receive dialog box appears, as shown in [Figure 6–35](#).

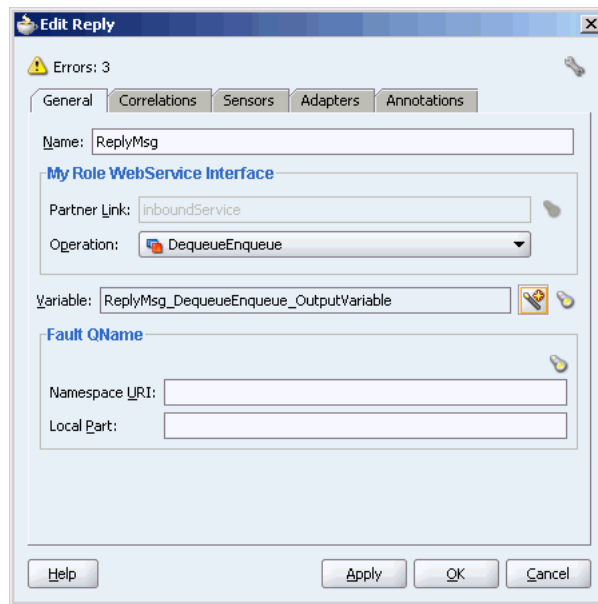
Figure 6–41 Edit Receive Dialog Box

7. Click **OK**.

6.5.2.4 Creating the Reply Activity

Perform the following steps to create a Reply activity that will send the response to the `inboundService` partnerlink.

1. Drag the **Reply** activity from the Components palette to below the `ReadMsg` activity.
2. Join the reply activity to the `inboundService` partner link. The Edit Reply dialog box is displayed.
3. Enter `ReplyMsg` in the **Name** field.
4. Click the Auto-Create Variable icon next to the **Variable** field. The Create variable dialog box is displayed.
5. Click **OK**. A `ReplyMsg_DequeueEnqueue_OutputVariable` variable is created. The Edit Reply dialog box appears, as shown in [Figure 6–42](#).

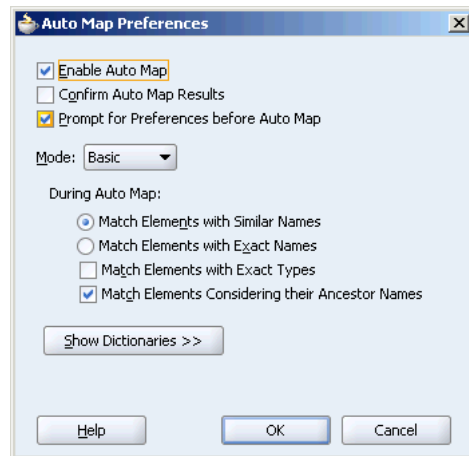
Figure 6–42 Edit Reply Dialog Box

6. Click OK.

6.5.2.5 Creating the Transform Activity

Perform the following steps to map the ReadMsg_DequeueEnqueue_InputVariable variable to ReplyMsg_DequeueEnqueue_OutputVariable variable:

1. Drag the **Transform** activity from the Components palette to between the ReadMsg and the ReplyMsg activity.
2. Double-click the **Transform** activity.
3. From the Source Variable box, select **ReadMsg_DequeueEnqueue_InputVariable**.
4. From the Target Variable box, select **ReplyMsg_DequeueEnqueue_OutputVariable**.
5. Click the Create Mappings icon. The Transformation_1.xsl window opens.
6. Drag the `tns:Address` element from <source>panel to the Address element of the <target> panel. The Auto Map Preferences dialog box is displayed, as shown in [Figure 6–43](#).

Figure 6–43 Auto Map Preferences Dialog Box

7. Deselect Match Elements Considering their Ancestor Names and click **OK**.
8. Drag a **concat** function from the Components palette to the *Street1* node in the <source> panel.
9. Join the *Street2* node to the Concat function. The *Transformation_1.xsl* window appears as shown in .
10. Close the *Transformation_1.xsl* window.

6.5.2.6 Run-Time

Perform the following steps to test the run-time of this scenario:

1. In the Applications navigator, click **Applications** and then right-click **Sync_Req_Res**.
2. Deploy the application.
3. Put the following text message to the *test_in* queue:

```
Name,Street1,Street2,City,State,Country;
```

The same text will be written to the *test_out* queue in the following format:

```
Name,Street1Street2,City,State,Country;
```

6.5.3 Demonstrations and Samples

A series of demonstrations, activity and conceptual reference materials, and tutorials are provided to increase conceptual knowledge and hands-on experience with Oracle Enterprise Service Bus and Oracle BPEL Process Manager adapters.

These samples are available in the *Samples* folder.

In addition, you can find updated adapter tutorials at following location:

http://www.oracle.com/technology/products/integration/adapters/dev_support.html#tutorials

Native Format Builder Wizard

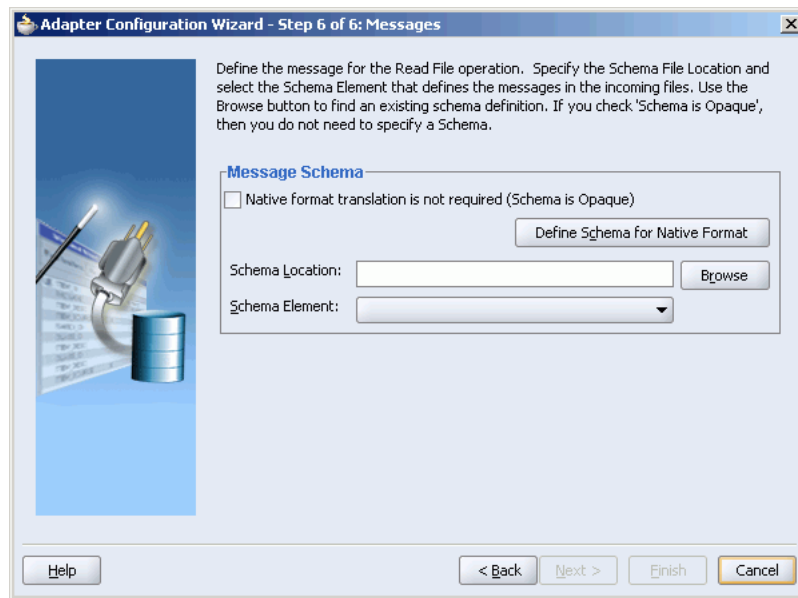
This chapter describes the Native Format Builder Wizard, which enables you to create native schemas used for translation. Use cases and constructs for the schema are also provided.

This chapter contains the following topics:

- [Section 7.1, "Creating Native Schema Files with the Native Format Builder Wizard"](#)
- [Section 7.2, "Understanding Native Schema"](#)
- [Section 7.3, "Native Schema Constructs"](#)

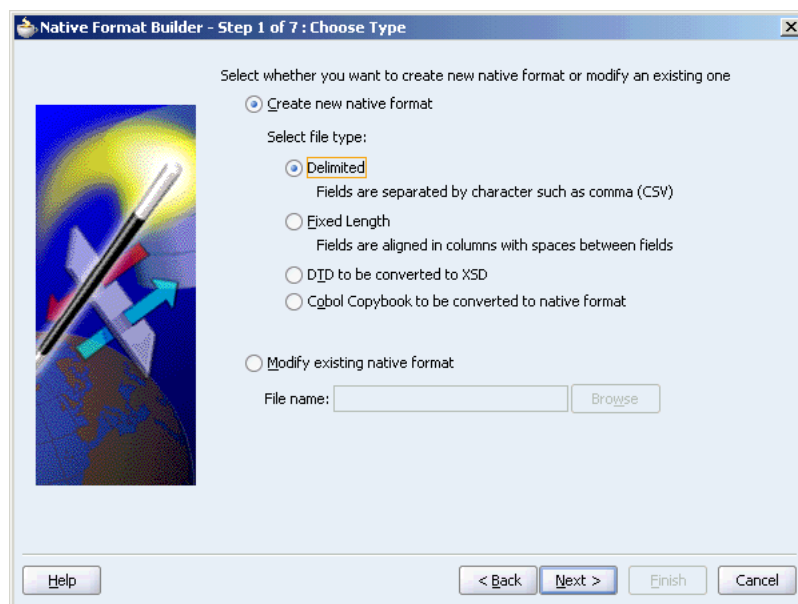
7.1 Creating Native Schema Files with the Native Format Builder Wizard

Oracle BPEL Process Manager and Oracle Enterprise Service Bus requires native schemas for translation, which are based on XML schema. However, not all commonly used formats use XML schema files. To address this situation, Oracle BPEL Process Manager and Oracle Enterprise Service Bus provides the Native Format Builder Wizard. When you click the **Define Schema for Native Format** button of the Messages dialog box of the Adapter Configuration Wizard shown in [Figure 7-1](#), the Native Format Builder Wizard is displayed. The Messages window is the last window to be displayed in the Adapter Configuration Wizard prior to the Finish window.

Figure 7–1 Starting the Native Format Builder Wizard

7.1.1 Supported Formats

The Native Format Builder Wizard guides you through the creation of a native schema file from the following formats shown in [Figure 7–2](#). A sample data file format for the selected type must already exist; you cannot create a native schema without one. You can also select to modify an existing native schema previously created with this wizard, except for those generated from a Document Type Definition (DTD) or COBOL copybook.

Figure 7–2 Native Format Builder Wizard

7.1.1.1 Delimited

This option enables you to create native schemas for records, where the fields are separated by a value such as a comma or number sign(#).

7.1.1.2 Fixed Length (Positional)

This option enables you to create native schemas for records, where all fields are all of fixed lengths.

7.1.1.3 DTD

This option enables you to generate native schema from the user-supplied DTD.

7.1.1.4 COBOL Copybook

This option enables you to generate native schema from the user-supplied COBOL copybook definition.

A COBOL mainframe application typically uses a COBOL copybook file to define its data layout. The converter creates a native schema from a copybook so that the run-time translator can parse the associated data file.

A COBOL copybook is typically a collection of group items (structures). These group items contain other items, which can be groups or elementary items. Elementary items are items that cannot be further subdivided. For example:

```

01 Purchase-Order
    05 Buyer
        10 BuyerName PIC X(5) USAGE DISPLAY.
    04 Seller
        08 SellerName PICTURE XXXXX.

```

Purchase-order is a group item with two child group items (Buyer, Seller). The numbers 01, 05, 04, and so on indicate the level of the group (that is, the hierarchy of data within that group).

Groups can be defined that have different level-numbers for the same level in the hierarchy. For example, Buyer and Seller have different level numbers, but are at the same level in the hierarchy. A group item includes all group and elementary items that follow it until a level number less than or equal to the level number of that group is encountered.

Each of the group items (Buyer and Seller) has a child elementary item. The PIC or PICTURE clause defines the data layout. For example, BuyerName defines an alphanumeric type of size equal to five characters. SellerName has exactly the same data layout as BuyerName.

Group items in COBOL can be mapped to elements in XML schema with the complexType type. Similarly, elementary items can be mapped to elements of type simple type with certain native format annotations to help the run time translator parse the corresponding data file.

For example, the Buyer item can be mapped to the following definition:

```

<!--COBOL declaration : 05 Buyer-->
<element name="Buyer">
  <complexType>
    <sequence>
      <!--COBOL declaration : 10 Name PIC X(5)-->
      <element name="Name" type="string" nxsd:style="fixedLength"
        nxsd:padStyle="tail" nxsd:paddedBy=" " nxsd:length="5"/>
    
```

```

        </sequence>
    </complexType>
</element>

```

User Inputs

You are expected to provide the following information:

- Target namespace for the native schema to be generated
- Character set of the host computer on which the data file was generated. By default this is set to EBCDIC (ebcdic-cp-us).
- Byte order of the host computer on which the data file was generated. By default this is set to big-endian.
- Record delimiter, which is typically the newline character, or no delimiter, or any user-supplied string
- Container tag name for generated native schema. By default, this is set to Root-Element.

COBOL Clauses

Table 7–1 describes COBOL clauses. The numeric types covered in Table 7–1 are stored as one character per digit. Support for clauses is defined as follows:

- Y indicates that the clause is supported.
- N indicates that the clause is not supported.
- I indicates that the clause is ignored.

Table 7–1 COBOL Clauses (Numeric Types Stored as One Character Per Digit)

COBOL Clause	Design-Time Support	Run-Time Support	Supported Synonyms	Comments
PIC X(<i>n</i>)	Y	Y	XXX...	Alphanumeric – An allowable character from the character set of the computer. Each X corresponds to one byte.
PIC A(<i>n</i>)	Y	Y	AA...	Alphabetic – Any letter of the alphabet or space. Each A corresponds to one byte.
PIC 9(<i>n</i>) DISPLAY	Y	Y	9999...	Any character position that contains a numeral. Each nine is counted in the size of the item.
OCCURS <i>n</i> TIMES	Y	Y		Fixed-length array
JUSTIFIED	Y	Y		For A and X types. Right justifies with the space pad. data is aligned at the rightmost character position
REDEFINES	Y	Y		Allows the same computer memory area to be described by different data items.
PIC 9(<i>m</i>)V9(<i>n</i>) DISPLAY	Y	Y		Size = <i>n+m</i> bytes
OCCURS DEPENDING ON	Y	Y		NA

Table 7–1 (Cont.) COBOL Clauses (Numeric Types Stored as One Character Per Digit)

COBOL Clause	Design-Time Support	Run-Time Support	Supported Synonyms	Comments
BLANK WHEN ZERO	I	I		Ignored
RENAMES	N	N		This is rarely seen in COBOL copybooks
INDEX	N	N		Four-byte index
SYNCHRONIZED	I	I	SYNC	NA
POINTER	N	N		NA
PROCEDURE-POINTER				NA
FILLER	Y	Y		NA

The numeric types described in [Table 7–1](#) are stored as one character per digit. [Table 7–2](#) describes the numeric types that are stored in a more efficient manner.

Table 7–2 COBOL Clauses (Numeric Types Stored More Efficiently)

COBOL Clause	Design-Time Support	Run-Time Support	Supported Synonyms	Comments
USAGE [IS]	Y	Y		Both these keywords are optional.
PIC 9(n) COMP	Y	Y	COMPUTATIONAL, BINARY, COMP-4	Length varies with <i>n</i> : <ul style="list-style-type: none"> ■ <i>n</i> = 1–4 (2 bytes) ■ <i>n</i> = 5–9 (4 bytes) ■ <i>n</i> = 10–18 (8 bytes)
COMP-1	Y	Y	COMPUTATIONAL-1	Single precision, floating point number that is four bytes long.
COMP-2	Y	Y	COMPUTATIONAL-2	Double precision, floating point number that is eight bytes long.
PIC 9(n) COMP-3	Y	Y	PACKED-DECIMAL, COMPUTATIONAL-3	Two digits are stored in each byte. An additional half byte at the end is allocated for the sign, even if the value is unsigned.
PIC 9(n) COMP-4	Y	Y	COMPUTATIONAL-4	Treated the same as a COMP type and given its own data type for customization requirements.
PIC 9(n) COMP-5	N	N		Capacity of the native binary representation.
PIC S9(n) DISPLAY	Y	Y	PIC S99...	Sign nibble in the right-most zone by default. <i>S</i> is not counted in the size.
PIC S9(n) COMP	Y	Y		Same as COMP. Negative numbers are represented as two's complement.
PIC S9(n) COMP-3	Y	Y		NA
PIC 9(m)V9(n) COMP	Y	Y		Length is the same as COMP.

Table 7–2 (Cont.) COBOL Clauses (Numeric Types Stored More Efficiently)

COBOL Clause	Design-Time Support	Run-Time Support	Supported Synonyms	Comments
PIC 9 (m) V9 (m) COMP-3	Y	Y		Length = Ceiling $((n+m+1)/2)$

The following clauses can be added to impact the sign position.

- SIGN IS LEADING
Used with signed zoned numerics.
- SIGN IS TRAILING
Used with signed zoned numerics.
- SIGN IS LEADING SEPARATE
The character S is counted in the size.
- SIGN IS TRAILING SEPARATE
The character S is counted in the size.

Note: These assume that the numerics are stored using IBM COBOL format. If these are generated for other platforms with different data storage formats, then a custom data handler for that type must be written.

Table 7–3 describes picture editing types.

Table 7–3 Edited Pictures

Edited Pictures	Supported Editing Types	Unsupported Editing Types
Edited alphanumeric	Simple Insertion: B(blank) 0 / ,	
Edited float numeric	Special insertion: . (period)	
Edited numeric	<ul style="list-style-type: none"> ■ Simple Insertion: B(blank) 0 / , ■ Special insertion: . (period) ■ Fixed Insertion: cs + - CR DB (Inserts a symbol at the beginning or end) 	<ul style="list-style-type: none"> ■ Floating Insertion: cs + - ■ Zero suppression: Z * ■ Replacement insertion: Z * + - c

Edited pictures are more for presentation purposes and are rarely seen in data files. It is assumed that the editing symbols are also present in the data. For example, if you have:

```
05  AMOUNT      PIC 999.99
```

Then, this field is six bytes wide and has a decimal point in the data.

Simple, special, and fixed insertions are handled by this method. Floating insertion, zero suppression, and replacement insertion are not supported.

7.1.2 Native Format Builder Wizard Windows

For delimited and fixed-length files, you are guided through dialog boxes that prompt you for the following information to create definitions in native schema format:

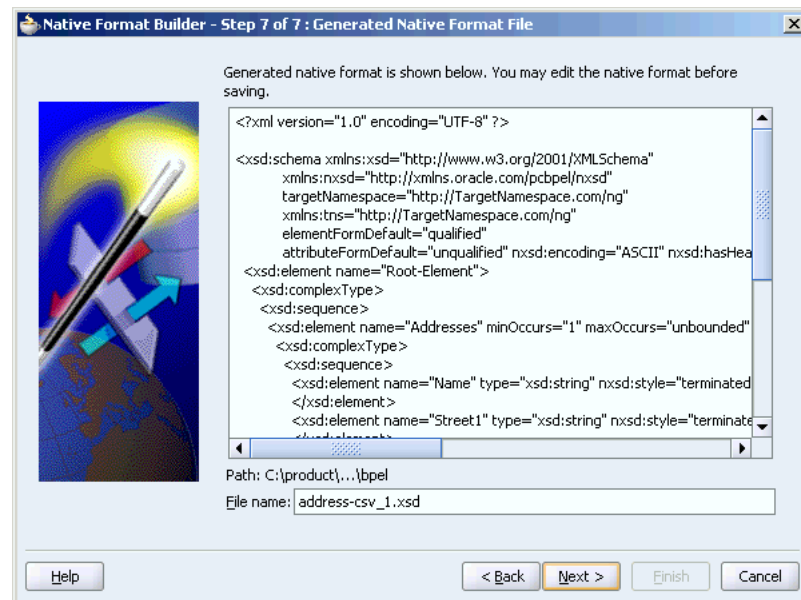
- The data file to sample (from which to create a native schema) and its encoding
- The number of rows to skip and the number of rows to sample in the file
- If the file contains multiple records, are they the same type or different types
- The target namespace, container element name, and record name
- The record delimiter (for example, end-of-line) and field delimiter (for example, comma or hash sign (#)) or field length
- The field properties (such as name, data_type, delimiter, and length)

DTD and COBOL copybook files already include definitions in their native formats. For these formats, the Native Format Builder Wizard prompts you for the following information to create native schema versions of these definitions.

- The filename of the DTD or COBOL copybook definition, namespace, and root element
- The character set, byte order, and records delimiter (for COBOL copybook only)

As you move through the wizard dialog boxes, the native schema file being created is displayed at the bottom. This enables you to watch the native schema file being built. The final window displays the generated native schema for the native format shown in [Figure 7-3](#). You can edit this format before clicking **Next**.

Figure 7-3 Native Schema Generated From Native Format



When you click **Finish**, you are returned to the Messages dialog box of the Adapter Configuration Wizard shown in [Figure 7-1](#). The **Schema File URL** and **Schema Element** fields are filled in with details about your newly created native schema file. You can now use the Adapter Configuration Wizard to create a WSDL file for the adapter to communicate with your BPEL process or ESB service.

7.2 Understanding Native Schema

This section provides use cases and explains various constructs of native schema to translate the native format data to XML.

This section contains the following topics:

- [Use Cases for the Native Format Builder](#)
- [Native Schema Constructs](#)

Note: Not all native schemas can be generated from the Native Format Builder Wizard. This wizard can handle only basic scenarios. This section describes the capabilities of the native schema using various examples and use cases.

7.2.1 Use Cases for the Native Format Builder

This section contains the following topics:

- [Defining a Comma-Separated Value File Structure](#)
- [Defining a * Separated Value File Structure](#)
- [Defining a Fixed-Length Structure](#)
- [Defining a More Complex Structure - Invoice](#)
- [COBOL Copybook](#)

7.2.1.1 Defining a Comma-Separated Value File Structure

A comma-separated value (CSV) file is a common non-XML file structure. A CSV file may or may not have the first few lines as headers, in which case, you may want to ignore them.

Native Data Format to Be Translated

The following native data format is provided:

```
Name,Street,City,State,Country
ABC Private Limited, Street 1, Bangalore, Karnataka, India
XYZ Private Limited, Street 2, Bangalore, Karnataka, India
```

Native Schema

The corresponding native schema definition can be defined as follows:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://www.oracle.com/ias/processconnect"
  xmlns:tns="http://www.oracle.com/ias/processconnect"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:encoding="US-ASCII"
  nxsd:headerLines="1"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <xsd:element name="AddressBook">
    <xsd:complexType>
      <xsd:sequence>
```

```

<xsd:element name="Address" minOccurs="1" maxOccurs="unbounded">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Name" type="xsd:string" nxsd:style="terminated"
        nxsd:terminatedBy=", " >
      </xsd:element>
      <xsd:element name="Street" type="xsd:string" nxsd:style="terminated"
        nxsd:terminatedBy=", " >
      </xsd:element>
      <xsd:element name="City" type="xsd:string" nxsd:style="terminated"
        nxsd:terminatedBy=", " >
      </xsd:element>
      <xsd:element name="State" type="xsd:string" nxsd:style="terminated"
        nxsd:terminatedBy=", " >
      </xsd:element>
      <xsd:element name="Country" type="xsd:string" nxsd:style="terminated"
        nxsd:terminatedBy="{$eol}" >
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>

```

The `nxsd:headerLines="1"` in this schema, at the `xsd:schema` construct, means that one line will be skipped in the native data before actually translating the rest of the data. This is because the first line is a header line. If `nxsd:stream` is specified as `chars` such as `nxsd:stream="chars"`, then it means that the data should be read as characters. If `nxsd:stream` is set as `bytes`, `nxsd:stream="bytes"`, then it means that the native data should be read as bytes. For each of the element declarations, `Name`, `Street`, `City`, `State`, `Country`, which have a corresponding scalar data, the `nxsd:style="terminated"` defines that the corresponding data is stored in terminated style. The actual terminator is then defined by the `nxsd:terminatedBy=", "` attribute specified at that construct. See [Section 7.2.2.2, "Defining Terminated Data"](#) for details on the terminated style.

Translated XML Using the Native Schema

The native data using the corresponding native schema format is translated to the following XML:

```

<AddressBook xmlns="http://www.oracle.com/ias/processconnect">
  <Address>
    <Name>ABC Private Limited</Name>
    <Street>Street 1</Street>
    <City>Bangalore</City>
    <State>Karnataka</State>
    <Country>India</Country>
  </Address>
  <Address>
    <Name>XYZ Private Limited</Name>
    <Street>Street 2</Street>
    <City>Bangalore</City>
    <State>Karnataka</State>
    <Country>India</Country>
  </Address>
</AddressBook>

```

7.2.1.2 Defining a * Separated Value File Structure

The use case defined in the previous example is just one specific case of the *SV class, where the wildcard can be substituted by any character or string. For example, for the native data containing a plus (+) separated value.

Native Data Format to Be Translated

The following native data format is provided:

```
a+b+c+d+e
f+g+h+i+j
```

Native Schema

The corresponding native schema definition is similar to the one in the previous use case except that instead of `nxsd:terminatedBy=","` you now define the terminated by format as `nxsd:terminatedBy="+"`. See [Section 7.2.2.2, "Defining Terminated Data"](#) for details about the terminated style.

7.2.1.3 Defining a Fixed-Length Structure

In this example, the native data used is the same as in the CSV case. The only difference is that here the data is fixed length, and not CSV.

Native Data Format to Be Translated

The following native data format is provided:

Name	Street	City	State	Country
ABC Private Limited	Street1	Bangalore	Karnataka	India
XYZ Private Limited	Street1	Bangalore	Karnataka	India

Native Schema

The corresponding native schema definition is similar to the definition of the CSV, but the style now changes from `nxsd:style="terminated"` to `nxsd:style="fixedLength"` along with the relevant attributes for the fixed-length style. For the fixed-length style, the one mandatory attribute is the length: `nxsd:length`. The value of `nxsd:length` is the actual length of the data to be read. The complete definition for fixed-length style for the native data to be translated can be defined as follows:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://www.oracle.com/ias/processconnect"
  xmlns:tns="http://www.oracle.com/ias/processconnect"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:encoding="US-ASCII"
  nxsd:headerLines="1"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <xsd:element name="AddressBook">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Address" minOccurs="1" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Name" type="xsd:string" nxsd:style="fixedLength"
                nxsd:length="31">
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

```

        <xsd:element name="Street" type="xsd:string" nxsd:style="fixedLength"
            nxsd:length="19">
        </xsd:element>
        <xsd:element name="City" type="xsd:string" nxsd:style="fixedLength"
            nxsd:length="10">
        </xsd:element>
        <xsd:element name="State" type="xsd:string" nxsd:style="fixedLength"
            nxsd:length="10">
        </xsd:element>
        <xsd:element name="Country" type="xsd:string" nxsd:style="terminated"
            nxsd:terminatedBy="{eol}">
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

</xsd:schema>

```

See [Section 7.2.2.1, "Defining Fixed-Length Data"](#) for details about the fixed-length style.

7.2.1.4 Defining a More Complex Structure - Invoice

An invoice is a more complex structure than the structure in the previous CSV, *SV, and fixed-length use cases. An invoice usually contains buyer information, seller information, and line items.

Native Data Format to Be Translated

The following native data format for an invoice is provided:

```

6335722^Company One^First Street 999 San Jose 95129USCA650-801-6250
^Oracle^Bridge Parkway 1600 Redwood Shores 94065USCA650-506-7000
001|BPEL Process Manager Enterprise Edition|20000,2,+40000+
002|BPEL Process Manager Standard Edition|10000,5,+50000+
003|BPEL Process Manager Developer Edition|1000,20,+20000+#110000

```

The first line in the native data is purchaser details, followed by seller details, followed by line items, and finally the total for the line items. Both purchaser and seller have the following same structure:

- The first seven characters are the UID
- This is followed by the buyer/seller name surrounded by “^”.
- This is followed by the address until the end of the line.

This address contains a fixed-length street, city, and so on. The last line item ends with the number symbol “#”, followed by the line-item total.

Native Schema

The native schema definition corresponding to the preceding native data can be defined as follows:

```

<schema attributeFormDefault="qualified" elementFormDefault="qualified"

targetNamespace="http://xmlns.oracle.com/ias/pcbpel/fatransschema/demo"
  xmlns:tns="http://xmlns.oracle.com/ias/pcbpel/fatransschema/demo"
  xmlns="http://www.w3.org/2001/XMLSchema"

```

```

        xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
        nxsd:version="NXSD" nxsd:stream="chars">

<element name="invoice" type="tns:invoiceType" />

<complexType name="invoiceType">
  <sequence>
    <element name="purchaser" type="tns:partnerType" />
    <element name="seller" type="tns:partnerType" />
    <element name="line-item" type="tns:line-itemType"
      maxOccurs="unbounded" nxsd:style="array"
      nxsd:cellSeparatedBy="{eol}" nxsd:arrayTerminatedBy="#" />
    <element name="total" type="double" nxsd:style="terminated"
      nxsd:terminatedBy="{eol}" />
  </sequence>
</complexType>

<complexType name="partnerType">
  <sequence>
    <element name="uid" type="string" nxsd:style="fixedLength"
      nxsd:length="7" nxsd:padStyle="tail" nxsd:paddedBy=" " />
    <element name="name" type="string" nxsd:style="surrounded"
      nxsd:surroundedBy="^" />
    <element name="address" type="tns:addressType" />
  </sequence>
</complexType>

<complexType name="addressType">
  <sequence>
    <element name="street1" type="string" nxsd:style="fixedLength"
      nxsd:length="15" nxsd:padStyle="tail" nxsd:paddedBy=" " />
    <element name="street2" type="string" nxsd:style="fixedLength"
      nxsd:length="10" nxsd:padStyle="tail" nxsd:paddedBy=" " />
    <element name="city" type="string" nxsd:style="fixedLength"
      nxsd:length="15" nxsd:padStyle="tail" nxsd:paddedBy=" " />
    <element name="postal-code" type="string" nxsd:style="fixedLength"
      nxsd:length="5" nxsd:padStyle="none" />
    <element name="country" type="string" nxsd:style="fixedLength"
      nxsd:length="2" nxsd:padStyle="none" />
    <element name="state" type="string" nxsd:style="fixedLength"
      nxsd:length="2" nxsd:padStyle="none" />
    <element name="phone" type="string" nxsd:style="terminated"
      nxsd:terminatedBy="{eol}" />
  </sequence>
</complexType>

<complexType name="line-itemType">
  <sequence>
    <element name="uid" type="string" nxsd:style="fixedLength"
      nxsd:length="3" nxsd:padStyle="none" />
    <element name="description" type="string" nxsd:style="surrounded"
      nxsd:surroundedBy="|" />
    <element name="price" type="double" nxsd:style="terminated"
      nxsd:terminatedBy="," />
    <element name="quantity" type="integer" nxsd:style="terminated"
      nxsd:terminatedBy="," />
    <element name="line-total" type="double" nxsd:style="surrounded"
      nxsd:surroundedBy="+" />
  </sequence>

```

```

    </complexType>

</schema>

```

Translated XML Using the Native Schema

The translated XML looks as follows:

```

<invoice xmlns="http://xmlns.oracle.com/pcbpel/demoSchema/invoice-nxsd">
  <purchaser>
    <uid>6335722</uid>
    <name>Company One</name>
    <address>
      <street1>First Street</street1>
      <street2>999</street2>
      <city>San Jose</city>
      <postal-code>95129</postal-code>
      <country>US</country>
      <state>CA</state>
      <phone>650-801-6250</phone>
    </address>
  </purchaser>
  <seller>
    <uid/>
    <name>Oracle</name>
    <address>
      <street1>Bridge Parkway</street1>
      <street2>1600</street2>
      <city>Redwood Shores</city>
      <postal-code>94065</postal-code>
      <country>US</country>
      <state>CA</state>
      <phone>650-506-7000</phone>
    </address>
  </seller>
  <line-item>
    <uid>001</uid>
    <description>BPEL Process Manager Enterprise Edition</description>
    <price>20000</price>
    <quantity>2</quantity>
    <line-total>40000</line-total>
  </line-item>
  <line-item>
    <uid>002</uid>
    <description>BPEL Process Manager Standard Edition</description>
    <price>10000</price>
    <quantity>5</quantity>
    <line-total>50000</line-total>
  </line-item>
  <line-item>
    <uid>003</uid>
    <description>BPEL Process Manager Developer Edition</description>
    <price>1000</price>
    <quantity>20</quantity>
    <line-total>20000</line-total>
  </line-item>
  <total>110000</total>
</invoice>

```

7.2.1.5 Using the Native Format Translator for Removing or Adding Namespaces to XML with No Namespace

When the native data is XML and that XML has no namespace, the Native Format Translator can be used to add a namespace to an inbound XML document and remove the namespace from an outbound XML document.

The XML has no namespace when either of the following is true:

- The XML has a corresponding XML schema and there is no target namespace specified in that XML schema.
- The XML has a corresponding DTD which was converted to the XML schema.

In both cases, you need to create a wrapper schema with `targetNamespace` specified, and the wrapper schema should include the actual schema. In addition, the wrapper schema should also have the `nxsd:version` attribute set to DTD. For example:

```
--wrapper.xsd
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="myNamespace"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  nxsd:version="DTD">
  <include schemaLocation="actual.xsd"/>
</schema>
```

Note: Ensure that `elementFormDefault="qualified"` is specified in the actual schema.

Using this `wrapper.xsd` in place of `actual.xsd` would add `myNamespace` namespace to the inbound xml and would remove `myNamespace` namespace from the outbound xml.

7.2.1.6 COBOL Copybook

A demonstration is provided that shows how the file and FTP adapters process a file in COBOL copybook format (through use of the Native Format Builder Wizard) to create a native schema file for translation. For a demonstration that uses the Native Format Builder Wizard to convert a COBOL copybook file to a native schema file, go to

`Oracle_Home\integration\orabpel\samples\tutorials\121.FileAdapter\CobolCopyBook`

The following COBOL copybook examples are also provided:

- [Multiple Root Levels](#)
- [Single Root Level, Virtual Decimal, Fixed-Length Array](#)
- [Variable Length Array](#)
- [Numeric Types](#)

Multiple Root Levels

A COBOL copybook can have multiple root levels. If all root levels are at 01 level, then each such group implicitly redefines the other.

```
01  PAYROLL-E-RECORD.
    05  PAYROLL-E-EMPLOYEE-NUMBER      PIC X(10).
    05  PAYROLL-E-TRANS-CODE           PIC X(02).
```



```

05  PAYROLL-E-NAME          PIC X(08).
05  FILLER                  PIC X(25).

01  PAYROLL-F-RECORD.
05  PAYROLL-F-EMPLOYEE-NUMBER PIC X(10).
05  PAYROLL-F-TRANS-CODE     PIC X(02).
05  PAYROLL-F-IDENTIFIER-VALUE PIC X(03).
05  PAYROLL-F-NAME          PIC X(30).

01  PAYROLL-H-RECORD.
05  PAYROLL-H-EMPLOYEE-NUMBER PIC X(10).
05  PAYROLL-H-TRANS-CODE     PIC X(02).
05  PAYROLL-H-HED-NUMBER     PIC 9(03).
05  FILLER                  PIC X(30).

```

The generated schema is like the following:

```

<?xml version="1.0" encoding="UTF-8" ?>
<!--Native format was generated from COBOL copybook :
D:\work\jDevProjects\CCB\Copybooks\payroll.cpy-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  xmlns:extn="http://xmlns.oracle.com/pcbpel/nxsd/extensions"
  targetNamespace="http://TargetNamespace.com/ccb/implicitRedefines"
  xmlns:tns="http://TargetNamespace.com/ccb/implicitRedefines"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  nxsd:version="NXSD" nxsd:encoding="cp037" nxsd:byteOrder="bigEndian"
  nxsd:stream="chars">
  <xsd:element name="Payroll-Records">
    <xsd:complexType>
      <!--Please add values for nxsd:lookAhead attributes for the elements in the
        choice model group.-->
      <xsd:choice minOccurs="1" maxOccurs="unbounded">
        <!--COBOL declaration : 01 PAYROLL-E-RECORD -->
        <xsd:element name="PAYROLL-E-RECORD" nxsd:lookAhead="" nxsd:lookFor="">
          <xsd:complexType>
            <xsd:sequence>
              <!--COBOL declaration : 05 PAYROLL-E-EMPLOYEE-NUMBER PIC X(10)-->
              <xsd:element name="PAYROLL-E-EMPLOYEE-NUMBER" type="xsd:string"
                nxsd:style="fixedLength" nxsd:padStyle="tail"
                nxsd:paddedBy=" " nxsd:length="10"/>
              <!--COBOL declaration : 05 PAYROLL-E-TRANS-CODE PIC X(02)-->
              <xsd:element name="PAYROLL-E-TRANS-CODE" type="xsd:string"
                nxsd:style="fixedLength" nxsd:padStyle="tail"
                nxsd:paddedBy=" " nxsd:length="2"/>
              <!--COBOL declaration : 05 PAYROLL-E-NAME PIC X(08)-->
              <xsd:element name="PAYROLL-E-NAME" type="xsd:string"
                nxsd:style="fixedLength" nxsd:padStyle="tail"
                nxsd:paddedBy=" " nxsd:length="8"/>
              <!--COBOL declaration : 05 FILLER PIC X(25)-->
              <xsd:element name="FILLER" type="xsd:string"
                nxsd:style="fixedLength" nxsd:padStyle="tail"
                nxsd:paddedBy=" " nxsd:length="25"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <!--COBOL declaration : 01 PAYROLL-F-RECORD -->
        <xsd:element name="PAYROLL-F-RECORD" nxsd:lookAhead="" nxsd:lookFor="">
          <xsd:complexType>
            <xsd:sequence>

```

```

<!--COBOL declaration : 05 PAYROLL-F-EMPLOYEE-NUMBER PIC X(10)-->
<xsd:element name="PAYROLL-F-EMPLOYEE-NUMBER" type="xsd:string"
  nxsd:style="fixedLength" nxsd:padStyle="tail"
  nxsd:paddedBy=" " nxsd:length="10"/>
<!--COBOL declaration : 05 PAYROLL-F-TRANS-CODE PIC X(02)-->
<xsd:element name="PAYROLL-F-TRANS-CODE" type="xsd:string"
  nxsd:style="fixedLength" nxsd:padStyle="tail"
  nxsd:paddedBy=" " nxsd:length="2"/>
<!--COBOL declaration : 05 PAYROLL-F-IDENTIFIER-VALUE PIC X(03)-->
<xsd:element name="PAYROLL-F-IDENTIFIER-VALUE" type="xsd:string"
  nxsd:style="fixedLength" nxsd:padStyle="tail"
  nxsd:paddedBy=" " nxsd:length="3"/>
<!--COBOL declaration : 05 PAYROLL-F-NAME PIC X(30)-->
<xsd:element name="PAYROLL-F-NAME" type="xsd:string"
  nxsd:style="fixedLength" nxsd:padStyle="tail"
  nxsd:paddedBy=" " nxsd:length="30"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<!--COBOL declaration : 01 PAYROLL-H-RECORD -->
<xsd:element name="PAYROLL-H-RECORD" nxsd:lookAhead=" " nxsd:lookFor=" ">
<xsd:complexType>
<xsd:sequence>
<!--COBOL declaration : 05 PAYROLL-H-EMPLOYEE-NUMBER PIC X(10)-->
<xsd:element name="PAYROLL-H-EMPLOYEE-NUMBER" type="xsd:string"
  nxsd:style="fixedLength" nxsd:padStyle="tail"
  nxsd:paddedBy=" " nxsd:length="10"/>
<!--COBOL declaration : 05 PAYROLL-H-TRANS-CODE PIC X(02)-->
<xsd:element name="PAYROLL-H-TRANS-CODE" type="xsd:string"
  nxsd:style="fixedLength" nxsd:padStyle="tail"
  nxsd:paddedBy=" " nxsd:length="2"/>
<!--COBOL declaration : 05 PAYROLL-H-HED-NUMBER PIC 9(03)-->
<xsd:element name="PAYROLL-H-HED-NUMBER" type="xsd:long"
  nxsd:style="fixedLength" nxsd:padStyle="head"
  nxsd:paddedBy="0" nxsd:length="3"/>
<!--COBOL declaration : 05 FILLER PIC X(30)-->
<xsd:element name="FILLER" type="xsd:string"
  nxsd:style="fixedLength" nxsd:padStyle="tail"
  nxsd:paddedBy=" " nxsd:length="30"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:choice>
</xsd:complexType>
</xsd:element>
</xsd:schema>

```

The top-level payroll records are enclosed in a choice model group. Each payroll record also has two attributes: `nxsd:lookAhead` and `nxsd:lookFor` that help identify the type of record during runtime processing of the data file. So, you must add values for these attributes. For example, assume `PAYROLL-F-RECORD` occurs when the `PAYROLL-F-TRANS-CODE` field has a value of `FR`. The record element then looks as follows:

```
<xsd:element name="PAYROLL-F-RECORD" nxsd:lookAhead="10" nxsd:lookFor="FR">
```

The value 10 indicates the position of the lookahead field.

The following COBOL copybook has multiple root elements at the 05 level:

```
05 ORG-NUM          PIC 99.
```

```

05 EMP-RECORD.
   10 EMP-SSN      PIC 9(4)V(6).
   10 EMP-WZT      PIC 9(6).

```

The generated schema:

```

<?xml version="1.0" encoding="UTF-8" ?>
<!--Native format was generated from COBOL copybook :
D:\work\jDevProjects\CCB\Copybooks\multipleRoot.cpy-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  xmlns:extn="http://xmlns.oracle.com/pcbpel/nxsd/extensions"
  targetNamespace="http://TargetNamespace.com/ccb/multipleRoots"
  xmlns:tns="http://TargetNamespace.com/ccb/multipleRoots"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  nxsd:version="NXSD" nxsd:encoding="ASCII"
  nxsd:byteOrder="littleEndian" nxsd:stream="chars">
  <xsd:element name="emp-info">
    <xsd:complexType>
      <xsd:sequence minOccurs="1" maxOccurs="unbounded">
        <!--COBOL declaration : 05 ORG-NUM PIC 99-->
        <xsd:element name="ORG-NUM" type="xsd:long" nxsd:style="fixedLength"
          nxsd:padStyle="head" nxsd:paddedBy="0" nxsd:length="2"/>
        <!--COBOL declaration : 05 EMP-RECORD-->
        <xsd:element name="EMP-RECORD">
          <xsd:complexType>
            <xsd:sequence>
              <!--COBOL declaration : 10 EMP-SSN PIC 9(4)V(6)-->
              <xsd:element name="EMP-SSN" type="xsd:decimal"
                nxsd:style="virtualDecimal" extn:assumeDecimal="4"
                extn:picSize="9"/>
              <!--COBOL declaration : 10 EMP-WZT PIC 9(6)-->
              <xsd:element name="EMP-WZT" type="xsd:long"
                nxsd:style="fixedLength" nxsd:padStyle="head"
                nxsd:paddedBy="0" nxsd:length="6"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

In this (non-01 level) case, an unbounded sequence of the root level items is generated.

Single Root Level, Virtual Decimal, Fixed-Length Array

The following COBOL copybook has a single root level item PO-RECORD. In a single root level case, the level number does not matter because the converter works in same way. This COBOL copybook also shows an example of a field declared as a virtual decimal (PO-ITEM-PRICE).

```

05 PO-RECORD.
   10 PO-BUYER.
     15 PO-UID      PIC 9(7).
     15 PO-NAME     PIC X(15).
     15 PO-ADDRESS.
       20 PO-STREET PIC X(15).
       20 PO-CITY   PIC X(10).
       20 PO-ZIP    PIC 9(5).
       20 PO-STATE  PIC X(2).

```

```

10 PO-ITEM.
   15 POITEM OCCURS 3 TIMES.
       20 PO-LINE-ITEM.
           25 PO-ITEM-ID          PIC 9(3).
           25 PO-ITEM-NAME        PIC X(40).
           25 PO-ITEM-QUANTITY    PIC 9(2).
           25 PO-ITEM-PRICE       PIC 9(5)V9(2).
10 PO-TOTAL PIC 9(7)V9(2).

```

The generated schema looks as follows:

```

<?xml version="1.0" encoding="UTF-8" ?>
<!--Native format was generated from COBOL copybook : D:\work\
jDevProjects\CCB\Copybooks\po-ccb.cpy-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  xmlns:extn="http://xmlns.oracle.com/pcbpel/nxsd/extensions"
  targetNamespace="http://TargetNamespace.com/ccb/singleRoot"
  xmlns:tns="http://TargetNamespace.com/ccb/singleRoot"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  nxsd:version="NXSD" nxsd:encoding="cp037" nxsd:byteOrder="bigEndian"
  nxsd:stream="chars">
  <xsd:element name="Root-Element">
    <xsd:complexType>
      <xsd:sequence>
        <!--COBOL declaration : 05 PO-RECORD -->
        <xsd:element name="PO-RECORD" minOccurs="1" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <!--COBOL declaration : 10 PO-BUYER-->
              <xsd:element name="PO-BUYER">
                <xsd:complexType>
                  <xsd:sequence>
                    <!--COBOL declaration : 15 PO-UID PIC 9(7)-->
                    <xsd:element name="PO-UID" type="xsd:long"
                      nxsd:style="fixedLength" nxsd:padStyle="head"
                      nxsd:paddedBy="0" nxsd:length="7"/>
                    <!--COBOL declaration : 15 PO-NAME PIC X(15)-->
                    <xsd:element name="PO-NAME" type="xsd:string"
                      nxsd:style="fixedLength" nxsd:padStyle="tail"
                      nxsd:paddedBy=" " nxsd:length="15"/>
                    <!--COBOL declaration : 15 PO-ADDRESS-->
                    <xsd:element name="PO-ADDRESS">
                      <xsd:complexType>
                        <xsd:sequence>
                          <!--COBOL declaration : 20 PO-STREET PIC X(15)-->
                          <xsd:element name="PO-STREET" type="xsd:string"
                            nxsd:style="fixedLength"
                            nxsd:padStyle="tail" nxsd:paddedBy=" "
                            nxsd:length="15"/>
                          <!--COBOL declaration : 20 PO-CITY PIC X(10)-->
                          <xsd:element name="PO-CITY" type="xsd:string"
                            nxsd:style="fixedLength"
                            nxsd:padStyle="tail" nxsd:paddedBy=" "
                            nxsd:length="10"/>
                          <!--COBOL declaration : 20 PO-ZIP PIC 9(5)-->
                          <xsd:element name="PO-ZIP" type="xsd:long"
                            nxsd:style="fixedLength"
                            nxsd:padStyle="head" nxsd:paddedBy="0"
                            nxsd:length="5"/>
                          <!--COBOL declaration : 20 PO-STATE PIC X(2)-->

```

```

        <xsd:element name="PO-STATE" type="xsd:string"
            nxsd:style="fixedLength"
            nxsd:padStyle="tail" nxsd:paddedBy=" "
            nxsd:length="2"/>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<!--COBOL declaration : 10 PO-ITEM-->
<xsd:element name="PO-ITEM">
    <xsd:complexType>
        <xsd:sequence>
            <!--COBOL declaration : 15 POITEM OCCURS 3 TIMES-->
            <xsd:element name="POITEM" minOccurs="3" maxOccurs="3">
                <xsd:complexType>
                    <xsd:sequence>
                        <!--COBOL declaration : 20 PO-LINE-ITEM-->
                        <xsd:element name="PO-LINE-ITEM">
                            <xsd:complexType>
                                <xsd:sequence>
                                    <!--COBOL declaration : 25 PO-ITEM-ID PIC 9(3)-->
                                    <xsd:element name="PO-ITEM-ID" type="xsd:long"
                                        nxsd:style="fixedLength"
                                        nxsd:padStyle="head"
                                        nxsd:paddedBy="0" nxsd:length="3"/>
                                    <!--COBOL declaration : 25 PO-ITEM-NAME PIC X(40)-->
                                    <xsd:element name="PO-ITEM-NAME"
                                        type="xsd:string"
                                        nxsd:style="fixedLength"
                                        nxsd:padStyle="tail"
                                        nxsd:paddedBy=" " nxsd:length="40"/>
                                    <!--COBOL declaration : 25 PO-ITEM-QUANTITY PIC 9(2)-->
                                    <xsd:element name="PO-ITEM-QUANTITY"
                                        type="xsd:long"
                                        nxsd:style="fixedLength"
                                        nxsd:padStyle="head"
                                        nxsd:paddedBy="0" nxsd:length="2"/>
                                    <!--COBOL declaration : 25 PO-ITEM-PRICE PIC 9(5)V9(2)-->
                                    <xsd:element name="PO-ITEM-PRICE"
                                        type="xsd:decimal"
                                        nxsd:style="virtualDecimal"
                                        extn:assumeDecimal="5"
                                        extn:picSize="7"/>
                                </xsd:sequence>
                            </xsd:complexType>
                        </xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
<!--COBOL declaration : 10 PO-TOTAL PIC 9(7)V9(2)-->
<xsd:element name="PO-TOTAL" type="xsd:decimal"
    nxsd:style="virtualDecimal" extn:assumeDecimal="7"
    extn:picSize=" " />
</xsd:sequence>
</xsd:complexType>

```

```

    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>

```

Variable Length Array

```

05 EMP-RECORD .
  10 EMP-NAME          PIC X(30) .
  10 EMP-DIV-NUM       PIC 9(5) .
  10 DIV-ENTRY OCCURS 1 TO 50 TIMES
    DEPENDING ON EMP-DIV-NUM.
  20 DIV-CODE          PIC X(30) .

```

The generated schema looks as follows:

```

<?xml version="1.0" encoding="UTF-8" ?>
<!--Native format was generated from COBOL copybook : D:\work\
jDevProjects\CCB\Copybooks\odo.cpy-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  xmlns:extn="http://xmlns.oracle.com/pcbpel/nxsd/extensions"
  targetNamespace="http://TargetNamespace.com/ccb/varLengthArray"
  xmlns:tns="http://TargetNamespace.com/ccb/varLengthArray"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  nxsd:version="NXSD" nxsd:encoding="cp037" nxsd:byteOrder="bigEndian"
  nxsd:stream="chars">
  <xsd:element name="Root-Element">
    <xsd:complexType>
      <xsd:sequence>
        <!--COBOL declaration :05 EMP-RECORD -->
        <xsd:element name="EMP-RECORD" minOccurs="1" maxOccurs="unbounded">
          <xsd:annotation>
            <xsd:appinfo>
              <nxsd:variables>
                <nxsd:variable name="DIV-ENTRY_var0"/>
              </nxsd:variables>
            </xsd:appinfo>
          </xsd:annotation>
          <xsd:complexType>
            <xsd:sequence>
              <!--COBOL declaration : 10 EMP-NAME PIC X(30)-->
              <xsd:element name="EMP-NAME" type="xsd:string"
                nxsd:style="fixedLength" nxsd:padStyle="tail"
                nxsd:paddedBy=" " nxsd:length="30"/>
              <!--COBOL declaration : 10 EMP-DIV-NUM PIC 9(5)-->
              <xsd:element name="EMP-DIV-NUM" type="xsd:long"
                nxsd:style="fixedLength" nxsd:padStyle="head"
                nxsd:paddedBy="0" nxsd:length="5">
                <xsd:annotation>
                  <xsd:appinfo>
                    <nxsd:variables>
                      <nxsd:assign name="DIV-ENTRY_var0" value="{0}"/>
                    </nxsd:variables>
                  </xsd:appinfo>
                </xsd:annotation>
              </xsd:element>
            <!--COBOL declaration : 10 DIV-ENTRY OCCURS 1 TO 50 TIMES DEPENDING ON
EMP-DIV-NUM-->
            <xsd:element name="DIV-ENTRY" nxsd:style="array"

```

```

        nxsd:arrayLength="{DIV-ENTRY_var0}" minOccurs="1"
        maxOccurs="50">
<xsd:complexType>
<xsd:sequence>
<!--COBOL declaration : 20 DIV-CODE PIC X(30)-->
<xsd:element name="DIV-CODE" type="xsd:string"
        nxsd:style="fixedLength" nxsd:padStyle="tail"
        nxsd:paddedBy=" " nxsd:length="30"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>

```

Numeric Types

```

01  NUMERIC-FORMATS.
    05  Salary      PIC 9(5) COMP-3.
    05  Rating      PICTURE S9(5).
    05  Age         PIC 9(3) USAGE COMP.
    05  Revenue     PIC 9(3)V9(2).
    05  Growth      PIC S9(3) SIGN IS LEADING.
    05  Computation COMP-1.

```

The generated schema looks as following:

```

<?xml version="1.0" encoding="UTF-8" ?>
<!--Native format was generated from COBOL copybook :
D:\work\jDevProjects\CCB\Copybooks\numeric.cpy-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
        xmlns:extn="http://xmlns.oracle.com/pcbpel/nxsd/extensions"
        targetNamespace="http://TargetNamespace.com/ccb/numeric"
        xmlns:tns="http://TargetNamespace.com/ccb/numeric"
        elementFormDefault="qualified" attributeFormDefault="unqualified"
        nxsd:version="NXSD" nxsd:encoding="cp037" nxsd:byteOrder="bigEndian"
        nxsd:stream="bytes">
<xsd:element name="Numerics">
<xsd:complexType>
<xsd:sequence>
<!--COBOL declaration :01 NUMERIC-FORMATS-->
<xsd:element name="NUMERIC-FORMATS" minOccurs="1" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<!--COBOL declaration : 05 Salary PIC 9(5) COMP-3-->
<xsd:element name="Salary" type="xsd:long" nxsd:style="comp3"
        extn:sign="unticked" extn:picSize="5"/>
<!--COBOL declaration : 05 Rating PICTURE S9(5)-->
<xsd:element name="Rating" type="xsd:string"
        nxsd:style="signZoned" extn:sign="ticked"
        extn:picSize="5" extn:signPosn="tailUpperNibble"/>
<!--COBOL declaration : 05 Age PIC 9(3) USAGE COMP-->
<xsd:element name="Age" type="xsd:long" nxsd:style="comp"
        extn:picSize="3" extn:sign="unticked"/>
<!--COBOL declaration : 05 Revenue PIC 9(3)V9(2)-->
<xsd:element name="Revenue" type="xsd:decimal"

```

```
        nxsd:style="virtualDecimal" extn:assumeDecimal="3"
        extn:picSize="5"/>
<!--COBOL declaration : 05 Growth PIC S9(3) SIGN IS LEADING-->
<xsd:element name="Growth" type="xsd:string"
        nxsd:style="signZoned" extn:sign="ticked"
        extn:picSize="3" extn:signPosn="headUpperNibble"/>
<!--COBOL declaration : 05 Computation COMP-1-->
<xsd:element name="Computation" type="xsd:float"
        nxsd:style="comp1" extn:sign="ticked"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

In this case, all the numeric types follow formats specified according to IBM COBOL formats. If the data file originates from a different system using different layouts, then the generated schema requires modification.

7.2.2 Native Schema Constructs

This section contains the following topics:

- [Defining Fixed-Length Data](#)
- [Defining Terminated Data](#)
- [Defining Surrounded Data](#)
- [Defining Lists](#)
- [Defining Arrays](#)
- [Conditional Processing](#)
- [Defining Dates](#)
- [Using Variables](#)
- [Defining Prefixes and Suffixes](#)
- [Defining Skipping Data](#)
- [Defining fixed and default values](#)
- [Defining write](#)
- [Defining LookAhead](#)
- [Defining outboundHeader](#)

7.2.2.1 Defining Fixed-Length Data

Fixed-length data in the native format can be defined in the native schema using the fixed-length style. There are three types of fixed length:

- With padding
- Without padding
- With the actual length also being read from the native data

Native Data Format to Be Translated: With Padding

The actual data may be less than the length specified. In this case, you can specify the `paddedBy` and `padStyle` as head or tail. When the data is read, the pads are trimmed accordingly.

GBP*UK000012550.00

Native Schema: With Padding

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://www.oracle.com/ias/processconnect"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="fixedlength">
    <complexType>
      <sequence>
        <element name="currency_code" nxsd:style="fixedLength" nxsd:length="4"
          nxsd:padStyle="tail" nxsd:paddedBy="*">
          <simpleType>
            <restriction base="string">
              <maxLength value="4" />
            </restriction>
          </simpleType>
        </element>
        <element name="country_code" nxsd:style="fixedLength" nxsd:length="2"
          nxsd:padStyle="none">
          <simpleType>
            <restriction base="string">
              <length value="2" />
            </restriction>
          </simpleType>
        </element>
        <element name="to_usd_rate" nxsd:style="fixedLength" nxsd:length="12"
          nxsd:padStyle="head" nxsd:paddedBy="0">
          <simpleType>
            <restriction base="string">
              <maxLength value="12" />
            </restriction>
          </simpleType>
        </element>
      </sequence>
    </complexType>
  </element>

</schema>
```

Translated XML Using the Native Schema: With Padding

```
<fixedlength xmlns="http://www.oracle.com/ias/processconnect">
  <currency_code>GBP</currency_code>
  <country_code>UK</country_code>
  <to_usd_rate>12550.00</to_usd_rate>
</fixedlength>
```

Native Data Format to Be Translated: Without Padding

To define a fixed-length data in native schema, you can use the fixed-length style. In case the actual data is less than the length specified, the white spaces are not trimmed.

GBP*UK000012550.00

Native Schema: Without Padding

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://www.oracle.com/ias/processconnect"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="fixedlength">
    <complexType>
      <sequence>
        <element name="currency_code" nxsd:style="fixedLength" nxsd:length="4">
          <simpleType>
            <restriction base="string">
              <maxLength value="4" />
            </restriction>
          </simpleType>
        </element>
        <element name="country_code" nxsd:style="fixedLength" nxsd:length="2">
          <simpleType>
            <restriction base="string">
              <length value="2" />
            </restriction>
          </simpleType>
        </element>
        <element name="to_usd_rate" nxsd:style="fixedLength" nxsd:length="12">
          <simpleType>
            <restriction base="string">
              <maxLength value="12" />
            </restriction>
          </simpleType>
        </element>
      </sequence>
    </complexType>
  </element>

</schema>
```

Translated XML Using the Native Schema: Without Padding

```
<fixedlength xmlns="http://www.oracle.com/ias/processconnect">
  <currency_code>GBP*</currency_code>
  <country_code>UK</country_code>
  <to_usd_rate>000012550.00</to_usd_rate>
</fixedlength>
```

Native Data Format to Be Translated: Actual Length Also Being Read from the Native Data

When the length of the data is also stored in the native stream, this style is used to first read the length, and subsequently read the data according to the length read.

03joe13DUZac.1HKVmIY

Native Schema: Actual Length Also Being Read from the Native Data

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://www.oracle.com/ias/processconnect"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="fixedlength">
    <complexType>
      <sequence>
        <element name="user" type="string" nxsd:style="fixedLength"
          nxsd:identifierLength="2" />
        <element name="encr_user" type="string" nxsd:style="fixedLength"
          nxsd:identifierLength="2" />
      </sequence>
    </complexType>
  </element>

</schema>
```

Translated XML Using the Native Schema: Actual Length Also Being Read from the Native Data

```
<fixedlength xmlns="http://www.oracle.com/ias/processconnect">
  <user>joe</user>
  <encr_user>DUZac.1HKVmIY</encr_user>
</fixedlength>
```

7.2.2.2 Defining Terminated Data

This format is used when the terminating mark itself is supposed to be treated as part of the actual data and not as a delimiter. When it is not clear whether the mark is part of actual data or not, you can use the `nxsd:quotedBy` to be safe. Specifying `nxsd:quotedBy` means that the corresponding native data may or may not be quoted. If it is quoted, then the actual data is read from the begin quotation to the end quotation as specified in `nxsd:quotedBy`. Otherwise, it is read until the `terminatedBy` character is found.

The examples for the Optionally quoted and Not quoted scenarios are provided in following sections:

Native Data Format to Be Translated: Optionally Quoted

Fred,"2 Old Street, Old Town,Manchester",20-08-1954,0161-499-1718

Native Schema: Optionally Quoted

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://www.oracle.com/ias/processconnect"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="terminated">
    <complexType>
```

```
<sequence>
  <element name="PersonName" type="string" nxsd:style="terminated"
    nxsd:terminatedBy=", " />
  <element name="Address" type="string" nxsd:style="terminated"
    nxsd:terminatedBy=", " nxsd:quotedBy="&quot;" />
  <element name="DOB" type="string" nxsd:style="terminated"
    nxsd:terminatedBy=", " />
  <element name="Telephone" type="string" nxsd:style="terminated"
    nxsd:terminatedBy="{$eol}" />
</sequence>
</complexType>
</element>
```

Translated XML Using the Native Schema: Optionally Quoted

```
<terminated xmlns="http://www.oracle.com/ias/processconnect">
  <PersonName>Fred</PersonName>
  <Address>2 Old Street, Old Town,Manchester</Address>
  <DOB>20-08-1954</DOB>
  <Telephone>0161-499-1718</Telephone>
</terminated>
```

Native Data Format to Be Translated: Not Quoted

This is used when the data is terminated by a particular string or character.

1020,16,18,,1580.00

Native Schema: Not Quoted

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://www.oracle.com/ias/processconnect"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="terminated">
    <complexType>
      <sequence>
        <element name="product" type="string" nxsd:style="terminated"
          nxsd:terminatedBy=", " />
        <element name="ordered" type="string" nxsd:style="terminated"
          nxsd:terminatedBy=", " />
        <element name="inventory" type="string" nxsd:style="terminated"
          nxsd:terminatedBy=", " />
        <element name="backlog" type="string" nxsd:style="terminated"
          nxsd:terminatedBy=", " />
        <element name="listprice" type="string" nxsd:style="terminated"
          nxsd:terminatedBy="{$eol}" />
      </sequence>
    </complexType>
  </element>

</schema>
```

Translated XML Using the Native Schema: Not Quoted

```
<terminated xmlns="http://www.oracle.com/ias/processconnect">
  <product>1020</product>
```

```

<ordered>16</ordered>
<inventory>18</inventory>
<backlog></backlog>
<listprice>1580.00</listprice>
</terminated>

```

7.2.2.3 Defining Surrounded Data

This is used when the native data is surrounded by a mark.

The following examples are provided:

- Left and right surrounding marks are different
- Left and right surrounding marks are the same

Native Data Format to Be Translated: Left and Right Surrounding Marks Are Different

```
(Ernest Hemingway Museum){Whitehead St.}
```

Native Schema: Left and Right Surrounding Marks Are Different

```

<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  xmlns:tns="http://www.oracle.com/ias/processconnect"
  targetNamespace="http://www.oracle.com/ias/processconnect"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">
<element name="limstring">
  <complexType>
    <sequence>
      <element name="Landmark" type="string" nxsd:style="surrounded"
nxsd:leftSurroundedBy="(" nxsd:rightSurroundedBy=")" />
      <element name="Street" type="string" nxsd:style="surrounded"
nxsd:leftSurroundedBy="{ " nxsd:rightSurroundedBy="}" />
    </sequence>
  </complexType>
</element>
</schema>

```

Translated XML Using the Native Schema: Left and Right Surrounding Marks Are Different

```

<limstring xmlns="http://www.oracle.com/ias/processconnect">
  <Landmark>Ernest Hemingway Museum</Landmark>
  <Street>Whitehead St.</Street>
</limstring>

```

Native Data Format to Be Translated: Left and Right Surrounding Marks Are the Same

```
.FL..Florida Keys.+Key West+
```

Native Schema: Left and Right Surrounding Marks Are the Same

```

<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  xmlns:tns="http://www.oracle.com/ias/processconnect"

```

```

        targetNamespace="http://www.oracle.com/ias/processconnect"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified"
        nxsd:stream="chars"
        nxsd:version="NXSD">
<element name="limstring">
  <complexType>
    <sequence>
      <element name="State" type="string" nxsd:style="surrounded"
nxsd:surroundedBy="." />
      <element name="Region" type="string" nxsd:style="surrounded"
nxsd:surroundedBy="." />
      <element name="City" type="string" nxsd:style="surrounded"
nxsd:surroundedBy="+" />
    </sequence>
  </complexType>
</element>
</schema>

```

Translated XML Using the Native Schema: Left and Right Surrounding Marks Are the Same

```

<limstring xmlns="http://www.oracle.com/ias/processconnect">
  <State>FL</State>
  <Region>Florida Keys</Region>
  <City>Key West</City>
</limstring>

```

7.2.2.4 Defining Lists

This format applies to lists with the following characteristics:

- All Items Separated by the Same Mark, But the Last Item Terminated by a Different Mark (Bounded)
- All Items Separated by the Same Mark, Including the Last Item (Unbounded)

All Items Separated by the Same Mark, But the Last Item Terminated by a Different Mark (Bounded)

Following sections explain the format of the data to be translated, the native schema, and the translated XML.

Native Data Format to Be Translated:

125,200,255

Native Schema:

```

<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  xmlns:tns="http://xmlns.oracle.com/pcbpel/nxsd/smoketest"
  targetNamespace="http://xmlns.oracle.com/pcbpel/nxsd/smoketest"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="list" type="tns:Colors" />

```

```

<complexType name="Colors" nxsd:style="list" nxsd:itemSeparatedBy=","
  nxsd:listTerminatedBy="{eol}">

  <sequence>
    <element name="Red" type="string" />
    <element name="Green" type="string" />
    <element name="Blue" type="string" />
  </sequence>
</complexType>

</schema>

```

Translated XML Using the Native Schema:

```

<list xmlns="http://www.oracle.com/ias/processconnect">
  <Red>125</Red>
  <Green>200</Green>
  <Blue>255</Blue>
</list>

```

All Items Separated by the Same Mark, Including the Last Item (Unbounded)

Following sections explain the format of the data to be translated, the native schema, and the translated XML.

Native Data Format to Be Translated:

```
configure;startup;runttest;shutdown;
```

Native Schema:

```

<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  xmlns:tns="http://xmlns.oracle.com/pcbpel/nxsd/smoketest"
  targetNamespace="http://xmlns.oracle.com/pcbpel/nxsd/smoketest"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="list" type="tns:CommandSet" />

  <complexType name="CommandSet" nxsd:style="list" nxsd:itemSeparatedBy=";">
    <sequence>
      <element name="Cmd1" type="string" />
      <element name="Cmd2" type="string" />
      <element name="Cmd3" type="string" />
      <element name="Cmd4" type="string" />
    </sequence>
  </complexType>

</schema>

```

Translated XML Using the Native Schema:

```

<list xmlns="http://www.oracle.com/ias/processconnect">
  <Cmd1>configure</Cmd1>
  <Cmd2>startup</Cmd2>
  <Cmd3>runttest</Cmd3>
  <Cmd4>shutdown</Cmd4>

```

</list>

7.2.2.5 Defining Arrays

This is for an array of complex types where the individual cells are separated by a separating character and the last cell of the array is terminated by a terminating character.

The following examples are provided:

- [All Cells Separated by the Same Mark, But the Last Cell Terminated by a Different Mark \(Bounded\)](#)
- [All Items Separated by the Same Mark, Including the Last Item \(Unbounded\)](#)
- [Cells Not Separated by Any Mark, But the Last Cell Terminated by a Mark \(Bounded\)](#)
- [The Number of Cells Being Read from the Native Data](#)
- [Explicit Array Length](#)

All Cells Separated by the Same Mark, But the Last Cell Terminated by a Different Mark (Bounded)

Following sections explain the format of the data to be translated, the native schema, and the translated XML.

Native Data Format to Be Translated:

```
"Smith, John","1 Old Street, Old Town, Manchester",,"0161-499-1717".
Fred,"2 Old Street, Old Town,Manchester","20-08-1954","0161-499-1718".
"Smith, Bob",,,0161-499-1719.#
```

Native Schema:

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://www.oracle.com/ias/processconnect"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="array">
    <complexType>
      <sequence>
        <element name="Member" maxOccurs="unbounded"
          nxsd:style="array" nxsd:cellSeparatedBy="{eol}"
          nxsd:arrayTerminatedBy="#">
          <complexType>
            <sequence>
              <element name="Name" type="string" nxsd:style="terminated"
                nxsd:terminatedBy="," nxsd:quotedBy="'" />
              <element name="Address" type="string" nxsd:style="terminated"
                nxsd:terminatedBy="," nxsd:quotedBy="'" />
              <element name="DOB" type="string" nxsd:style="terminated"
                nxsd:terminatedBy="," nxsd:quotedBy="'" />
              <element name="Telephone" type="string" nxsd:style="terminated"
                nxsd:terminatedBy="." nxsd:quotedBy="'" />
            </sequence>
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>
</schema>
```



```

        </complexType>
      </element>
    </sequence>
  </complexType>
</element>

</schema>

```

Translated XML Using the Native Schema

```

<array xmlns="http://www.oracle.com/ias/processconnect">
  <Member>
    <Name>Smith, John</Name>
    <Address>1 Old Street, Old Town, Manchester</Address>
    <DOB></DOB>
    <Telephone>0161-499-1717</Telephone>
  </Member>
  <Member>
    <Name>Fred</Name>
    <Address>2 Old Street, Old Town, Manchester</Address>
    <DOB>20-08-1954</DOB>
    <Telephone>0161-499-1718</Telephone>
  </Member>
  <Member>
    <Name>Smith, Bob</Name>
    <Address></Address>
    <DOB></DOB>
    <Telephone>0161-499-1719</Telephone>
  </Member>
</array>

```

All Cells Separated by the Same Mark, Including the Last Cell (Unbounded)

Following sections explain the format of the data to be translated, the native schema, and the translated XML.

Native Data Format to Be Translated:

```

"Smith, John","1 Old Street, Old Town, Manchester",,"0161-499-1717".
Fred,"2 Old Street, Old Town, Manchester","20-08-1954","0161-499-1718".
"Smith, Bob",,"0161-499-1719".

```

Native Schema:

```

<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://www.oracle.com/ias/processconnect"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="array">
    <complexType>
      <sequence>
        <element name="Member" maxOccurs="unbounded"
          nxsd:style="array" nxsd:cellSeparatedBy="\r\n">
          <complexType>

```

```
<sequence>
  <element name="Name" type="string" nxsd:style="terminated"
    nxsd:terminatedBy="," nxsd:quotedBy="'" />
  <element name="Address" type="string" nxsd:style="terminated"
    nxsd:terminatedBy="," nxsd:quotedBy="'" />
  <element name="DOB" type="string" nxsd:style="terminated"
    nxsd:terminatedBy="," nxsd:quotedBy="'" />
  <element name="Telephone" type="string" nxsd:style="terminated"
    nxsd:terminatedBy="." nxsd:quotedBy="'" />
</sequence>
</complexType>
</element>
</sequence>
</complexType>
</element>

</schema>
```

Translated XML Using the Native Schema

```
<array xmlns="http://www.oracle.com/ias/processconnect">
  <Member>
    <Name>Smith, John</Name>
    <Address>1 Old Street, Old Town, Manchester</Address>
    <DOB></DOB>
    <Telephone>0161-499-1717</Telephone>
  </Member>
  <Member>
    <Name>Fred</Name>
    <Address>2 Old Street, Old Town,Manchester</Address>
    <DOB>20-08-1954</DOB>
    <Telephone>0161-499-1718</Telephone>
  </Member>
  <Member>
    <Name>Smith, Bob</Name>
    <Address></Address>
    <DOB></DOB>
    <Telephone>0161-499-1719</Telephone>
  </Member>
</array>
```

Cells Not Separated by Any Mark, But the Last Cell Terminated by a Mark (Bounded)

Following sections explain the format of the data to be translated, the native schema, and the translated XML.

Native Data Format to Be Translated:

```
"Smith, John","1 Old Street, Old Town, Manchester",,"0161-499-1717"
Fred,"2 Old Street, Old Town,Manchester","20-08-1954","0161-499-1718"
"Smith, Bob",,,0161-499-1719
#
```

Native Schema:

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
```

```

xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
targetNamespace="http://www.oracle.com/ias/processconnect"
elementFormDefault="qualified"
attributeFormDefault="unqualified"
nxsd:stream="chars"
nxsd:version="NXSD">

<element name="array">
  <complexType>
    <sequence>
      <element name="Member" maxOccurs="unbounded"
        nxsd:style="array" nxsd:arrayTerminatedBy="#">
        <complexType>
          <sequence>
            <element name="Name" type="string" nxsd:style="terminated"
              nxsd:terminatedBy="," nxsd:quotedBy="'" />
            <element name="Address" type="string" nxsd:style="terminated"
              nxsd:terminatedBy="," nxsd:quotedBy="'" />
            <element name="DOB" type="string" nxsd:style="terminated"
              nxsd:terminatedBy="," nxsd:quotedBy="'" />
            <element name="Telephone" type="string" nxsd:style="terminated"
              nxsd:terminatedBy="\r\n" nxsd:quotedBy="'" />
          </sequence>
        </complexType>
      </element>
    </sequence>
  </complexType>
</element>

</schema>

```

Translated XML Using the Native Schema

```

<array xmlns="http://www.oracle.com/ias/processconnect">
  <Member>
    <Name>Smith, John</Name>
    <Address>1 Old Street, Old Town, Manchester</Address>
    <DOB></DOB>
    <Telephone>0161-499-1717</Telephone>
  </Member>
  <Member>
    <Name>Fred</Name>
    <Address>2 Old Street, Old Town, Manchester</Address>
    <DOB>20-08-1954</DOB>
    <Telephone>0161-499-1718</Telephone>
  </Member>
  <Member>
    <Name>Smith, Bob</Name>
    <Address></Address>
    <DOB></DOB>
    <Telephone>0161-499-1719</Telephone>
  </Member>
</array>

```

The Number of Cells Being Read from the Native Data

Following sections explain the format of the data to be translated, the native schema, and the translated XML.

Native Data Format to Be Translated:

```
3"Smith, John","1 Old Street, Old Town, Manchester",,"0161-499-1717"
Fred,"2 Old Street, Old Town,Manchester","20-08-1954","0161-499-1718"
"Smith, Bob",,,0161-499-1719
```

Native Schema:

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://www.oracle.com/ias/processconnect"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="arrayidentifierlength">
    <complexType>
      <sequence>
        <element name="Member" maxOccurs="unbounded" nxsd:style="array"
          nxsd:arrayIdentifierLength="1">
          <complexType>
            <sequence>
              <element name="Name" type="string" nxsd:style="terminated"
                nxsd:terminatedBy="," nxsd:quotedBy="'"'/>
              <element name="Address" type="string" nxsd:style="terminated"
                nxsd:terminatedBy="," nxsd:quotedBy="'"'/>
              <element name="DOB" type="string" nxsd:style="terminated"
                nxsd:terminatedBy="," nxsd:quotedBy="'"'/>
              <element name="Telephone" type="string" nxsd:style="terminated"
                nxsd:terminatedBy="\r\n" nxsd:quotedBy="'"'/>
            </sequence>
          </complexType>
        </element>
      </sequence>
    </complexType>
  </element>

</schema>
```

Translated XML Using the Native Schema

```
<arrayidentifierlength xmlns="http://www.oracle.com/ias/processconnect">
  <Member>
    <Name>Smith, John</Name>
    <Address>1 Old Street, Old Town, Manchester</Address>
    <DOB></DOB>
    <Telephone>0161-499-1717</Telephone>
  </Member>
  <Member>
    <Name>Fred</Name>
    <Address>2 Old Street, Old Town,Manchester</Address>
    <DOB>20-08-1954</DOB>
    <Telephone>0161-499-1718</Telephone>
  </Member>
  <Member>
    <Name>Smith, Bob</Name>
    <Address></Address>
    <DOB></DOB>
    <Telephone>0161-499-1719</Telephone>
  </Member>
</arrayidentifierlength>
```

```

    </Member>
  </arrayidentifierlength>

```

Explicit Array Length

Following sections explain the format of the data to be translated, the native schema, and the translated XML.

Native Data Format to Be Translated:

```
3;John;Steve;Paul;Todd;
```

Native Schema:

```

<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://xmlns.oracle.com/pcbpel/nxsd/smoketest"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="array">
    <annotation>
      <appinfo>
        <nxsd:variables>
          <nxsd:variable name="len" />
        </nxsd:variables>
      </appinfo>
    </annotation>

    <complexType>
      <sequence>
        <element name="TotalMembers" type="string" nxsd:style="terminated"
          nxsd:terminatedBy=";" />
        <annotation>
          <appinfo>
            <nxsd:variables>
              <nxsd:assign name="len" value="{0}" />
            </nxsd:variables>
          </appinfo>
        </annotation>
      </element>
      <element name="Member" type="string" minOccurs="0" maxOccurs="unbounded"
        nxsd:style="array,terminated" nxsd:arrayLength="{len}"
        nxsd:terminatedBy=";" />
      </sequence>
    </complexType>
  </element>

</schema>

```

Translated XML Using the Native Schema

```

<array xmlns="http://xmlns.oracle.com/pcbpel/nxsd/smoketest">
  <TotalMembers>3</TotalMembers>
  <Member>John</Member>
  <Member>Steve</Member>

```

```
<Member>Paul</Member>
</array>
```

7.2.2.6 Conditional Processing

This section provides the following examples of conditional processing:

- [Processing One Element Within a Choice Model Group Based on the Condition](#)
- Processing elements based within a sequence model group on the condition

Processing One Element Within a Choice Model Group Based on the Condition

Following sections explain the format of the data to be translated, the native schema, and the translated XML.

Native Data Format to Be Translated:

```
PO28/06/2004^|ABCD Inc.|Oracle
OracleApps025070,000.00
Database 021230,000.00
ProcessCon021040,000.00
PO01/07/2004^|EFGH Inc.|Oracle
Websphere 025070,000.00
DB2 021230,000.00
Eclipse 021040,000.00
SO29/06/2004|Oracle Apps|5
Navneet Singh
PO28/06/2004^|IJKL Inc.|Oracle
Weblogic 025070,000.00
Tuxedo 021230,000.00
JRockit 021040,000.00
IN30/06/2004;Navneet Singh;Oracle;Oracle Apps;5;70,000.00;350,000.00
```

Native Schema:

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  xmlns:tns="http://www.oracle.com/ias/processconnect"
  targetNamespace="http://www.oracle.com/ias/processconnect"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="container">

    <complexType>
      <choice maxOccurs="unbounded" nxsd:choiceCondition="fixedLength"
        nxsd:length="2">

        <element ref="tns:PurchaseOrder" nxsd:conditionValue="PO" />

        <element ref="tns:SalesOrder" nxsd:conditionValue="SO" />

        <element ref="tns:Invoice" nxsd:conditionValue="IN" />

      </choice>
    </complexType>
  </element>
```

```

<!-- PO -->
<element name="PurchaseOrder" type="tns:POType" />

<complexType name="POType">
  <sequence>

    <element name="Date" type="string" nxsd:style="terminated"
      nxsd:terminatedBy="^" />
    <element name="Buyer" type="string" nxsd:style="surrounded"
      nxsd:surroundedBy="|" />
    <element name="Supplier" type="string" nxsd:style="terminated"
      nxsd:terminatedBy="{$eol}" />
    <element name="Items">
      <complexType>
        <sequence>
          <element name="Line-Item" minOccurs="3" maxOccurs="3">
            <complexType>
              <group ref="tns:LineItems" />
            </complexType>
          </element>
        </sequence>
      </complexType>
    </element>
  </sequence>
</complexType>

<group name="LineItems">
  <sequence>
    <element name="Id" type="string" nxsd:style="fixedLength" nxsd:length="10"
      nxsd:padStyle="none" />
    <element name="Quantity" type="string" nxsd:style="fixedLength"
      nxsd:identifierLength="2" />
    <element name="Price" type="string" nxsd:style="terminated"
      nxsd:terminatedBy="{$eol}" />
  </sequence>
</group>

<!-- SO -->
<element name="SalesOrder" type="tns:SOType" />

<complexType name="SOType">
  <sequence>
    <element name="Date" type="string" nxsd:style="terminated"
      nxsd:terminatedBy="|" />
    <element name="Item" type="string" nxsd:style="terminated"
      nxsd:terminatedBy="|" />
    <element name="Quantity" type="string" nxsd:style="terminated"
      nxsd:terminatedBy="{$eol}" />
    <element name="Buyer" type="string" nxsd:style="terminated"
      nxsd:terminatedBy="{$eol}" />
  </sequence>
</complexType>

<!-- INV -->
<element name="Invoice" type="tns:INVType" />

<complexType name="INVType">
  <sequence>
    <element name="Date" type="string" nxsd:style="terminated"

```

```
        nxsd:terminatedBy=";" />
    <element name="Purchaser" type="string" nxsd:style="terminated"
        nxsd:terminatedBy=";" />
    <element name="Seller" type="string" nxsd:style="terminated"
        nxsd:terminatedBy=";" />
    <element name="Item" type="string" nxsd:style="terminated"
        nxsd:terminatedBy=";" />
    <element name="Price" type="string" nxsd:style="terminated"
        nxsd:terminatedBy=";" />
    <element name="Quantity" type="string" nxsd:style="terminated"
        nxsd:terminatedBy=";" />
    <element name="TotalPrice" type="string" nxsd:style="terminated"
        nxsd:terminatedBy=" ${eol}" />
</sequence>
</complexType>

</schema>
```

Translated XML Using the Native Schema

```
<container xmlns="http://www.oracle.com/ias/processconnect">
  <PurchaseOrder>
    <Date>28/06/2004</Date>
    <Buyer>ABCD Inc.</Buyer>
    <Supplier>Oracle</Supplier>
    <Items>
      <Line-Item>
        <Id>OracleApps</Id>
        <Quantity>50</Quantity>
        <Price>70,000.00</Price>
      </Line-Item>
      <Line-Item>
        <Id>Database </Id>
        <Quantity>12</Quantity>
        <Price>30,000.00</Price>
      </Line-Item>
      <Line-Item>
        <Id>ProcessCon</Id>
        <Quantity>10</Quantity>
        <Price>40,000.00</Price>
      </Line-Item>
    </Items>
  </PurchaseOrder>
  <PurchaseOrder>
    <Date>01/07/2004</Date>
    <Buyer>EFGH Inc.</Buyer>
    <Supplier>Oracle</Supplier>
    <Items>
      <Line-Item>
        <Id>Websphere </Id>
        <Quantity>50</Quantity>
        <Price>70,000.00</Price>
      </Line-Item>
      <Line-Item>
        <Id>DB2 </Id>
        <Quantity>12</Quantity>
        <Price>30,000.00</Price>
      </Line-Item>
      <Line-Item>
        <Id>Eclipse </Id>
        <Quantity>10</Quantity>
```



```

        <Price>40,000.00</Price>
    </Line-Item>
</Items>
</PurchaseOrder>
<SalesOrder>
    <Date>29/06/2004</Date>
    <Item>Oracle Apps</Item>
    <Quantity>5</Quantity>
    <Buyer>Navneet Singh</Buyer>
</SalesOrder>
<PurchaseOrder>
    <Date>28/06/2004</Date>
    <Buyer>IJKL Inc.</Buyer>
    <Supplier>Oracle</Supplier>
    <Items>
        <Line-Item>
            <Id>Weblogic </Id>
            <Quantity>50</Quantity>
            <Price>70,000.00</Price>
        </Line-Item>
        <Line-Item>
            <Id>Tuxedo </Id>
            <Quantity>12</Quantity>
            <Price>30,000.00</Price>
        </Line-Item>
        <Line-Item>
            <Id>JRockit </Id>
            <Quantity>10</Quantity>
            <Price>40,000.00</Price>
        </Line-Item>
    </Items>
</PurchaseOrder>
<Invoice>
    <Date>30/06/2004</Date>
    <Purchaser>Navneet Singh</Purchaser>
    <Seller>Oracle</Seller>
    <Item>Oracle Apps</Item>
    <Price>5</Price>
    <Quantity>70,000.00</Quantity>
    <TotalPrice>350,000.00</TotalPrice>
</Invoice>
</container>

```

Processing Elements Within a Sequence Model Group Based on the Condition

Following sections explain the format of the data to be translated, the native schema, and the translated XML.

Native Data Format to Be Translated:

```

PO28/06/2004^|ABCD Inc.|Oracle
OracleApps025070,000.00
Database 021230,000.00
ProcessCon021040,000.00
PO01/07/2004^|EFGH Inc.|Oracle
Websphere 025070,000.00
DB2 021230,000.00
Eclipse 021040,000.00
SO29/06/2004|Oracle Apps|5
Navneet Singh
PO28/06/2004^|IJKL Inc.|Oracle

```

```
Weblogic 025070,000.00
Tuxedo 021230,000.00
JRockit 021040,000.00
IN30/06/2004;Navneet Singh;Oracle;Oracle Apps;5;70,000.00;350,000.00
```

Native Schema:

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  xmlns:tns="http://www.oracle.com/ias/processconnect"
  targetNamespace="http://www.oracle.com/ias/processconnect"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="container">

    <complexType>
      <sequence maxOccurs="unbounded">

        <element ref="tns:PurchaseOrder" minOccurs="0" nxsd:startsWith="PO" />

        <element ref="tns:SalesOrder" minOccurs="0" nxsd:startsWith="SO" />

        <element ref="tns:Invoice" minOccurs="0" nxsd:startsWith="IN" />

      </sequence>
    </complexType>
  </element>

  <!-- PO -->
  <element name="PurchaseOrder" type="tns:POType"/>

  <complexType name="POType">
    <sequence>

      <element name="Date" type="string" nxsd:style="terminated"
        nxsd:terminatedBy="^" />

      <element name="Buyer" type="string" nxsd:style="surrounded"
        nxsd:surroundedBy="|" />
      <element name="Supplier" type="string" nxsd:style="terminated"
        nxsd:terminatedBy="{eol}" />
      <element name="Items">
        <complexType>
          <sequence>
            <element name="Line-Item" minOccurs="3" maxOccurs="3">
              <complexType>
                <group ref="tns:LineItems" />
              </complexType>
            </element>
          </sequence>
        </complexType>
      </element>
    </sequence>
  </complexType>

  <group name="LineItems">
    <sequence>
```

```

        <element name="Id" type="string" nxsd:style="fixedLength" nxsd:length="10"
            nxsd:padStyle="none"/>
        <element name="Quantity" type="string" nxsd:style="fixedLength"
            nxsd:identifierLength="2" />
        <element name="Price" type="string" nxsd:style="terminated"
            nxsd:terminatedBy="{eol}" />
    </sequence>
</group>

<!-- SO -->
<element name="SalesOrder" type="tns:SOType" />

<complexType name="SOType">
    <sequence>
        <element name="Date" type="string" nxsd:style="terminated"
            nxsd:terminatedBy="|" />
        <element name="Item" type="string" nxsd:style="terminated"
            nxsd:terminatedBy="|" />
        <element name="Quantity" type="string" nxsd:style="terminated"
            nxsd:terminatedBy="{eol}" />
        <element name="Buyer" type="string" nxsd:style="terminated"
            nxsd:terminatedBy="{eol}" />
    </sequence>
</complexType>

<!-- INV -->
<element name="Invoice" type="tns:INVType" />

<complexType name="INVType">
    <sequence>
        <element name="Date" type="string" nxsd:style="terminated"
            nxsd:terminatedBy=";" />
        <element name="Purchaser" type="string" nxsd:style="terminated"
            nxsd:terminatedBy=";" />
        <element name="Seller" type="string" nxsd:style="terminated"
            nxsd:terminatedBy=";" />
        <element name="Item" type="string" nxsd:style="terminated"
            nxsd:terminatedBy=";" />
        <element name="Price" type="string" nxsd:style="terminated"
            nxsd:terminatedBy=";" />
        <element name="Quantity" type="string" nxsd:style="terminated"
            nxsd:terminatedBy=";" />
        <element name="TotalPrice" type="string" nxsd:style="terminated"
            nxsd:terminatedBy="{eol}" />
    </sequence>
</complexType>

</schema>

```

Translated XML Using the Native Schema:

```

<container xmlns="http://www.oracle.com/ias/processconnect">
    <PurchaseOrder>
        <Date>28/06/2004</Date>
        <Buyer>ABCD Inc.</Buyer>
        <Supplier>Oracle</Supplier>
        <Items>
            <Line-Item>
                <Id>OracleApps</Id>
                <Quantity>50</Quantity>
                <Price>70,000.00</Price>
            </Line-Item>
        </Items>
    </PurchaseOrder>
</container>

```

```
</Line-Item>
<Line-Item>
  <Id>Database </Id>
  <Quantity>12</Quantity>
  <Price>30,000.00</Price>
</Line-Item>
<Line-Item>
  <Id>ProcessCon</Id>
  <Quantity>10</Quantity>
  <Price>40,000.00</Price>
</Line-Item>
</Items>
</PurchaseOrder>
<PurchaseOrder>
  <Date>01/07/2004</Date>
  <Buyer>EFGH Inc.</Buyer>
  <Supplier>Oracle</Supplier>
  <Items>
    <Line-Item>
      <Id>Websphere </Id>
      <Quantity>50</Quantity>
      <Price>70,000.00</Price>
    </Line-Item>
    <Line-Item>
      <Id>DB2 </Id>
      <Quantity>12</Quantity>
      <Price>30,000.00</Price>
    </Line-Item>
    <Line-Item>
      <Id>Eclipse </Id>
      <Quantity>10</Quantity>
      <Price>40,000.00</Price>
    </Line-Item>
  </Items>
</PurchaseOrder>
<SalesOrder>
  <Date>29/06/2004</Date>
  <Item>Oracle Apps</Item>
  <Quantity>5</Quantity>
  <Buyer>Navneet Singh</Buyer>
</SalesOrder>
<PurchaseOrder>
  <Date>28/06/2004</Date>
  <Buyer>IJKL Inc.</Buyer>
  <Supplier>Oracle</Supplier>
  <Items>
    <Line-Item>
      <Id>Weblogic </Id>
      <Quantity>50</Quantity>
      <Price>70,000.00</Price>
    </Line-Item>
    <Line-Item>
      <Id>Tuxedo </Id>
      <Quantity>12</Quantity>
      <Price>30,000.00</Price>
    </Line-Item>
    <Line-Item>
      <Id>JRockit </Id>
      <Quantity>10</Quantity>
      <Price>40,000.00</Price>
    </Line-Item>
  </Items>
</PurchaseOrder>
```

```

        </Line-Item>
    </Items>
</PurchaseOrder>
<Invoice>
    <Date>30/06/2004</Date>
    <Purchaser>Navneet Singh</Purchaser>
    <Seller>Oracle</Seller>
    <Item>Oracle Apps</Item>
    <Price>5</Price>
    <Quantity>70,000.00</Quantity>
    <TotalPrice>350,000.00</TotalPrice>
</Invoice>
</container>

```

7.2.2.7 Defining Dates

This example shows how to define dates.

Native Data Format to Be Translated

```

11/16/0224/11/02
11-20-2002
23*11*2002
01/02/2003 01:02
01/02/2003 03:04:05

```

Native Schema

```

<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://xmlns.oracle.com/pcbpel/nxsd/smoketest"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="dateformat">
    <complexType>
      <sequence>
        <element name="StartDate" type="dateTime" nxsd:dateFormat="MM/dd/yy"
          nxsd:style="fixedLength" nxsd:length="8" />
        <element name="EndDate" type="dateTime" nxsd:dateFormat="dd/MM/yy"
          nxsd:style="terminated" nxsd:terminatedBy="{eol}" />
        <element name="Milestone" type="dateTime" nxsd:dateFormat="MM-dd-yyyy"
          nxsd:style="terminated" nxsd:terminatedBy="{eol}" />
        <element name="DueDate" type="dateTime" nxsd:dateFormat="dd*MM*yyyy"
          nxsd:style="terminated" nxsd:terminatedBy="{eol}" />
        <element name="Date" type="dateTime" nxsd:dateFormat="MM/dd/yyyy hh:mm"
          nxsd:style="terminated" nxsd:terminatedBy="{eol}" />
        <element name="Date" type="dateTime" nxsd:dateFormat="MM/dd/yyyy hh:mm:ss"
          nxsd:style="terminated" nxsd:terminatedBy="{eol}" />
      </sequence>
    </complexType>
  </element>

</schema>

```

Translated XML Using the Native Schema

```

<dateformat xmlns="http://xmlns.oracle.com/pcbpel/nxsd/smoketest">
  <StartDate>2002-11-16T00:00:00</StartDate>

```

```
<EndDate>2002-11-24T00:00:00</EndDate>
<Milestone>2002-11-20T00:00:00</Milestone>
<DueDate>2002-11-23T00:00:00</DueDate>
<Date>2003-01-02T01:02:00</Date>
<Date>2003-01-02T03:04:05</Date>
</dateformat>
```

Note: nxsd:dateParsingMode="lax/strict" and locale support have been added to the existing date format.

The following example depicts the use of nxsd:dateParsingMode="lax/strict" and locale support.

Native Data Format to Be Translated

```
11/16/0224/11/02
11-20-2002
23*11*2002
01/02/2003 01:02
01/02/2003 03:04:05
Thu, 26 May 2005 15:50:11 India Standard Time
Do, 26 Mai 2005 15:43:10 Indische Normalzeit
20063202
```

Native Schema

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://xmlns.oracle.com/pcbpel/nxsd/smoketest"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="dateformat">
    <complexType>
      <sequence>
        <element name="StartDate" type="date" nxsd:dateFormat="MM/dd/yy"
          nxsd:localeLanguage="en" nxsd:style="fixedLength" nxsd:length="8" />
        <element name="EndDate" type="date" nxsd:dateFormat="dd/MM/yy"
          nxsd:style="terminated" nxsd:terminatedBy="{eol}" />
        <element name="Milestone" type="dateTime" nxsd:dateFormat="MM-dd-yyyy"
          nxsd:style="terminated" nxsd:terminatedBy="{eol}" />
        <element name="DueDate" type="dateTime" nxsd:dateFormat="dd*MM*yyyy"
          nxsd:style="terminated" nxsd:terminatedBy="{eol}" />

        <element name="Date" type="dateTime" nxsd:dateFormat="MM/dd/yyyy hh:mm"
          nxsd:style="terminated" nxsd:terminatedBy="{eol}" />
        <element name="Date" type="dateTime" nxsd:dateFormat="MM/dd/yyyy hh:mm:ss"
          nxsd:style="terminated" nxsd:terminatedBy="{eol}" />

        <element name="LongDateInEnglish" type="dateTime" nxsd:dateFormat="EEE, d
          MMM yyyy HH:mm:ss zzzz" nxsd:localeLanguage="en" nxsd:localeCountry="US"
          nxsd:style="terminated" nxsd:terminatedBy="{eol}" />
        <element name="LongDateInGerman" type="dateTime" nxsd:dateFormat="EEE, d
          MMM yyyy HH:mm:ss zzzz" nxsd:localeLanguage="de" nxsd:style="terminated"
          nxsd:terminatedBy="{eol}" />
      </sequence>
    </complexType>
  </element>
</schema>
```

```

        <element name="InvalidDate" type="dateTime"    nxsd:dateParsingMode="lax"
nxsd:dateFormat="yyyyMMdd" nxsd:style="terminated" nxsd:terminatedBy="{eol}" />

    </sequence>
</complexType>
</element>

</schema>

```

Translated XML

```

<dateformat xmlns="http://xmlns.oracle.com/pcbpel/nxsd/smoketest">
  <StartDate>2002-11-16</StartDate>
  <EndDate>2002-11-24</EndDate>
  <Milestone>2002-11-20T00:00:00</Milestone>
  <DueDate>2002-11-23T00:00:00</DueDate>
  <Date>2003-01-02T01:02:00</Date>
  <Date>2003-01-02T03:04:05</Date>
  <LongDateInEnglish>2005-05-26T15:50:11</LongDateInEnglish>
  <LongDateInGerman>2005-05-26T15:43:10</LongDateInGerman>
  <InvalidDate>2008-08-02T00:00:00</InvalidDate>
</dateformat>

```

7.2.2.8 Using Variables

This example shows how to use variables.

Native Data Format to Be Translated

```

{,;}Fred,"2 Old Street, Old Town,Manchester","20-08-1954";"0161-499-1718"
phone-2
phone-3

```

Native Schema

```

<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://xmlns.oracle.com/pcbpel/nxsd/smoketest"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="variable">
    <annotation>
      <documentation>
        1. var1 - variable declaration
        2. var2 - variable declaration with default value
        3. EOL - variable declaration with referencing a system variable
      </documentation>
    <appinfo>
      <junkies/>
      <nxsd:variables>
        <nxsd:variable name="var1" />
        <nxsd:variable name="var2" value=", " />
        <nxsd:variable name="SystemEOL" value="{system.line.separator}" />
      </nxsd:variables>
    </appinfo>
  </element>

```

```

        <junkies/>
        <junkies/>
        <junkies/>
    </appinfo>
</annotation>

<complexType>
  <sequence>
    <element name="delims" type="string" nxsd:style="surrounded"
      nxsd:leftSurroundedBy="{ " nxsd:rightSurroundedBy="}" >
      <annotation>
        <appinfo>
          <junkies/>
          <junkies/>
          <junkies/>
          <nxsd:variables>
            <nxsd:assign name="var1" value="{0,1}"/>
            <nxsd:assign name="var2" value="{1}" />
          </nxsd:variables>
        </appinfo>
      </annotation>
    </element>

    <element name="PersonName" type="string" nxsd:style="terminated"
      nxsd:terminatedBy="{var1}" nxsd:quotedBy="&quot;" />
    <element name="Address" type="string" nxsd:style="terminated"
      nxsd:terminatedBy="{var1}" nxsd:quotedBy="&quot;" />
    <element name="DOB" type="string" nxsd:style="terminated"
      nxsd:terminatedBy="{var2}" nxsd:quotedBy="'" />
    <element name="Telephone1" type="string" nxsd:style="terminated"
      nxsd:terminatedBy="{eol}" nxsd:quotedBy="'" />
    <element name="Telephone2" type="string" nxsd:style="terminated"
      nxsd:terminatedBy="{eol}" nxsd:quotedBy="'" />
    <element name="Telephone3" type="string" nxsd:style="terminated"
      nxsd:terminatedBy="{eol}" nxsd:quotedBy="'" />
  </sequence>
</complexType>
</element>

</schema>

```

Translated XML Using the Native Schema

```

<variable xmlns="http://xmlns.oracle.com/pcbpel/nxsd/smoketest">
  <delims>,</delims>
  <PersonName>Fred</PersonName>
  <Address>2 Old Street, Old Town,Manchester</Address>
  <DOB>20-08-1954</DOB>
  <Telephone1>0161-499-1718</Telephone1>
  <Telephone2>phone-2</Telephone2>
  <Telephone3>phone-3</Telephone3>
</variable>

```

7.2.2.9 Defining Prefixes and Suffixes

In native format; when the data is read, it prefixes or suffixes data specified with prefix, suffix, or both, as shown in the following example.

Native Data to Be Translated

```
Fred,"2 Old Street, Old Town,Manchester","20-08-1954",0161-499-1718
```


Native Schema

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://xmlns.oracle.com/pcbpel/nxsd/smoketest"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD"
>

<element name="terminated">
  <complexType>
    <sequence>
      <element name="PersonName" type="string" nxsd:prefixWith="Mr."
nxsd:style="terminated" nxsd:terminatedBy=", " nxsd:quotedBy=""" />
      <element name="Address" type="string" nxsd:suffixWith="]]"
nxsd:prefixWith="[[ " nxsd:style="terminated" nxsd:terminatedBy=", "
nxsd:quotedBy=""" />
      <element name="DOB" type="string" nxsd:style="terminated"
nxsd:terminatedBy=", " nxsd:quotedBy="'" />
      <element name="Telephone" type="string" nxsd:style="terminated"
nxsd:terminatedBy="}${eol}" nxsd:quotedBy="'" />
    </sequence>
  </complexType>
</element>

</schema>
```

Translated XML Using the Native Schema

```
<terminated xmlns="http://xmlns.oracle.com/pcbpel/nxsd/smoketest">
  <PersonName>Mr.Fred</PersonName>
  <Address>[[2 Old Street, Old Town,Manchester]]</Address>
  <DOB>20-08-1954</DOB>
  <Telephone>0161-499-1718</Telephone>
</terminated>
```

7.2.2.10 Defining Skipping Data

The translator will skip, before or after the data is read depending on the skipMode, as shown in the following example:

Native Data to Be Translated

```
Fred,"2 Old Street, Old Town,Manchester","20-08-1954",0161-499-1718
```

Native Schema

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://xmlns.oracle.com/pcbpel/nxsd/smoketest"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
```

```
        nxsd:version="NXSD"
      >

<element name="terminated">
  <complexType>
    <sequence>
      <element name="PersonName" type="string" nxsd:skip="5"
nxsd:style="terminated" nxsd:terminatedBy="," nxsd:quotedBy="&quot;" />
      <element name="Address" type="string" nxsd:skipMode="before" nxsd:skip="3"
nxsd:style="terminated" nxsd:terminatedBy="," nxsd:quotedBy="&quot;" />
      <element name="DOB" type="string" nxsd:skipMode="after" nxsd:skip="6"
nxsd:style="terminated" nxsd:terminatedBy="," nxsd:quotedBy="'" />
      <element name="Telephone" type="string" nxsd:style="terminated"
nxsd:terminatedBy="{eol}" nxsd:quotedBy="'" />
    </sequence>
  </complexType>
</element>

</schema>
```

Translated XML Using Native Schema

```
<terminated xmlns="http://xmlns.oracle.com/pcbpel/nxsd/smoketest">
  <PersonName>Fred</PersonName>
  <Address>2 Old Street, Old Town,Manchester</Address>
  <DOB>20-08-1954</DOB>
  <Telephone>99-1718</Telephone>
</terminated>
```

7.2.2.11 Defining fixed and default values

When an element is declared without nxsd annotations, but the values specified is either, fixed or default, then the translator should just use that value and move on, and not throw any exception.

Native Data to Be Translated

Fred,"2 Old Street, Old Town,Manchester","20-08-1954","0161-499-1718"

Native Schema

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://xmlns.oracle.com/pcbpel/nxsd/smoketest"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="terminated">
    <annotation>
      <appinfo>
        <nxsd:variables>
          <nxsd:variable name="x" value="hello" />
        </nxsd:variables>
        <junkies/>
        <junkies/>
        <junkies/>
      </appinfo>
    </annotation>
  </element>
```

```

    </appinfo>
  </annotation>

  <complexType>
    <sequence>
      <element name="PersonName" type="string" nxsd:style="terminated"
nxsd:terminatedBy="," nxsd:quotedBy="&quot;" />
      <element name="Age" type="string" fixed="16" />
      <element name="Address" type="string" nxsd:style="terminated"
nxsd:terminatedBy="," nxsd:quotedBy="&quot;" />
      <element name="DOB" type="string" nxsd:style="terminated"
nxsd:terminatedBy="," nxsd:quotedBy="'" />
      <element name="salutation" type="string" default="{x}" />
      <element name="Telephone" type="string" nxsd:style="terminated"
nxsd:terminatedBy="{eol}" nxsd:quotedBy="'" />
    </sequence>
  </complexType>
</element>

</schema>

```

Translated XML Using Native Schema

```

<terminated xmlns="http://xmlns.oracle.com/pcbpel/nxsd/smoketest">
  <PersonName>Fred</PersonName>
  <Age>16</Age>
  <Address>2 Old Street, Old Town,Manchester</Address>
  <DOB>20-08-1954</DOB>
  <salutation>hello</salutation>
  <Telephone>0161-499-1718</Telephone>
</terminated>

```

7.2.2.12 Defining write

`write` writes the literal at the current position in the output stream, either before writing the actual data or after writing it.

Input XML

```

<terminated xmlns="http://xmlns.oracle.com/pcbpel/nxsd/smoketest">
  <PersonName>Fred</PersonName>
  <Address>2 Old Street, Old Town,Manchester</Address>
  <DOB>20-08-1954</DOB>
  <Telephone>0161-499-1718</Telephone>
</terminated>

```

Native Schema

```

<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://xmlns.oracle.com/pcbpel/nxsd/smoketest"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD"
>

<element name="terminated">

```

```
<complexType>
  <sequence>
    <element name="PersonName" type="string" nxsd:writeMode="before"
nxsd:write="Mr." nxsd:style="terminated" nxsd:terminatedBy=", "
nxsd:quotedBy="&quot;" />
    <element name="Address" type="string" nxsd:writeMode="after"
nxsd:write="Over." nxsd:style="terminated" nxsd:terminatedBy=", "
nxsd:quotedBy="&quot;" />
    <element name="DOB" type="string" nxsd:style="terminated"
nxsd:terminatedBy=", " nxsd:quotedBy="'" />
    <element name="Telephone" type="string" nxsd:style="terminated"
nxsd:terminatedBy="{eol}" nxsd:quotedBy="'" />
  </sequence>
</complexType>
</element>

</schema>
```

Translated Data Using Native Schema

Mr.Fred,"2 Old Street, Old Town,Manchester",Over.20-08-1954,0161-499-1718

7.2.2.13 Defining LookAhead

LookAhead are available in two types:

- Type 1: LookAhead X chars, read the value from a position using a style, and match against the specified literal.
- Type 2: LookAhead X chars, read the value from a position using a style, and store that value in a variable to be used later.

LookAhead: Type 1

In native schema, LookAhead X chars , read the value from a position using a style, and store that value in a variable to be used later.

Native Data Format to Be Translated

Name1,"2 Old Street, Old Town,Manchester",20-08-1954,"0161-499-1718", YES
Name2,"2 Old Street, Old Town,Manchester",20-08-1954,"0161-499-1718", NO
Name3,"2 Old Street, Old Town,Manchester",20-08-1954,"0161-499-1718", NO
Name4,"2 Old Street, Old Town,Manchester",20-08-1954,"0161-499-1718", YES

Native Schema

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://xmlns.oracle.com/pcbpel/nxsd/smoketest"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <!--
nxsd:lookAhead="70" nxsd:scan="3"
-->

  <element name="LookAhead">
```

```

    <complexType>
      <choice maxOccurs="unbounded" nxsd:choiceCondition="{x}" nxsd:lookAhead="70"
nxsd:scanLength="3" nxsd:assignTo="{x}">
        <element name="Record1" type="string" nxsd:conditionValue="YES"
nxsd:style="terminated" nxsd:terminatedBy=", " nxsd:skipMode="after"
nxsd:skipUntil="{eol}" />
        <element name="Record2" type="string" nxsd:conditionValue="NO "
nxsd:style="terminated" nxsd:terminatedBy=", " nxsd:skipMode="after"
nxsd:skipUntil="{eol}" />
      </choice>
    </complexType>
  </element>

</schema>

```

Translated XML Using Native Schema

```

<LookAhead xmlns="http://xmlns.oracle.com/pcbpel/nxsd/smoketest">
  <Record1>Name1</Record1>
  <Record2>Name2</Record2>
  <Record2>Name3</Record2>
  <Record1>Name4</Record1>
</LookAhead>

```

LookAhead: Type 2

LookAhead X chars, read the value from a position using a style, and match against the specified literal.

Native Data Format to Be Translated

Fred,"2 Old Street, Old Town,Manchester","20-08-1954","0161-499-1718",YES

Native Schema

```

<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://xmlns.oracle.com/pcbpel/nxsd/smoketest"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD">

  <element name="LookAhead">
    <complexType>
      <sequence minOccurs="0" nxsd:lookAhead="70" nxsd:lookFor="YES">
        <element name="PersonName" type="string" nxsd:style="terminated"
nxsd:terminatedBy=", " nxsd:quotedBy="&quot;" />
        <element name="Address" type="string" nxsd:style="terminated"
nxsd:terminatedBy=", " nxsd:quotedBy="&quot;" />
        <element name="DOB" type="string" nxsd:style="terminated"
nxsd:terminatedBy=", " nxsd:quotedBy="'" />
        <element name="Telephone" type="string" nxsd:style="terminated"
nxsd:terminatedBy=", " nxsd:quotedBy="'" />
      </sequence>
    </complexType>
  </element>

```

```
</schema>
```

Translated XML Using Native Schema

```
<LookAhead xmlns="http://xmlns.oracle.com/pcbpel/nxsd/smoketest">
  <PersonName>Fred</PersonName>
  <Address>2 Old Street, Old Town,Manchester</Address>
  <DOB>20-08-1954</DOB>
  <Telephone>0161-499-1718</Telephone>
</LookAhead>
```

7.2.2.14 Defining outboundHeader

The actual content of `outboundHeader` can use variables, specifically `${eol}`. When `headerLines` and `outboundHeader` both are available, `outboundHeader` takes precedence in the outbound.

Note: In the inbound direction, skipping headers feature is supported. Only predefined variables can be used in a header because other variables might either not be accessible or would have only literals.

Input XML

```
<terminated xmlns="http://xmlns.oracle.com/pcbpel/nxsd/smoketest">
  <PersonName>Fred</PersonName>
  <Address>2 Old Street, Old Town,Manchester</Address>
  <DOB>20-08-1954</DOB>
  <Telephone>0161-499-1718</Telephone>
</terminated>
```

Native Schema

```
<?xml version="1.0" encoding="US-ASCII"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  xmlns:nxsd="http://xmlns.oracle.com/pcbpel/nxsd"
  targetNamespace="http://xmlns.oracle.com/pcbpel/nxsd/smoketest"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified"
  nxsd:stream="chars"
  nxsd:version="NXSD"
  nxsd:hasHeader="true"
  nxsd:outboundHeader="This is a header ${eol}">

  <element name="terminated">
    <complexType>
      <sequence>
        <element name="PersonName" type="string" nxsd:style="terminated"
nxsd:terminatedBy=", " nxsd:quotedBy="&quot;" />
        <element name="Address" type="string" nxsd:style="terminated"
nxsd:terminatedBy=", " nxsd:quotedBy="&quot;" />
        <element name="DOB" type="string" nxsd:style="terminated"
nxsd:terminatedBy=", " nxsd:quotedBy="'" />
        <element name="Telephone" type="string" nxsd:style="terminated"
nxsd:terminatedBy="${eol}" nxsd:quotedBy="'" />
      </sequence>
    </complexType>
```

```
</element>
```

```
</schema>
```

Translated Data

This is a header

Fred,"2 Old Street, Old Town,Manchester",20-08-1954,0161-499-1718

7.3 Native Schema Constructs

Table 7–4 shows the constructs applicable only on the `<schema>` tag.

Table 7–4 Constructs Applicable Only on the `<schema>` Tag

Construct	Description
<code>byteOrder</code>	The byte order of the native data as <code>bigEndian</code> or <code>littleEndian</code> .
<code>encoding</code>	The encoding in which the actual data is stored. Any legal encoding supported by <code>java.io.InputStreamReader</code> .
<code>headerLines</code>	A positive integer specifying the number of lines to be skipped, before translating the native data.
<code>headerLinesTerminated</code> <code>By</code>	Skip until the specified string, before translating the native data.
<code>standalone</code>	If declared, adds the standalone attribute in the XML declaration prolog of the translated XML, with the actual value as that specified in <code>nxsd:standalone</code> . Allowed values are <code>true</code> and <code>false</code> .
<code>stream</code>	Whether the data is stored as characters or bytes. Allowed values are <code>CHARS</code> and <code>BYTES</code> .
<code>uniqueMessageSeparator</code>	String specifying the unique message separator in the native data, in case of a batch of messages.
<code>version</code>	The type of native data. Possible values are <code>NXSD</code> , <code>DTD</code> , <code>XSD</code> , and <code>OPAQUE</code> .
<code>xmlversion</code>	If declared, adds the XML declaration prolog to the translated XML with the actual value as that specified in <code>nxsd:xmlversion</code> . Allowed values are <code>1.0</code> and <code>1.1</code> .

Table 7–5 shows the constructs applicable on all tags other than the `<schema>` tag.

Table 7–5 Constructs Applicable On All Tags Other Than the `<schema>` Tag

Construct	Description
<code>arrayIdentifierLength</code>	The length of the array being stored in the native data occupying the specified length
<code>arrayLength</code>	<p>The value of this construct is used as the length of the array, which can also be a variable resolved to a valid number. This value overrides any <code>minOccurs</code> and <code>maxOccurs</code> attributes of the particle where it is specified. Use this feature as follows:</p> <pre>nxsd:style="array" nxsd:arrayLength="10"</pre> <p>This indicates that the array length is 10.</p>

Table 7–5 (Cont.) Constructs Applicable On All Tags Other Than the <schema> Tag

Construct	Description
arrayTerminatedBy	The last item in the array being terminated by the specified string
assign	Assigns a value to the variable already declared
cellSeparatedBy	The cells of the array in the native data being separated by the specified string
choiceCondition	Either <code>fixedLength</code> or <code>terminated</code>
conditionValue	Matches the string read from the native stream for the <code>choiceCondition</code> , against the specified string in the <code>conditionValue</code>
dateFormat	A valid Java date format representing the date in the native data
identifierLength	The number of characters and bytes in which the actual length of the data is stored
itemSeparatedBy	The items in the list being separated by the specified string
leftSurroundedBy, rightSurroundedBy	The native data surrounded
length	The length of the native data to be read. Used with fixed-length style.
listTerminatedBy	The last item in the list being terminated by the specified string
lookAhead	Looks for a match ahead of the current position in the input stream. If a match is found, then the node on which this construct is specified is processed; otherwise, it is skipped. Use this feature as follows: <code>nxsd:lookAhead="20" nxsd:lookFor="abc"</code> This indicates to skip 20 characters and look for the string <code>abc</code> starting from that location. If this is found, then the node is processed; otherwise, it is skipped.
paddedBy	The string used for padding
padStyle	<code>head</code> , <code>tail</code> , or <code>none</code>
quotedBy	The native data being quoted by the specified string
skip	Skips the specified number of bytes or characters
skipLines	Skips the number of lines specified
skipUntil	Skips until the string specified
startsWith	Looks for the specified string in the native data. If it exists, then proceeds with the element where it is specified; otherwise, skips and processes the next element.
style	The style used to read the native data from the input stream. Allowed values are <code>fixedLength</code> , <code>surrounded</code> , <code>terminated</code> , <code>list</code> , and <code>array</code> .
surroundedBy	The native data being surrounded by the specified string
terminatedBy	The native data being terminated by the string specified
variable	Declares a single variable
variables	Declares a set of variables or assigns the already declared variables a valid value

Troubleshooting and Workarounds

This appendix describes Oracle BPEL Process Manager and Oracle Enterprise Service Bus troubleshooting methods.

This appendix contains the following topics:

- [Section A.1, "Troubleshooting the Oracle Application Server Adapter for Databases"](#)
- [Section A.2, "Troubleshooting the Oracle Application Server Adapter for Databases When Using Stored Procedures"](#)
- [Section A.3, "Troubleshooting the Oracle Application Server Adapter for Files/FTP"](#)
- [Section A.4, "Troubleshooting the Oracle Application Server Adapter for Advanced Queuing"](#)
- [Section A.5, "Troubleshooting the Oracle Application Server Adapter for Java Message Service \(JMS\)"](#)

A.1 Troubleshooting the Oracle Application Server Adapter for Databases

The following sections describe possible issues and solutions when using the Oracle Application Server Adapter for Databases (database adapter).

This section comprises the following issues:

- [Could Not Create TopLink Session Exception](#)
- [Could Not Find Adapter for eis/DB/my_connection](#)
- [Changes Through TopLink Workbench](#)
- [Redeploying from the Command Line](#)
- [Cannot Change Customers_table.xsd](#)
- [No Target Foreign Keys Error](#)
- [No Primary Key Exception](#)
- [dateTime Conversion Exceptions](#)
- [Issues with Oracle DATE](#)
- [TIMESTAMP Datatype Is Not Supported for a Microsoft SQL Server Database](#)
- [Handling a Database Adapter Fault](#)
- [Table Not Found: SQL Exception](#)

- Switching from a Development Database to a Production Database
- Only One Employee Per Department Appears
- Outbound SELECT on a CHAR(X) or NCHAR Column Returns No Rows
- ORA-00932: Inconsistent Datatypes Exception Querying CLOBs
- ORA-17157: 4K/32K Driver Limit with CLOBs and BLOBs
- MERGE Sometimes Does UPDATE Instead of INSERT, or Vice Versa
- Message Loss with the MERGE Invoke Operation
- Integrity Violation Occurs with Delete or DeletePollingStrategy
- Some Queried Rows Appear Twice or Not at All in the Query Result
- Importing a Same-Named Table, with Same Schema Name, but Different Databases
- Problems Creating a Relationship Manually for a Composite Primary Key
- Must Fully Specify Relationships Involving Composite Primary Keys
- Database Adapter Throws an Exception When Using a BFILE
- During Design-Time, Wizard Does Not Allow Deletion of a Table
- Changes to JDeveloper Project Are Made Even If Wizard Is Cancelled
- Problems Removing a Relationship, Then Adding a New Relationship with the Same Name
- Problems Importing Third-Party Database Tables
- Problems Importing Object Tables
- Relationships Not Autogenerated When Tables Are Imported Separately
- Primary Key Is Not Saved
- Table Column Name Is a Java Keyword
- Catching a Database Exception
- Connection Settings Error: Too Many Transactions
- ORA-01747: invalid user.table.column, table.column, or column specification, When Using SELECT FOR UPDATE
- Update Only Sometimes Performs Inserts/Deletes for Child Records

A.1.1 Could Not Create TopLink Session Exception

Problem

At run time, you may see the "Could not create the TopLink session" exception.

Solution

This common error occurs when the run-time connection is not configured properly. See "[Deployment](#)" for more information.

A.1.2 Could Not Find Adapter for eis/DB/my_connection

Problem

You may see the "Could not find adapter for eis/DB/my_connection/...." exception.

Solution

See ["Deployment"](#) for more information.

A.1.3 Changes Through TopLink Workbench

Changes through TopLink Workbench require you to run the Adapter Configuration Wizard again in edit mode to force a refresh of the `toplink_mappings.xml` file.

A.1.4 Redeploying from the Command Line

If you redeploy using `obant`, unless the `bpel.xml` or `.bpel` files have changed, the redeployment is skipped by design.

A.1.5 Cannot Change Customers_table.xsd

Problem

Changes to `Customers_table.xsd` are not reflected, or you get an exception.

Solution

You cannot specify the XSD format that the database adapter produces. See ["XML Schema Definition \(XSD\)"](#) for details.

A.1.6 No Target Foreign Keys Error

Problem

After clicking **Finish**, or at deployment, you may see the following exception:

```
Caused by Exception [TOPLINK-0] (OracleAS TopLink - 10g (9.0.4.4) (Build 040705)):
oracle.toplink.exceptions.IntegrityException
```

```
Descriptor Exceptions:
```

```
-----
```

```
Exception [TOPLINK-64] (OracleAS TopLink - 10g (9.0.4.4) (Build 040705)):
oracle.toplink.exceptions.DescriptorException
Exception Description: No target foreign keys have been specified for this
mapping.
Mapping: oracle.toplink.mappings.OneToManyMapping[phonesCollection]
Descriptor: Descriptor[Test.Customers --> [DatabaseTable(CUSTOMERS)]]
```

This generally means that there was a problem in the wizard.

Solution

The simplest solution is to create all constraints on the database first. Also, depending on the problem, you may only need to fix something in the offline tables, and then run the wizard again.

If you want to create a one-to-many mapping from **CUSTOMERS** to **PHONES**, you need a foreign key constraint on **PHONES**.

This procedure assumes that this constraint does not exist on the database, and that you tried to create it with the wizard and it generated an exception.

- In your JDeveloper BPEL Designer project, click the plus sign (+) in the **Applications Navigator** to add files to your project. Select **database > schemaName > schemaName.schema**. This imports all your database objects.
- Open the **PHONES** table and manually create the foreign key constraint from **PHONES** to **CUSTOMERS**.
- Save all.
- Now open the TopLink project. In JDeveloper BPEL Designer, go to **Application Sources > TopLink > TopLink Mappings**. In the **Structure** window, open **CUSTOMERS** and double-click the **phonesCollection** mapping.

You should now see a **Table Reference** tab in the main window. This was probably blank previously. From the menu, select the one you just created.

- Save again.
- Edit the database partner link.

Click **Next** to the end in the wizard, and then click **Finish** and **Close**.

This refreshes `toplink_mappings.xml` for your project.

- Open the `toplink_mappings.xml` file in JDeveloper BPEL Designer. You may need to add it to the project first.
- Search for `phonesCollection`.
 - You should find a tag like this:

```
<database-mapping>
<attribute-name>phonesCollection</attribute-name>
```

- Scroll down and you should now see something like this:

```
<source-key-fields>
<field>CUSTOMERS.someColumn</field>
</source-key-fields>
<target-foreign-key-fields>
<field>PHONES.someColumn</field>
</target-foreign-key-fields>
```

- If you do not see the tags shown here, then manually add them. This is not the preferred method because `toplink_mappings.xml` is a generated file, which gets refreshed whenever you edit a database partner link.
- Undeploy the old process in Oracle BPEL Control, and redeploy your fixed process with a new revision number.

A.1.7 No Primary Key Exception

Problem

After clicking **Finish**, or at deployment, you may see the following exception:

```
Caused by Exception [TOPLINK-0] (OracleAS TopLink - 10g (9.0.4.4) (Build 040705)):
oracle.toplink.exceptions.IntegrityException
```

Descriptor Exceptions:

```
Exception [TOPLINK-46] (OracleAS TopLink - 10g (9.0.4.4) (Build 040705)):
oracle.toplink.exceptions.DescriptorException
Exception Description: There should be one non-read-only mapping defined for the
primary key field [PHONES.ID].
Descriptor: Descriptor(Test.Phones --> [DatabaseTable(PHONES)])
```

This probably means that no primary key was defined for PHONES.

Solution

If this exception appears in conjunction with the **No Target Foreign Keys** error, then see ["No Target Foreign Keys Error"](#) and resolve that problem first. Otherwise, do the following:

- **Open Application Sources > TopLink > TopLink Mappings.**

- In the **Structure** window, double-click **PHONES**.

On the first page, you should see **Primary Keys**. Ensure that columns are selected and that they are mapped in the project.

- Save.

- **Edit the database partner link.**

Click **Next** to the end in the wizard, and then click **Finish** and **Close**.

- **Open toplink_mappings.xml.** For the PHONES descriptor, you should see something like this:

```
<primary-key-fields>
<field>PHONES.ID</field>
</primary-key-fields>
```

- Make sure that there is at least one field, and for that field, make sure you can find it somewhere else in the `toplink_mappings.xml` file. If you can, then this means that the database adapter can detect it. If not, then you must map this field.

- **Open Application Sources > Test > Phones.**

You should see Java code. Add a line as follows:

```
long id;
```

- **Save.**

- **Now open Application Sources > TopLink > TopLink Mappings.**

Double-click **PHONES** and go to the Structure window.

You should be able to map the id as **DirectToField**, and set the database field to **ID**.

- **Save and then edit the database partner link.**

Click **Next** to the end in the wizard, and then click **Finish** and **Close**.

- **Undeploy the old process in Oracle BPEL Control, and redeploy your fixed process with a new revision number.**

A.1.8 dateTime Conversion Exceptions

Problem

You may get a conversion exception when you pass in an `xs:dateTime` value to the database adapter.

Solution

If an attribute is of type `xs:dateTime`, then the database adapter is expecting a string in one of the following formats:

```
1999-12-25T07:05:23-8:00
1999-12-25T07:05:23.000-8:00
1999-12-25T15:05:23:000Z
1999-12-25T15:05:23
```

The format `1999-12-25` is accepted, although it is not a valid `xs:dateTime` value. The `xs:dateTime` format is `yyyy-MM-ddTHH:mm:ss.SSSZ`, where

- `yyyy` is the year (2005, for example)
- `MM` is the month (01 through 12)
- `dd` is the day (01 through 31)
- `HH` is the hour (00 through 23)
- `mm` is the minute (00 through 59)
- `ss` is the second (00 through 59)
- `SSS` is milliseconds (000 through 999), optional
- `Z` is the time zone designator (`+hh:mm` or `-hh:mm`), optional

A `DATE` column may exist on an Oracle Database, which can accept the `25-DEC-1999` date format. However, this is not a date format that the database adapter can accept. The following workaround applies to TopLink only.

- If you want to pass in the `25-DEC-1999` date format, then map the attribute as a plain string. The database adapter passes the value through as-is.
 - To do this, you must edit the offline database table and change the column datatype from `DATE` to `VARCHAR2`.
- Save.
- Edit the database partner link.

Click **Next** to the end in the wizard, and then click **Finish** and **Close**.

While not a valid `xs:dateTime` format, the format `yyyy-mm-dd` is a valid `xs:date` format.

A.1.9 Issues with Oracle DATE

Problem

The *time* portion of `DATE` fields may be truncated on Oracle9 or greater platforms when using `oracle.toplink.internal.databaseaccess.DatabasePlatform`. For example, `2005-04-28 16:21:56` becomes `2005-04-28T00:00:00.000+08:00`.

Or, the millisecond portion of DATE fields may be truncated on Oracle9 or greater platforms when using `oracle.toplink.internal.databaseaccess.Oracle9Platform`. For example, `2005-04-28 16:21:56.789` becomes `2005-04-28T16:21:56.000+08:00`.

Or, you may have trouble with `TIMESTAMP` (time stamp with time zone) or `TIMESTAMPLTZ` (time stamp with local time zone).

Solution

You must set the `platformClassName` for Oracle platforms, because these include special workarounds for working with date-time values on Oracle. So, if you are connecting to an Oracle9 platform, you must set the `platformClassName` accordingly.

Due to an issue with the time portion of DATE being truncated with Oracle9 JDBC drivers, the property `oracle.jdbc.V8Compatible` was set when using any Oracle platform class name. Therefore, use `oracle.toplink.internal.databaseaccess.Oracle9Platform` to solve the time truncation problem.

However, starting with Oracle9, dates started to include millisecond precision. Setting `oracle.jdbc.V8Compatible` in response had the drawback of returning the milliseconds as 000, as an Oracle8 database did. (This also introduced an issue with null IN/OUT DATE parameters for stored procedure support.) You do not see any truncation (of the time portion or milliseconds) when using the `Oracle9Platform` class.

You must also use the `Oracle9Platform` class if you have `TIMESTAMP` and `TIMESTAMPLTZ`.

If you want DATE to be treated like a date (with no time portion), set the attribute-classification in the `toplink_mappings.xml` to `java.sql.Date`.

In general, if you are having an issue with a particular database, check to see if TopLink has a custom `platformClassName` value for that database, and whether you are using it.

See ["Deployment"](#) and for more information.

A.1.10 TIMESTAMP Datatype Is Not Supported for a Microsoft SQL Server Database

Because the `TIMESTAMP` datatype is not supported, the best approach is to unmap a `TIMESTAMP` column. Note the following in support of unmapping `TIMESTAMP`:

- `TIMESTAMP` values can never be used as the primary key.
- JDeveloper offline tables interprets `TIMESTAMP` as a `dateTime` type, although it is actually a binary value; therefore, you must change the type anyway.
- As a binary value, `TIMESTAMP` has no meaning or use after it is converted to XML and base64 encoded.
- `TIMESTAMP` values cannot be modified; therefore, at a minimum, you must mark it read only.

Note that `TIMESTAMP` is similar to the pseudocolumn `ROWID`, which is technically a column but is never mapped by default by the database adapter.

See ["The TopLink Workbench Project"](#) and go to the steps for unmapping a mapping.

A.1.11 Handling a Database Adapter Fault

To understand how to handle faults, such as a unique constraint violation on inserts or when a database or network is temporarily unavailable, see the `InsertWithCatch` tutorial at `Oracle_Home\bpel\samples\tutorials\122.DBAdapter`.

A.1.12 Table Not Found: SQL Exception

Problem

A BPEL process modeled against one database does not run against another database.

The most likely cause for this problem is that you are using a different schema in the second database. For example, if you run the wizard and import the table `SCOTT.EMPLOYEE`, then, in the `toplink_mappings.xml` file, you see `SCOTT.EMPLOYEE`. If you run the sample in the `USER` schema on another database, you get a "table not found" exception.

Solution

Until qualifying all table names with the schema name is made optional, manually edit `toplink_mappings.xml` and replace `SCOTT.` with nothing, as shown in the following example.

Change:

```
<project>
  <project-name>toplink_mappings</project-name>
  <descriptors>
    <descriptor>
      <java-class>BPELProcess1.A</java-class>
      <tables>
        <table>SCOTT.A</table>
      </tables>
    </descriptor>
  </descriptors>
</project>
```

To:

```
<project>
  <project-name>toplink_mappings</project-name>
  <descriptors>
    <descriptor>
      <java-class>BPELProcess1.A</java-class>
      <tables>
        <table>A</table>
      </tables>
    </descriptor>
  </descriptors>
</project>
```

You must repeat this step every time after running the Adapter Configuration Wizard.

Note: Having `EMPLOYEE` on both the `SCOTT` and `USER` schemas, and querying against the wrong table, can result in a problem that is difficult to detect. For this reason, the database adapter qualifies the table name with the schema name.

See "[Deployment](#)" for more information.

A.1.13 Switching from a Development Database to a Production Database

See the following for more information:

- "Deployment"
- "Table Not Found: SQL Exception"

A.1.14 Only One Employee Per Department Appears

Problem

Many departments with many employees are read in, but only one employee per department appears.

Solution

You must use a transform with a `for-each` statement. An **Assign** activity with a too-simplistic XPath query can result in only the first employee being copied over.

For an example of how to use a transform for database adapter outputs, go to

`Oracle_Home\bpel\samples\tutorials\122.DBAdapter\MasterDetail`

A.1.15 Outbound SELECT on a CHAR(X) or NCHAR Column Returns No Rows

Problem

If you use an outbound `SELECT` to find all employees where `firstName = some_parameter`, then you have a problem if `firstName` on the database is a `CHAR` column, as opposed to a `VARCHAR2` column.

It is a known problem with some databases that if you insert a `CHAR` value (for example, 'Jane') into a `CHAR(8)` field, then the database pads the value with extra spaces (for example, 'Jane ').

If you then execute the query

```
SELECT ... WHERE firstName = 'Jane';
```

no rows may be returned. Although you are querying for the same value that you inserted, and some tools such as SQL*Plus and SQL Worksheet operate as expected, the query does not work with the database adapter.

Solution

The best practice is to use a `CHAR` column for fields that must be fixed, such as `SSN`, and `VARCHAR2` for columns that can take a variable length, such as `firstName`.

Transforming the value to add padding may be difficult, and using `SELECT` to trim the value on the database (as opposed to padding the other side) requires using SQL statements. For example:

```
SELECT ... WHERE trim(firstName) = #firstName;
```

Note that `#` is an TopLink convention for denoting input parameters.

A.1.16 ORA-00932: Inconsistent Datatypes Exception Querying CLOBs

Problem

When querying on table A, which has a one-to-one relationship to B, where B contains a CLOB, you may see the following exception:

```
Exception Description: java.sql.SQLException: ORA-00932: inconsistent
datatypes: expected - got CLOB
```

Solution

A SELECT returning CLOB values must not use the DISTINCT clause. The simplest way to avoid DISTINCT is to disable batch attribute reading from A to B. Batch reading is a performance enhancement that attempts to simultaneously read all Bs of all previously queried As. This query uses a DISTINCT clause. Use joined reading instead, or neither joined reading nor batch attribute reading.

Because both DISTINCT and CLOBs are common, you may see this problem in other scenarios. For example, an expression like the following uses a DISTINCT clause:

```
SELECT DISTINCT dept.* from Department dept, Employee emp WHERE ((dept.ID =
emp.DEPTNO) and (emp.name = 'Bob Smith'));
```

See "[Relational-to-XML Mappings \(toplink_mappings.xml\)](#)" for more information about batch reading and joined reading.

A.1.17 ORA-17157: 4K/32K Driver Limit with CLOBs and BLOBs

Problem

When inserting large objects (LOBs), you may get an exception such as

```
java.sql.SQLException: setString can only process strings of less than 32766
characters Error Code: 17157
```

Solution

Check the platformClassName property in the oc4j-ra.xml file. For an Oracle database, set the property to Oracle8Platform (for Oracle8) or Oracle9Platform (for Oracle9i and Oracle10g). See [Table 4-17, "Database Platform Names"](#) for a list of platformClassName properties for Oracle and third-party databases.

For more information, see "How-To: Map Large Objects (LOBs) to Oracle Databases with OracleAS TopLink" at

<http://www.oracle.com/technology/products/ias/toplink/technical/tips/lob/index.html>

If you are using Oracle Database 10g and having difficulties with CLOBs, configure the database adapter to use a data source, and add

```
<propertyname="SetBigStringTryClob" value="true" /> to the
<data-source> element in the OC4J data-sources.xml file.
```

Also, see "Handling CLOBs - Made Easy with Oracle JDBC 10g" at

http://www.oracle.com/technology/sample_code/tech/java/codesnippet/jdbc/clob10g/handlingclobsinoraclejdbc10g.html

A.1.18 MERGE Sometimes Does UPDATE Instead of INSERT, or Vice Versa

Problem

You may sometimes notice that MERGE performs an UPDATE when it should do an INSERT, or vice versa.

Solution

MERGE works by first determining, for each element in the XML, whether the corresponding database row exists or not. For each row, it does an existence check. There are two known limitations with the existence check.

First, you can configure the existence check to either **Check cache** or **Check database**. You can configure this for each descriptor (mapped table) in your Mapping Workbench Project. The default is **Check database**, but TopLink's check database works like "check cache first, then database" for performance reasons. If a row exists in the cache, but was deleted from the database (the cache is stale), then you may see an UPDATE when you expect an INSERT. You can configure caching and a `WeakIdentityMap` is used by default, meaning rows are only held in memory while being processed. However, Java garbage collection is not controlled by the adapter. Therefore, if you insert a row, delete it in a separate process, and insert it again, all within a very short time, you may see an INSERT and then an UPDATE. One solution is to use `NoIdentityMap`. However, performance may suffer, and if you are doing SELECT statements on a mapped schema with complex cycles (which you should avoid!), then the adapter can be trapped in an endless loop when building the XML.

Second, there is a timing issue to doing a read first and then later an INSERT or UPDATE. If the same row is simultaneously inserted by multiple invokes, then each may do an existence check that returns false, and then all attempt an INSERT. This does not seem realistic, but the following scenario did come up:

A polling receive reads 100 employee rows and their departments from database A. With `maxRaiseSize` set to 1, 100 business process instances were initiated. This led to 100 simultaneous invokes to database B, one for each employee row. No problems were encountered when existence checking on employee, but some employees had the same department. Hence, many of the 100 invokes failed because the existence checks on department were more or less simultaneous.

There are two solutions to this problem. The first is to avoid it. In a data synchronization-style application, setting `maxRaiseSize` to `unlimited` boosts performance and eliminates this problem. A second solution is to retry MERGE in your BPEL process. Optimistic lock and concurrency exceptions are common, and the best solution is usually to wait and try again a short time later.

A.1.19 Message Loss with the MERGE Invoke Operation

Problem

Using the MERGE invoke operation, 100 rows are passed to the database adapter. However, not all the rows are inserted into the database table as expected.

See ["MERGE Sometimes Does UPDATE Instead of INSERT, or Vice Versa"](#) for more information about this problem. A flaw in MERGE existence checking allows cases where a row is not inserted when it should be, thus appearing as message loss.

Solution

Use `NoIdentityMap` instead of `WeakIdentityMap`.

1. In JDeveloper BPEL Designer, open your project.
2. In the **Applications - Navigator**, double-click the **TopLink Mappings** node under **Application Sources** for your project.

The TopLink project appears in the **TopLink Mappings - Structure** pane.

3. Click each descriptor in the **TopLink Mappings - Structure** pane so that it appears in the main window.

Along the top of the main window, you see the following tabs: **Descriptor Info**, **Queries**, **Query Keys**, and **Identity**.

4. Click the **Identity** tab.

5. From the Identity Map list, select **NoIdentityMap**.
6. Set **Existence Checking** to **Check database** (the default).
The value **Check Cache** becomes illegal when no caching is used.
7. From **File**, select **Save All**.
8. Run the Adapter Configuration Wizard again in edit mode to regenerate `toplink_mappings.xml`.
9. (Optional) Verify that the solution worked by closing and then reopening `toplink_mappings.xml`.
You will see that `NoIdentityMap` globally replaced `WeakIdentityMap`.
10. Redeploy the process.

See "[The TopLink Workbench Project](#)" for more information on editing the underlying TopLink project.

A.1.20 Integrity Violation Occurs with Delete or DeletePollingStrategy

Problem

Child records found an integrity violation with `DeletePollingStrategy`.

When deleting rows, you must be aware of integrity constraints. For example, if `DEPARTMENT` has a one-to-many relationship to `EMPLOYEE`, that means `DEPTID` is a foreign key on `EMPLOYEE`. If you delete a `DEPARTMENT` record, but not its employees, then `DEPTID` becomes a broken link and this can trigger an integrity constraint.

This problem occurs because you imported a table by itself and did not import its related tables. For example, if you import only the `DEPARTMENT` table from the database and not the `EMPLOYEE` table, which has an integrity constraint on column `DEPTID`, then the database adapter does not know about `EMPLOYEE` and it cannot delete a record from `DEPARTMENT`. You receive an exception.

Solution

Make sure you import the master table and all its privately-owned relationships. Or set the constraint on the database to `CASCADE` for deletions, or use a nondelete polling strategy.

Make sure that the one-to-many relationship between `DEPARTMENT` and `EMPLOYEE` is configured to be privately owned. It is by default, but if the above fails, check the run-time X-R mappings file. See "[Relational-to-XML Mapping](#)" for more information.

If the problem is not this simple, TopLink supports shallow/two-phase inserts (but does not support this for `DELETE`). For example, if A has a foreign key pointing to B, and B has a foreign key pointing to A, then there is no satisfactory order by which you can delete both A and B. If you delete A first, then you orphan B. If you delete B first, then you orphan A. The safest `DELETE` is a two-phase `DELETE` that performs an `UPDATE` first as follows:

```
UPDATE B set A_FK = null;
DELETE from A;
DELETE from B;
```

A.1.21 Some Queried Rows Appear Twice or Not at All in the Query Result

Problem

When you execute a query, you may get the correct number of rows, but some rows appear multiple times and others do not appear at all.

This behavior is typically because the primary key is configured incorrectly. If the database adapter reads two different rows that it thinks are the same (for example, the same primary key), then it writes both rows into the same instance and the first row's values are overwritten by the second row's values.

Solution

- Open **Application Sources > TopLink > TopLink Mappings**. In the Structure window, double-click **PHONES**. On the first page, you should see **Primary Keys**. Make sure that the correct columns are selected to make a unique constraint.
- Save and then edit the database partner link.
Click **Next** to the end, and then click **Finish** and **Close**.
- Open your `toplink_mappings.xml` file. For the `PHONES` descriptor, you should see something like this:

```
<primary-key-fields>
<field>PHONES.ID1</field>

<field>PHONES.ID2</field>
</primary-key-fields>
```

A.1.22 Importing a Same-Named Table, with Same Schema Name, but Different Databases

Problem

Importing a table from a database on one host and also importing a table with the same name, and the same schema name, from a database on another host raises an error.

Solution

Create one project against database #1 and model the adapter service. Next create a second project against database #2 and model the adapter service. (Because the databases are on different hosts, you use different database connections.) Then create a third project, but do not run the Adapter Configuration Wizard. Instead, copy the BPEL artifacts (WDSL, XSD, and `toplink_mappings.xml`) from projects one and two. Deploy only the third project.

If the two tables are identical, or if the data you are interested in is identical, then you need not follow the preceding procedure.

A.1.23 Problems Creating a Relationship Manually for a Composite Primary Key

Problem

In the Relationship window of the Adapter Configuration Wizard, all elements of the primary key appear and cannot be removed. Therefore, a foreign key referring to only part of the composite primary key cannot be created.

Solution

Because foreign key constraints must map to every part of the primary key (not a subset), there is no solution. The database adapter allows a foreign key only with a corresponding primary key at the other end.

A.1.24 Must Fully Specify Relationships Involving Composite Primary Keys

The wizard does not let you create an ambiguous relationship. For example, assume that a `PurchaseOrder` has a 1-1 `billTo` relationship to a `Contact`. For uniqueness, the primary key of `Contact` is name and province. This means `PurchaseOrder` must have two foreign keys (`bill_to_name` and `bill_to_province`). If there is only one foreign key (`bill_to_name`), then the wizard does not allow you to create that ambiguous 1-1 relationship. Otherwise, the same purchase order can be billed to multiple people.

A.1.25 Database Adapter Throws an Exception When Using a BFILE

The `BFILE`, `USER_DEFINED`, `OBJECT`, `STRUCT`, `VARRAY`, and `REF` types are not supported.

A.1.26 During Design-Time, Wizard Does Not Allow Deletion of a Table

Because a table may be used for other services inside the same project, it cannot be deleted within the interface. No problems occur if the unneeded table remains as part of your project.

A.1.27 Changes to JDeveloper Project Are Made Even If Wizard Is Cancelled

Problem

When you run the Adapter Configuration Wizard, any changes that affect the TopLink project (importing tables, creating or removing mappings, specifying an expression, and so on) are applied immediately, and are *not undone* if you cancel the wizard.

Solution

If you remove one or more of the autogenerated relationships and later want to get them back, you must reimport the tables containing the corresponding database constraints.

A.1.28 Problems Removing a Relationship, Then Adding a New Relationship with the Same Name

Problem

The database adapter can become unstable if, within the same wizard session, you remove a relationship and then immediately create a relationship with the same name.

Solution

You can do one of the following:

- Give the new relationship a different name from the one you removed.
- Finish the wizard after you remove the first relationship. Then, start the wizard again in edit mode to add the new relationship, using the same name as the deleted relationship.

A.1.29 Problems Importing Third-Party Database Tables

Problem

When you import tables from some third-party databases, JDeveloper can encounter problems handling certain datatypes. You may see error messages such as "Columns of type VARCHAR cannot have a size specified" or "A primary or unique key must define at least one column." An error message such as "null" may also indicate an unsupported datatype.

Solution

Use the following workaround:

1. Click **OK** to dismiss the error message; then cancel the Adapter Configuration Wizard.
2. Edit the offline table definition to change the type of the columns mentioned in the error message to the closest supported type.

The offline table definitions for your project are under the **Database Objects > schema name** node in the **Applications Navigator** of JDeveloper BPEL Designer.

The following summarizes how to find the closest offline database table datatype:

Third-Party Database Datatype	Offline Database Table Datatype	XML Datatype
VARCHAR, CLOB, anything string-like	VARCHAR2	xs:string
RAW, BYTE, BLOB	BLOB	xs:base64Binary
Any imprecise number	FLOAT	xs:double
Anything numeric	NUMERIC	xs:decimal
Any time value	DATE	xs:dateTime

The preceding table summarizes [Table 4-4, "Mapping Database Data Types to XML Primitive Types"](#). Use [Table 4-4](#) to find the closest supported type for your third-party database. For `xs:string`, for example, use *any* of the types listed in the Database Type column. When you change a datatype, only the corresponding XML type is relevant. The `varchar2`, `varchar`, `clob`, and `nvarchar` types all map to `xs:string`.

3. Run the Adapter Configuration Wizard again and continue with the rest of the wizard.

If you do not see the **Database Objects > schema name** node in the **Applications Navigator** after importing your tables, then add it manually by clicking the **Add to project name.jpr** button in the **Applications Navigator**. Then select the **database > schemaName > schemaName.schema** file.

See [Configuring Offline Database Tables](#) for more information.

See [Design Time: Using the Command-Line Utility](#) for information about using the correct third-party JDBC drivers.

A.1.30 Problems Importing Object Tables

Problem

JDeveloper does not currently support importing object tables. If you try to import an object table, then you see the following message: "The following tables were 'Object Tables' and aren't supported offline."

Solution

There is currently no workaround for this problem.

A.1.31 Relationships Not Autogenerated When Tables Are Imported Separately

Problem

If tables are imported one at a time, relationships are not generated even if foreign key constraints exist on the database.

Solution

Relationship mappings can be autogenerated only if all the related tables are imported in one batch. When importing tables, you can select multiple tables to be imported as a group. If you have related tables, then they should all be imported at the same time.

A.1.32 Primary Key Is Not Saved

Problem

If you try to create a relationship that has the same name as the primary key field name, then you encounter a problem in which the PK field becomes unmapped.

Solution

To add the PK mapping back manually, follow these instructions:

1. Open the Java source for the descriptor to which you want to add the mapping (for example, `Movies.java`).
2. Add a new Java attribute appropriate for the field to which you are mapping. For example, if the PK of the `Movies` table is a `VARCHAR` field named `TITLE`, then create a new attribute: `private String title;`
3. Save the Java file.
4. Click the **TopLink Mappings** node in the **Applications - Navigator** pane; then choose the **Descriptor** from the **TopLink Mappings - Structure** pane. You see the newly created attribute in the **Descriptor** as unmapped (in this example, **title**).
5. Right-click the new attribute and select **Map As > Direct To Field**.
6. Double-click the new attribute. The TopLink Mappings editor should appear in the main JDeveloper window. Change the database field to match the PK field on the database (in this example, **TITLE**).
7. Click the **Descriptor** in the **TopLink Mappings - Structure** pane. Ensure that the PK field has a check box next to it in the **Primary Keys** list.
8. Run the Adapter Configuration Wizard again and continue with the rest of the wizard.

A.1.33 Table Column Name Is a Java Keyword

Problem

If you import a database table that contains a column whose name is a Java keyword, you receive the following error message:

The following error occurred: null

Solution

Do the following in JDeveloper BPEL Designer:

1. Click **OK** in the error dialog box.
2. Click **Cancel** in the Adapter Configuration Wizard.
3. Click **Cancel** in the Create Partner Link dialog box.
4. Open the `.java` file that was generated during the failed import. In the **Application Navigator**, click **Applications**, then *WorkspaceName*, then *ProcessName*, then **Application Sources**, then *ProcessName*, and then *TableName.java*.

5. Rename any Java fields that have errors. For example, you may see

```
private String class;
```

If you have syntax error highlighting turned on, this line will be underlined in red, indicating there is a Java error. Change the line to

```
private String myClass;
```

(Or use some other nonreserved word.)

6. Delete all the methods from the Java class. This step is normally handled automatically by the database adapter, but must be done manually here because of the error encountered during import. After you delete the methods, your class looks something like this:

```
package MyBPELProcess;
public class MyDatabaseTable {
    private String fieldOne;
    private String fieldTwo;
    ...
    private String fieldN;
}
```

7. Remap the field that you renamed in Step 5. Click the **Mapping** tab at the bottom of the Java Class editor. The **Structure** pane updates to show the Java class attributes. Right-click the field that you renamed in step 5 (unlike the other fields, it has a single dot icon indicating it is unmapped) and select **Map As -> Direct To Field**. In the main editor window, select the database field that this Java field maps to (the field has the same name as the attribute did before you renamed it). Then close the Java Class editor.
8. From **File**, select **Save All**.
9. Rerun the Adapter Configuration Wizard. When you get to the Select Table page, your database table will already be in the list. Select it and continue with the wizard. Do not import the table again.

A.1.34 Catching a Database Exception

Problem

Extra steps are needed to add fault handling to a BPEL process.

Solution

The steps for catching a database exception are provided in the 122.DBAdapter tutorial. Go to

`Oracle_Home\bpel\samples\tutorials\122.DBAdapter\InsertWithCatch`

See the `readme.txt` files in *both* the `Insert` and `InsertWithCatch` directories.

The `readme` for the `InsertWithCatch` tutorial describes two kinds of faults—binding faults (for example, inserting a row that already exists) and remote faults (for example, inserting a row when the network or database is unavailable). The `readme` provides steps for catching an exception and a list of common Oracle database error codes.

See *Oracle Database Error Messages* for a complete list of error codes.

A.1.35 Connection Settings Error: Too Many Transactions

Problem

When using Oracle Lite, you may see the following error:

```
java.sql.SQLException: [POL-3261] there are too many transactions
```

This means that the maximum number of connections from the database has been exceeded.

Oracle BPEL Server uses a data source called `BPELServerDataSource`, which is configured with a large connection pool size. The connections may all be allocated to the BPEL engine, thus leaving no connections available for the database adapter.

Solutions

Try the following solutions, in order of preference.

Solution 1

Use an Oracle database instead of Oracle Lite. This error occurs with Oracle Lite only.

Solution 2

Try reducing the values for `max-connections` and, in particular, for `min-connections`. You may need to experiment to find values that work in your environment. Start with a value of 5 for both `max-connections` and `min-connections` and see if your performance is acceptable. You must use higher values for a production database.

To set the values for `max-connections` and `min-connections`:

1. Open `data-sources.xml`, located in

`Oracle_Home\bpel\appserver\oc4j\j2ee\home\config`

2. In the Oracle Lite `datasources` section, set `max-connections` and `min-connections`:

```
<!-- Use these datasources to connect to Oracle Lite -->
<data-source class="com.evermind.sql.DriverManagerDataSource"
```

```

name="BPELServerDataSource"
location="loc/BPELServerDataSource"
xa-location="BPELServerDataSource"
ejb-location="jdbc/BPELServerDataSource"
connection-driver="oracle.lite.poljdbc.POLJDBCdriver"
username="system"
password="any"
max-connections="5"
min-connections="5"
connection-retry-interval="30"
max-connect-attempts="10"
url="jdbc:polite4@127.0.0.1:100:orabpel"/>

```

Solution 3

Reduce the number of applications that access Oracle Lite. Multiple concurrent BPEL processes are one application, but can use all the connections.

A.1.36 ORA-01747: invalid user.table.column, table.column, or column specification, When Using SELECT FOR UPDATE

Ensure that the `oc4j-ra.xml` file uses the correct value for `platformClassName`. See [Table 4-17, "Database Platform Names"](#) for the values for various databases. If the database you are using is listed in the table, use the value shown. For example, use `DB2Platform`, not `DatabasePlatform`, if you are using a DB2 database.

A.1.37 Update Only Sometimes Performs Inserts/Deletes for Child Records

The `Update Only` operation in the wizard sometimes performs inserts/deletes of the child records.

The `Insert Only` and `Update Only` are different in terms of recursively inserting/updating child records. The `Insert Only` will insert the top-level table and all related child records. It assumes the top level record and all related records are new. The `Update Only` is guaranteed to do an update only of the top-level records. It then checks whether to do an insert/update, or delete of the child records. It makes an assumption only about the existence of the top-level element. The `Merge` operation makes no assumptions about the existence of either top-level or child records.

A.2 Troubleshooting the Oracle Application Server Adapter for Databases When Using Stored Procedures

The following sections describe possible issues and solutions when using the database adapter for stored procedures:

- [Design Time: Unsupported or Undefined Parameter Types](#)
- [Design Time: Referencing User-Defined Types in Other Schemas](#)
- [Run Time: Parameter Mismatches](#)
- [Run Time: Stored Procedure Not Defined in the Database](#)
- [Configuring Multiple Adapters in the Inbound Direction Using Correlation Sets](#)

A.2.1 Design Time: Unsupported or Undefined Parameter Types

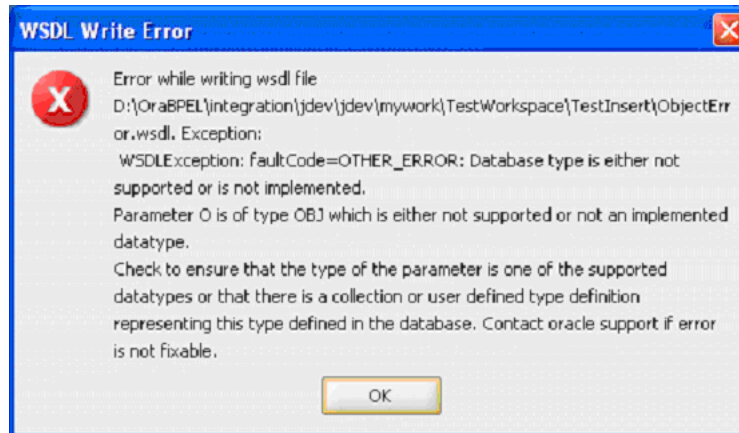
Problem

Using an unsupported or undefined parameter type in the chosen API is a common problem. Consider the following procedure:

```
PROCEDURE PROC (O OBJ) AS BEGIN ... END;
```

In this example, OBJ refers to a type that is undefined.

After you click **Finish** on the final page of the wizard, an attempt to generate the XSD is made, which produces the following error message:



The message indicates that the named parameter, O, which is of type OBJ, is either not defined or is otherwise inaccessible.

To generate an XSD for APIs containing parameters whose types are user-defined, those types must first be defined in the database and be accessible through the associated service connection. This error also occurs if such types have not been created (that is, *implemented*) in the database or if the database is inaccessible.

Solution

Ensure that only supported datatypes are used as types for parameters when choosing an API. If the types are user-defined, check to ensure that the types are defined in the database and that the database is accessible when the attempt to generate the XSD is made.

A.2.2 Design Time: Referencing User-Defined Types in Other Schemas

Problem

When the type of one or more of the parameters in the chosen API is a user-defined type that belongs to a different schema, a design-time problem can occur.

Assume type OBJ is created in SCHEMA2, as in

```
CREATE TYPE OBJ AS OBJECT (X NUMBER, Y VARCHAR2 (10));
```

And, a procedure is created in SCHEMA1 that has a parameter whose type is SCHEMA2.OBJ, as in

```
CREATE PROCEDURE PROC (O SCHEMA2.OBJ) AS BEGIN ... END;
```

If you attempt to create the XSD for procedure PROC, you may see the following error:

PLS-00201: identifier SCHEMA1.OBJ must be declared

Solution

SCHEMA1 must grant permission to SCHEMA2 so that SCHEMA2 can refer to type OBJ from SCHEMA1, as in

```
SQL> GRANT EXECUTE ON OBJ TO SCHEMA2;
```

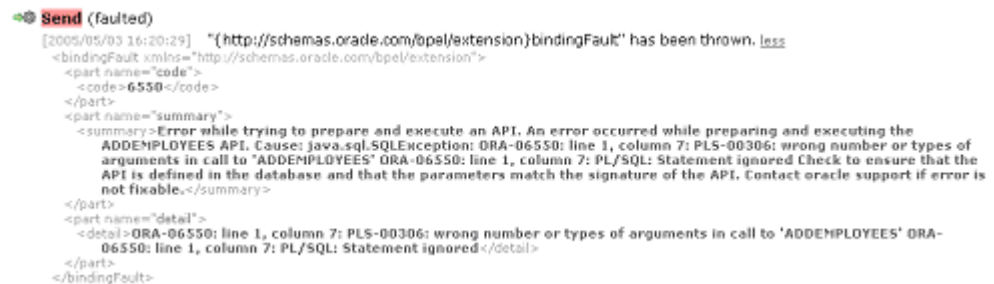
See "Referencing Types in Other Schemas" for more information.

A.2.3 Run Time: Parameter Mismatches

Problem

A mismatch between the formal parameters provided by the instance XML and the actual parameters that are defined in the signature of the stored procedure is a common run-time problem. When this type of error occurs, the **invoke** activity that tried to execute the stored procedure faults, as shown in [Figure A-1](#).

Figure A-1 Example of a Faulted Invoke Due to Mismatched Parameters



```
[2005/05/03 16:20:29] "{http://schemas.oracle.com/bpel/extension}bindingFault" has been thrown. less
<bindingFault xmlns="http://schemas.oracle.com/bpel/extension">
  <part name="code">
    <code>6550</code>
  </part>
  <part name="summary">
    <summary>Error while trying to prepare and execute an API. An error occurred while preparing and executing the
      ADDEMPLOYEES API. Cause: java.sql.SQLException: ORA-06550: line 1, column 7: PLS-00306: wrong number or types of
      arguments in call to 'ADDEMPLOYEES' ORA-06550: line 1, column 7: PL/SQL: Statement ignored Check to ensure that the
      API is defined in the database and that the parameters match the signature of the API. Contact oracle support if error is
      not fixable.</summary>
  </part>
  <part name="detail">
    <detail>ORA-06550: line 1, column 7: PLS-00306: wrong number or types of arguments in call to 'ADDEMPLOYEES' ORA-
      06550: line 1, column 7: PL/SQL: Statement ignored</detail>
  </part>
</bindingFault>
```

The `bindingFault` has three parts—code, summary, and detail. The information for these parts comes from the `java.sql.SQLException` that gets thrown by JDBC when it attempts to execute the stored procedure. The code is the ORA error number, seen in [Figure A-1](#) as 6550 and as ORA-06550 in the summary and detail parts. The summary includes an adapter-specific error message followed by the message from the `SQLException` and a suggested resolution to the issue. The detail contains just the message from the `SQLException`.

As shown in [Figure A-1](#), the `ADDEMPLOYEES` stored procedure failed to execute due to "wrong number or types of arguments" passed into the API. Possible causes for this problem include:

- An element corresponding to one of the required parameters was not provided in the instance XML.
Solution: Add the necessary element to resolve the issue.
- More elements than were specified in the XSD were included in the instance XML.
Solution: Remove the extra elements from the XML.
- The XSD does not accurately describe the signature of the stored procedure. For example, if the type of one of the parameters were to change and the XSD was not regenerated to reflect that change, a type mismatch can occur between the `db:type` of the element and the new type of the modified parameter.

Solution: Ensure that the parameters match the signature of the API, as indicated in the summary part of the `bindingFault`.

A.2.4 Run Time: Stored Procedure Not Defined in the Database

Problem

A `bindingFault` can also occur if the stored procedure is not defined in the database when an attempt to execute it is made. In this example, the `ADDEMPLOYEES` API is invoked, but fails to execute because it is not defined. The **invoke** activity faults, as shown in [Figure A-2](#).

Figure A-2



```

[2005/05/03 16:03:04] "(http://schemas.oracle.com/bpel/extension)bindingFault" has been thrown. less
<bindingFault xmlns='http://schemas.oracle.com/bpel/extension'>
  <part name='code'>
    <code>6550</code>
  </part>
  <part name='summary'>
    <summary>Error while trying to prepare and execute an API. An error occurred while preparing and executing the
      ADDEMPLOYEES API. Cause: java.sql.SQLException: ORA-06550: line 1, column 7: PLS-00201: identifier 'ADDEMPLOYEES'
      must be declared ORA-06550: line 1, column 7: PL/SQL: Statement ignored Check to ensure that the API is defined in the
      database and that the parameters match the signature of the API. Contact oracle support if error is not
      fixable.</summary>
  </part>
  <part name='detail'>
    <detail>ORA-06550: line 1, column 7: PLS-00201: identifier 'ADDEMPLOYEES' must be declared ORA-06550: line 1, column 7:
    PL/SQL: Statement ignored</detail>
  </part>
</bindingFault>

```

PL/SQL is revealing that "... identifier `ADDEMPLOYEES` must be declared," which is an indication that the stored procedure may not be defined in the database. This can occur, for example, if the procedure was dropped sometime between when the process was deployed and when the procedure was invoked. This can also occur if the required privileges to execute the stored procedure have not been granted.

Solution

Ensure that the API is defined in the database, as indicated in the summary, and that the appropriate privileges to execute that procedure have been granted.

Some run-time errors can occur if the instance XML does not conform to the XSD generated for the chosen API. XML validation can be enabled in the partner link that coincides with the execution of the API. Edit the `partnerLinkBinding` in the BPEL suitcase located in your process `bpel.xml` file, as shown in bold:

```

<partnerLinkBinding name="PROC">
  <property name="wsdlLocation">Proc.wsdl</property>
  <bproperty name="validateXML">true</bproperty>
</partnerLinkBinding>

```

Adding the `validateXML` property and setting it to `true` enables XML validation for all instance XML passed into the service associated with the partner link that executes the API.

XML validation can also be enabled globally in Oracle BPEL Control. Click **Manage BPEL Domain** and look for the **validateXML** property. Setting the value to **true** causes all XML to be validated. Turning on XML validation helps catch and avoid problems that are associated with the instance XML (rather than the adapter run time).

A.2.5 Configuring Multiple Adapters in the Inbound Direction Using Correlation Sets

Problem

When multiple adapter-based receive activities in the inbound direction use correlation sets in a process, the wrong property alias query is evaluated and the process fails at run time with the error:

```
Failed to evaluate correlation query
```

Workaround

As a workaround, ensure that the port type and operation values are unique between the two adapter WSDL files. For example, ensure that each adapter WSDL file has a unique operation name.

A.3 Troubleshooting the Oracle Application Server Adapter for Files/FTP

The following sections describe possible issues and solutions when using the Oracle Application Server Adapter for Files/FTP (file and FTP adapters):

- [Changing Logical Names with the Adapter Configuration Wizard](#)
- [Creating File Names with Spaces with the Native Format Builder Wizard](#)
- [Common User Errors](#)

A.3.1 Changing Logical Names with the Adapter Configuration Wizard

If you later rerun the Adapter Configuration Wizard and change a previously specified logical name to a different name, both the old and new logical names appear in the `bpel.xml` file. You must manually edit the `bpel.xml` file to remove the old logical name.

A.3.2 Creating File Names with Spaces with the Native Format Builder Wizard

While the Native Format Builder Wizard does not restrict you from creating native schema file names with spaces, it is recommended that your file names do not have spaces in them.

A.3.3 Common User Errors

This section describes common user errors.

- On the Adapter Configuration Wizard - Messages window ([Figure 2–9](#)), you can select the **Native format translation is not required (Schema is Opaque)** check box. Opaque cannot be selected in only one direction. Opaque must be selected in the both inbound and outbound directions.
- Messages have a different meaning based on whether they are inbound or outbound. For example, assume you make the following selections:
 - Select **2** from the **Publish Messages in Batches of** list ([Figure 2–7](#)) in the inbound direction.
 - Select **3** from the **Number of Messages Equal** list ([Figure 2–14](#)) in the outbound direction.

If an inbound file contains two records, it is split (debatched) into two messages. However, because **3** was specified in the outbound direction, a file is not created.

This is because there are not three outbound messages available. Ensure that you understand the meaning of inbound and outbound messages before selecting these options.

- If the file adapter or the FTP adapter is not able to read or get your file, respectively, it may be because you selected to match file names using the regular expression (regex), but are not correctly specifying the name ([Figure 2-7](#)). See [Table 2-3](#) for details.
- You may have content that does not require translation (for example, a JPG or GIF image) that you just want to send "as is." The file is passed through in base-64 encoding. This content is known as opaque. To do this, select the **Native format translation is not required (Schema is Opaque)** check box on the Adapter Configuration Wizard - Messages window ([Figure 2-9](#)). If you select this check box, you do not need to specify an XSD file for translation.
- The inbound directory *must* exist for the file adapter or the FTP adapter to read or get your file, respectively.
- If the FTP adapter cannot connect to a remote host, ensure that you have configured the `Oracle_Home\bpel\system\appserver\oc4j\j2ee\home\application-deployments\default\FtpAdapter\oc4j-ra.xml` deployment descriptor file for adapter instance JNDI name and FTP server connection information. See "[FTP Adapter for Get File Concepts](#)" for instructions.

```
Oracle_Home\bpel\system\appserver\oc4j\j2ee\home\application-deployments\default\FtpAdapter\oc4j-ra.xml
```

- You *cannot* use completely static names such as `po.txt` for outbound files. Instead, outgoing file names *must* be a combination of static and dynamic portions. This is to ensure the uniqueness of outgoing files names, which prevents files from being inadvertently overwritten. See "[Specifying the Outbound File Naming Convention](#)" for instructions on creating correct outbound file names.
- Two header files are created in the **Applications Navigator** after you finish running the Adapter Configuration Wizard in both directions:
 - `typeAdapterInboundHeader.wsdl`
Provides information such as the name of the file being processed and its directory path, as well as data about which message and part define the header operations
 - `typeAdapterOutboundHeader.wsdl`
Provides information about the outbound file name

where *type* is either `ftp` or `file`.

You can define properties in these header files. For example, you can specify dynamic inbound and outbound file names through use of the `InboundHeader_msg` and `OutboundHeader_msg` message names in the `typeAdapterInboundHeader.wsdl` and `typeAdapterOutboundHeader.wsdl` files, respectively.

You can also set header variables that appear in the BPEL process file. Header variables are useful for certain scenarios. For example, in a file propagation scenario, files are being moved from one file system to another using the file adapter. In this case, it is imperative that you maintain file names across the two

systems. Use file headers in both directions and set the file name in the outbound file header to use the file name in the inbound file header.

See the online help available with the Adapters tab of invoke, receive, reply, and pick - OnMessage branch activities for more information.

- The Adapter Configuration Wizard - File Modification Time window (Figure 2–24) prompts you to select a method for obtaining the modification times of files on the FTP server.

You must perform the following procedure to obtain this information:

1. Determine the modification time format supported by the FTP Server by running the command `mdtm` or `ls -al` (whichever is supported by the operating system).
2. Determine the time difference between the system time (time on which Oracle BPEL Server is running) and the file modification time. Obtain the file modification time by running either `mdtm` or `ls -al` on the FTP server.
3. Manually add the time difference to the `bpel.xml` as a property:

```
<activationAgents>
  <activationAgent ...>
    <property name="timestampOffset">2592000000</property>
```

4. Specify the **Substring Begin Index** field and **End Index** field values that you determine by running the `mdtm` or `ls -al` command on the FTP server.

A.4 Troubleshooting the Oracle Application Server Adapter for Advanced Queuing

The following sections describe possible issues and solutions when using the Oracle Application Server Adapter for Advanced Queuing (AQ adapter):

- [Inbound Errors](#)
- [Outbound Errors](#)
- [JDeveloper BPEL Designer Errors](#)
- [Translation Error](#)
- [Other Problems](#)

A.4.1 Inbound Errors

The following sections describe possible issues and solutions for inbound errors when using the AQ adapter.

A.4.1.1 JNDI Lookup Failed

Sample error:

```
<timestamp> <WARN> <default.collaxa.cube.activation> <AdapterFramework::Inbound>
  JNDI lookup of 'eis/AQ/aqSample2' failed due to: eis/AQ/aqSample2 not found
```

```
<timestamp> <ERROR> <default.collaxa.cube.activation> <AdapterFramework::Inbound>
  Error while performing endpoint Activation: ORABPEL-12510
```

Unable to locate the JCA Resource Adapter via WSDL port element `jca:address`.

The Adapter Framework is unable to startup the Resource Adapter specified in the

```
WSDL jca:address element: {http://xmlns.oracle.com/pcbpel/wsdl/jca/}address:
location='eis/AQ/aqSample2'
```

Problem

It is likely that either 1) the resource adapter's RAR file has not been deployed successfully to the OC4J Application Server or 2) the location attribute in `$J2EE_HOME/application-deployments/default/deployed-adapter-name/oc4j-ra.xml` has not been set to `eis/AQ/aqSample2`. In the last case, you may have to add a new connector-factory entry (connection) to the `oc4j-ra.xml`. Correct this and then restart the BPEL/OC4J Application Server.

Solution

1. Look for the file

```
$J2EE_HOME/application-deployments/default/aqAdapter/
oc4j-ra.xml.
```

This file should be created when the adapter is deployed, which occurs the first time Oracle BPEL Process Manager is started. If the adapter is undeployed for some reason, deploy the adapter with the following command, then follow step 2:

```
java -jar $J2EE_HOME/admin.jar ormi://localhost admin welcome
-deployconnector -file <path to AQAdapter.rar> -name
AqAdapter>
```

2. If `$J2EE_HOME/application-deployments/default/aqAdapter/oc4j-ra.xml` exists, make sure the JNDI location is defined in `oc4j-ra.xml`.

For example:

```
<connector-factory location="eis/AQ/aqSample2" connector-name="AQ Adapter">
  <config-property name="connectionString"
    value="jdbc:oracle:thin:@myhost:1521:appdb2"/>
  <config-property name="userName" value="scott"/>
  <config-property name="password" value="tiger"/>
</connector-factory>
```

A.4.1.2 During Initialization, I/O Exception: Network Adapter Did Not Establish the Connection

Sample error:

```
<timestamp> <ERROR> <default.collaxa.cube.activation> <AQ Adapter::Inbound>
DBConnection_connect: database error while try to connect to
jdbc:oracle:thin:@localhost:1521:ORCL : Io exception: The Network Adapter could
not establish the connection
```

Solution

If the `connectionString` is correct, make sure the database and listener are up, then redeploy the process.

If the `connectionString` is not correct:

1. Change the `connectionString` in `$J2EE_HOME\application-deployments\default\AqAdapter\oc4j-ra.xml`.
2. Restart Oracle BPEL Process Manager.

A.4.1.3 Incorrect Username/Password

Sample error:

```
<timestamp> <ERROR> <default.collaxa.cube.activation> <AdapterFramework::Inbound>
  Error while performing endpoint Activation: ORABPEL-11929
SQL error while creating managed (database) connection.
SQL error while creating managed (database) connection: Error while trying to
  connect to database.
Error while trying to connect to database using connect string
  "jdbc:oracle:thin:@localhost:1521:appdb2 - java.sql.SQLException: ORA-01017:
  invalid username/password; logon denied
```

Solution

1. Make sure you have the correct username and password in \$J2EE_HOME\application-deployments\default\AqAdapter\oc4j-ra.xml.
2. Restart Oracle BPEL Process Manager.

A.4.1.4 Queue Not Found

Sample error:

```
<timestamp> <ERROR> <default.collaxa.cube.activation> <AQ Adapter::Inbound>
  oracle.AQ.AQException: JMS-190: Queue SCOTT.AQ_SUPPORTED_ADT_IN not found
at oracle.AQ.AQUtil.throwAQEx(AQUtil.java:196)
at oracle.AQ.AQOracleSession.getQueue(AQOracleSession.java:720)
at oracle.tip.adapter.aq.database.Queue.connect(Queue.java:102)
at oracle.tip.adapter.aq.database.MessageReader.init(MessageReader.java:575)
```

Solution

Create the queue and redeploy the process. If this process is deployed from the samples, all queue creation scripts are located in sql\create_queues.sql under each project.

A.4.1.5 User Does Not Have DBMS_AQIN Privileges, Which Are Required by the AQ Java API

Sample error:

```
2005-04-20 16:10:52,695> <ERROR> <default.collaxa.cube.activation> <AQ
  Adapter::Inbound> oracle.AQ.AQOracleSQLException: ORA-06550: line 1, column 7:
  PLS-00201: identifier 'DBMS_AQIN' must be declared
ORA-06550: line 1, column 7:
  PL/SQL: Statement ignored
at oracle.AQ.AQOracleQueue.dequeue(AQOracleQueue.java:1795)
at oracle.AQ.AQOracleQueue.dequeue(AQOracleQueue.java:1307)
at
  oracle.tip.adapter.aq.database.MessageReader.readMessage(MessageReader.java:399)
at
  oracle.tip.adapter.aq.inbound.AQActivationSpecDequeuer.run(AQActivationSpecDequeuer
  .java:163)
at oracle.tip.adapter.fw.jca.work.WorkerJob.go(WorkerJob.java:51)
at oracle.tip.adapter.fw.common.ThreadPool.run(ThreadPool.java:267)
at java.lang.Thread.run(Thread.java:534)
```

Solution

Log on to the database using sys as sysdba, GRANT EXECUTE ON SYS.DBMS_AQIN to <username>; . No deployment is necessary because this failure occurs after the

connection has succeeded, the adapter automatically reconnects until the error is gone or the process is undeployed.

A.4.1.6 Translation Error

Sample error:

```
<timestamp> <ERROR> <default.collaxa.cube.activation> <AQ Adapter::Inbound>
  MessageReader_readMessage: Received TranslationException
<timestamp> <ERROR> <default.collaxa.cube.activation> <AQ Adapter::Inbound>
  ORABPEL-11211
  DOM Parsing Exception in translator.
  DOM parsing exception in inbound XSD translator while parsing InputStream.
  Check the error stack and fix the cause of the error. Contact oracle support if
  error is not fixable.
  at
  oracle.tip.pc.services.translation.xlators.xsd.XSDTranslator.translateFromNative(X
  SDTranslator.java:131)
```

Solution

Look for the rejected message and find out why it has failed translation. For instance, the message may not be XML, or the XML root element may be incorrect, or the message may be blank. If a rejection handler has been defined for this process, look for the message in the rejection handler. Otherwise, look for the message in the default rejection handler, which is located at

```
Oracle_Home\bpel\domains\default\archive\jca
\AQMessageRejectionHandler\rejectedMessages
```

For an example of how a user can define the rejection handler, look in

```
ORACLE_
HOME\bpel\samples\tutorials\124.AQAdapter\AQMessageRejectionHan
dler
```

A.4.1.7 Subscriber Already Exists When Using MessageRuleSelector

Sample error:

```
<timestamp> <INFO> <default.collaxa.cube.activation> <AQ Adapter::Inbound>
  Subscriber PriorityOneDequeuer already exists in the database. If the subscriber
  does not contain the rule that you want, please undeploy the business process,
  drop the subscriber with the following sql*plus command, and redeploy. DECLARE
  subscriber sys.aq$_agent;
  BEGIN
    subscriber := sys.aq$_agent('<subscriber_name>', NULL, NULL);
    DBMS_AQADM.REMOVE_SUBSCRIBER(
      queue_name => '<queue_name>',
      subscriber => subscriber); END;
```

Solution

This is not a problem if the subscriber has been generated with the rule the user expects. The adapter can create new rule-based subscribers, but cannot modify existing ones. Hence, the first time you deploy the adapter with a nonnull value for MessageSelectorRule, a subscriber is created if the consumer does not already exist, using the consumer as the subscriber and the MessageSelectorRule as the rule. This message appears in any subsequent redeployment or restart of Oracle BPEL Process Manager.

You can determine if the rule is what you want for the subscriber by entering the following SQL command:

```
SQL> select name, rule, queue from AQ$RuleBased_Raw_In_R;
NAME
-----
RULE
-----
QUEUE
-----
PRIORITYONEDEQUEUER
priority = 1
RULEBASED_RAW_IN
```

A.4.2 Outbound Errors

As a general note, problems in the outbound direction are often not caught at deployment time, because an outbound thread is only activated if there is a message going to outbound.

A.4.2.1 JNDI Lookup Failed

Sample error:

```
Adapter Framework unable to create outbound JCA connection.
file:/C:/050420/bpel/domains/default/tmp/.bpel_File2AQBLOB_
1.0.jar/EnqueueBlobPayload.wsdl [ Enqueue_ptt::Enqueue(opaque) ] - : The Adapter
Framework was unable to establish an outbound JCA connection due to the following
issue: ORABPEL-12510
Unable to locate the JCA Resource Adapter via WSDL port element jca:address.
The Adapter Framework is unable to startup the Resource Adapter specified in the
WSDL jca:address element: {http://xmlns.oracle.com/pcbpel/wsdl/jca/}address:
location='eis/AQ/aqSample3'
```

The reason for this is most likely that either 1) the resource adapter's RAR file has not been deployed successfully to the OC4J Application server or 2) the location attribute in \$J2EE_HOME/application-deployments/default/deployed-adapter-name/oc4j-ra.xml has not been set to eis/AQ/aqSample3. In the last case you may have to add a new connector-factory entry (connection) to oc4j-ra.xml. Correct this and then restart the BPEL/OC4J Application Server

Solution

See the solution section for the same problem in the inbound section, as described in ["Inbound Errors"](#).

A.4.2.2 I/O Exception: Network Adapter Could Not Establish the Connection

```
2005-04-20 18:41:40,570> <ERROR> <default.collaxa.cube.ws>
<AdapterFramework::Outbound>
file:/C:/050420/bpel/domains/default/tmp/.bpel_File2AQBLOB_
1.0.jar/EnqueueBlobPayload.wsdl [ Enqueue_ptt::Enqueue(opaque) ] - Could not
invoke operation 'Enqueue' against the 'AQ Adapter' due to: ORABPEL-12511
Adapter Framework unable to create outbound JCA connection.
file:/C:/050420/bpel/domains/default/tmp/.bpel_File2AQBLOB_
1.0.jar/EnqueueBlobPayload.wsdl [ Enqueue_ptt::Enqueue(opaque) ] - : The Adapter
Framework was unable to establish an outbound JCA connection due to the following
issue: ORABPEL-11929
SQL error while creating managed (database) connection.
SQL error while creating managed (database) connection: Error while trying to
```

```
connect to database.
Error while trying to connect to database using connect string
"jdbc:oracle:thin:@localhost:1521:appdb - java.sql.SQLException: Io exception:
The Network Adapter could not establish the connection"
```

Solution

If the `connectionString` is not correct, do the following:

1. Change the `connectionString` in `$J2EE_HOME\application-deployments\default\AqAdapter\oc4j-ra.xml`.
2. Restart Oracle BPEL Process Manager.

If the `connectionString` is correct, make sure the database and listener are up. If you had enabled outbound retry, the message should be automatically retried when the database and its listener are up.

To configure outbound retry, set the `retryMaxCount` property and `retryInterval` property for the partner link in `bpel.xml`.

For example, the following configuration means 10 retries, at 60 second intervals.

```
<partnerLinkBinding name="EnqueueBLOBPayload">
    <property name="wsdlLocation">EnqueueBlobPayload.wsdl</property>
    <property name="retryMaxCount">10</property>
    <property name="retryInterval">60</property>
</partnerLinkBinding>
```

A.4.2.3 Queue Not Found

```
<timestamp> <ERROR> <default.collaxa.cube.ws> <AQ Adapter::Outbound>
oracle.AQ.AQException: JMS-190: Queue SCOTT.BLOBPAYLOAD_QUEUE not found
    at oracle.AQ.AQUtil.throwAQEx(AQUtil.java:196)
    at oracle.AQ.AQOracleSession.getQueue(AQOracleSession.java:720)
    at oracle.tip.adapter.aq.database.Queue.connect(Queue.java:102)
    at oracle.tip.adapter.aq.database.MessageWriter.init(MessageWriter.java:231)
```

Solution

Same solution as the inbound Queue not found problem. Create the queue and redeploy the process. If this process is deployed from the samples, all queue creation scripts are located in `sql\create_queues.sql` under each project.

A.4.2.4 Incorrect Username/Password

Sample error:

```
file:/C:/050420/bpel/domains/default/tmp/.bpel_File2AQBLOB_
1.0.jar/EnqueueBlobPayload.wsdl [ Enqueue_ptt::Enqueue(opaque) ] - : The Adapter
Framework was unable to establish an outbound JCA connection due to the following
issue: ORABPEL-11929
SQL error while creating managed (database) connection.
SQL error while creating managed (database) connection: Error while trying to
connect to database.
Error while trying to connect to database using connect string
"jdbc:oracle:thin:@localhost:1521:appdb2 - java.sql.SQLException: ORA-01017:
invalid username/password; logon denied.
```

Solution

1. Make sure you have the correct username and password in `$J2EE_HOME\application-deployments\default\AqAdapter\oc4j-ra.xml`.

2. Restart Oracle BPEL Process Manager.

A.4.2.5 User Does Not Have DBMS_AQIN Privileges, Which Are Required by the AQ Java API

Sample error:

```
<timestamp> <ERROR> <default.collaxa.cube.ws> <AQ Adapter::Outbound>
  oracle.AQ.AQOracleSQLException: ORA-06550: line 1, column 7:
PLS-00201: identifier 'DBMS_AQIN' must be declared
ORA-06550: line 1, column 7:
PL/SQL: Statement ignored
    at oracle.AQ.AQOracleQueue.enqueue(AQOracleQueue.java:1267)
```

Solution

Log on to the database using sys as sysdba, GRANT EXECUTE ON SYS.DBMS_AQIN to <username>; . Again, if you have retry configured for this partner link, retry automatically happens.

A.4.2.6 Translation Error

Sample error:

```
<timestamp> <ERROR> <default.collaxa.cube.ws> <AQ Adapter::Outbound> ORABPEL-11101
Translation Failure.
Translation to native failed. Invalid text 'blahblah' in element: 'Root-Element'.
Check the error stack and fix the cause of the error. Contact oracle support if
  error is not fixable.
at
oracle.tip.pc.services.translation.xlators.nxsd.NXSDTranslatorImpl.translateToNative(NXSDTranslatorImpl.java:502)
at
oracle.tip.adapter.aq.database.MessageWriter.translateToNative(MessageWriter.java:1102)
at oracle.tip.adapter.aq.database.MessageWriter.doEnqueue(MessageWriter.java:494)
```

Solution

1. From Oracle BPEL Control, click the **Instances** tab.
2. Click the failed instance and select **Debug**.
3. Look for the variable passed to the invoke activity that failed. You should notice this variable match schema definition. Back track in the **Debug** window to find out why.

A.4.3 JDeveloper BPEL Designer Errors

I have an AQ inbound to AQ outbound end-to-end scenario. How do I copy the priority from an inbound queue to an outbound queue?

Solution: Create both the inbound header and outbound header, and use an assign activity to copy the priority.

Create the inbound header:

1. Click the **receive** activity that is linked to the AQ Inbound partner link.
2. Select the **Adapters** tab and click the **flashlight** icon to the right of the **Header Variable** field.
3. Right click **Variables** and select **Create Variable**.

4. Enter a name such as *inbound_header*.
5. Click **Message type** and the **flashlight** icon.
6. If a `payloadHeader` is not required (most cases):
In the Type Chooser window, click **Type Explorer > Message Types > Partner Links > inbound_partnerlink_name > inbound_service.wsdl > Imported WSDL > aqAdapterInboundHeader.wsdl > Message Types > Header**. Go to step 8.
7. If a `payloadHeader` is required (only if `PayloadHeaderRequired="true"` in the JCA operation section of *service_name.wsdl*):
In the Type Chooser window, click **Type Explorer > Message Types > Partner Links > inbound_partnerlink_name > Message Types > Header_msg**.
8. Click **OK** to exit the Type Chooser window.
9. While the variable is still highlighted, click **OK** to pick this variable.
The variable now appears as the header variable.
10. Click **OK** to exit the **receive** activity.

Create the outbound header:

1. Click the **invoke** activity that is linked to the AQ outbound partner link.
2. Select the **Adapters** tab and click the **flashlight** icon to the right of the **Header Variable** field.
3. Right click **Variables** and select **Create Variable**.
4. Enter a name such as *outbound_header*.
5. Click **Message type** and click the **flashlight** icon.
6. If a `payloadHeader` is not required (most cases):
In the Type Chooser window, click **Type Explorer > Message Types > Partner Links > outbound_partnerlink_name > outbound_service.wsdl > Imported WSDL > aqAdapterOutboundHeader.wsdl > Message Types > Header**. Go step 8.
7. If a `payloadHeader` is required, only if `PayloadHeaderRequired="true"` in the JCA operation section of *service_name.wsdl*:
In the Type Chooser window, click **Type Explorer > Message Types > Partner Links > outbound_partnerlink_name > Message Types > Header_msg**.
8. Click **OK** to exit the Type Chooser window.
9. While the variable is still highlighted, click **OK** to pick this variable.
10. The variable now appears as the header variable. Click **OK** to exit the **receive** activity.

In an Assign activity:

1. Click the **Copy Rules** tab and select **Create**.
2. In the **From** section: drill down to **priority** for the **inbound header variable**.
3. In the **To** section: drill down to **priority** for the **outbound header variable**.
4. Click **OK** to exit Create Copy Rule window.
5. Click **OK** to exit the Assign window.

How to delete header variables in Adapters?

Assuming that you have created a header variable for an adapter as follows, you cannot subsequently delete that variable through the same receive window. The workaround is to delete the header variable in the BPEL source code.

1. Create a BPEL project.
2. Double-click the **receive** activity.
3. Click the **Adapters** tab and define a header variable for the adapter.

I redefined the adapter WSDL by stepping through the wizard again. Why doesn't my service_name.wsdl change?

Solution:

It did work, but JDeveloper BPEL Designer did not refresh the file properly. Close the *service_name.wsdl* file and open it again.

I redefined the adapter service WSDL using the wizard. But at deployment time, I got the following error:

```
Process "AQSupportedADTTypes" (revision "1.0") compilation failed.
<timestamp> <ERROR> <default.collaxa.cube.engine.deployment>
  <CubeProcessLoader::create> BPEL validation failed.
BPEL source validation failed, the errors are:
[Error ORABPEL-10007]: unresolved messageType
[Description]: in line 16 of
  "C:\050420\bpel\domains\default\tmp\.bpel_AQSupportedADTTypes_
1.0.jar\AQSupportedADTTypes.bpel", WSDL messageType
  "{http://xmlns.oracle.com/pcbpel/adapter/aq/Enqueue/}Header_msg" of variable
  "out_header" is not defined in any of the WSDL files.
[Potential fix]: Make sure the WSDL messageType
  "{http://xmlns.oracle.com/pcbpel/adapter/aq/Enqueue/}Header_msg" is defined in
  one of the WSDLs referenced by the deployment descriptor.
```

Solution:

When you redefine the adapter service WSDL, you also need to redefine the header variables. The creation of header variables the input element in the JCA binding section which defines the header.

For example:

```
<input>
  <jca:header message="tns:Header_msg" part="Header"/>
</input>
```

When the adapter service is redefined, the old WSDL file is overwritten. Delete the old header variables and re-create them.

A.4.4 Translation Error

Sample error:

```
<timestamp> <ERROR> <default.collaxa.cube.activation> <AQ Adapter::Inbound>
MessageReader_readMessage: Received TranslationException
<timestamp> <ERROR> <default.collaxa.cube.activation> <AQ Adapter::Inbound>
ORABPEL-11211
DOM Parsing Exception in translator.
DOM parsing exception in inbound XSD translator while parsing InputStream.
Check the error stack and fix the cause of the error. Contact oracle support
if error is not fixable.
```

```
at
oracle.tip.pc.services.translation.xlators.xsd.XSDTranslator.translateFromNative(X
SDTranslator.java:131)
```

Solution:

Look for the rejected message and find out why it has failed translation. The message may not be XML or the XML root element maybe incorrect, or the message may be blank. If a rejection handler has been defined for this process, look for the message in the rejection handler. Otherwise, look for the message in the default rejection handler which is located at:

```
ORACLE_HOME\
bpel\domains\default\archive\jca\AQMessageRejectionHandler\rejectedMessages
```

For an example of how a user can define the rejection handler, look in

```
ORACLE_
HOME\bpel\samples\tutorials\124.AQAdapter\AQMessageRejectionHand
ler
```

A.4.5 Other Problems

I have a new adapter *.RAR file; how do I redeploy the adapter?

1. Save a copy of
`$J2EE_HOME\application-deployments\default\AqAdapter\oc4j-ra.xml` so you have your endpoint information.
2. Undeploy the adapter by entering the following command:

```
java -jar $JE22_HOME\admin.jar ormi://localhost admin welcome
-undeployconnector -name AqAdapter
```
3. Deploy the new .rar file by entering the following command:

```
java -jar $J2EE_HOME/admin.jar ormi://localhost admin welcome
-deployconnector -file <rarfile> -name AqAdapter
```
4. Modify
`$J2EE_HOME\application-deployments\default\AqAdapter\oc4j-ra.xml` to add your endpoints.
5. Restart Oracle BPEL Process Manager. You should not have to redeploy your business processes.

Subscriber already exists when using MessageRuleSelector

```
<timestamp> <INFO> <default.collaxa.cube.activation> <AQ Adapter::Inbound>
Subscriber PriorityOneDequeuer already exists in the database. If the
subscriber does not contain the rule that you want, please undeploy the
business process, drop the subscriber with the following sql*plus command,
and redeploy.
```

```
DECLARE
    subscriber sys.aq$_agent;
BEGIN
    subscriber := sys.aq$_agent('<subscriber_name>', NULL, NULL);
DBMS_AQADM.REMOVE_SUBSCRIBER(
    queue_name => '<queue_name>',
    subscriber => subscriber);
END;
```

Solution:

This is not a problem if the subscriber has been generated with the rule the user expects. The adapter can create new rule-based subscribers, but cannot modify existing ones. Therefore, the first time you deploy the adapter with a nonnull value for `MessageSelectorRule`, a subscriber is created if the consumer does not already exist, using the consumer as the subscriber and the `MessageSelectorRule` as the rule. This message would appear in any subsequent redeployment or restart of Oracle BPEL Process Manager.

You can determine if the rule is what you want for the subscriber by entering the following SQL command: `select name, rule, queue from AQ$ QUEUE_`
`TABLE_NAME_R;`

```
SQL> select name, rule, queue from AQ$RuleBased_Raw_In_R;
NAME
-----
RULE
-----
QUEUE
-----
PRIORITYONEDEQUEUEER
priority = 1
RULEBASED_RAW_IN
```

You do not have DBMS_AQIN privileges, which are required by the AQ Java API

```
2005-04-20 16:10:52,695> <ERROR> <default.collaxa.cube.activation> <AQ
Adapter::Inbound>
oracle.AQ.AQOracleSQLException: ORA-06550: line 1, column 7:
PLS-00201: identifier 'DBMS_AQIN' must be declared
ORA-06550: line 1, column 7:
PL/SQL: Statement ignored
at oracle.AQ.AQOracleQueue.dequeue(AQOracleQueue.java:1795)
at oracle.AQ.AQOracleQueue.dequeue(AQOracleQueue.java:1307)
at
oracle.tip.adapter.aq.database.MessageReader.readMessage(MessageReader.java:399)
at
oracle.tip.adapter.aq.inbound.AQActivationSpecDequeuer.run(AQActivationSpecDequeuer.java:163)
at oracle.tip.adapter.fw.jca.work.WorkerJob.go(WorkerJob.java:51)
@ at oracle.tip.adapter.fw.common.ThreadPool.run(ThreadPool.java:267)
at java.lang.Thread.run(Thread.java:534)
```

Solution:

Log on to the database using `sys` as `sysdba` 'GRANT EXECUTE ON SYS.DBMS_AQIN to *username*;' . No deployment is necessary because this failure occurs after the connection has succeeded. The adapter automatically attempts to reconnect until the error is gone or the process is undeployed.

Failed JNDI Lookup

```
<timestamp> <WARN> <default.collaxa.cube.activation>
<AdapterFramework::Inbound> JNDI lookup of 'eis/AQ/aqSample2' failed due to:
eis/AQ/aqSample2 not found
<timestamp> <ERROR> <default.collaxa.cube.activation>
<AdapterFramework::Inbound> Error while performing endpoint Activation:
ORABPEL-12510
Unable to locate the JCA Resource Adapter via WSDL port element jca:address.
The Adapter Framework is unable to startup the Resource Adapter specified in
the WSDL jca:address element:
```

```
@ {http://xmlns.oracle.com/pcbpel/wsdl/jca/}address:
location='eis/AQ/aqSample2'
```

Solution:

The reason for this is most likely that either:

1. The resource adapter's RAR file has not been deployed successfully to the OC4J Application Server.
2. The Resource Adapters RAR file has not been deployed successfully to the location attribute in:

```
$J2EE_
HOME/application-deployments/default/deployed-adapter-name/oc
4j-ra.xml has not been set to eis/AQ/aqSample2.
```

In this case, you may have to add a new connector-factory entry (connection) to the oc4j-ra.xml file. Add the correct entry and then restart the BPEL Application Server.

How to fix:

1. Look for the file \$J2EE_
 HOME/application-deployments/default/aqAdapter/oc4j-ra.xml.
 This file should be created when the adapter is deployed, which occurs the first time Oracle BPEL Process Manager is started. If the adapter is undeployed for some reason, deploy the adapter with the following command, then follow step 2:

```
java -jar $J2EE_HOME/admin.jar ormi://localhost admin welcome
-deployconnector -file <path to AQAdapter.rar> -name
AqAdapter
```

2. If \$J2EE_
 HOME/application-deployments/default/aqAdapter/oc4j-ra.xml
 exists, make sure the JNDI location is defined in the oc4j-ra.xml file.

```
<connector-factory location="eis/AQ/aqSample" connector-name="AQ Adapter">
<config-property name="connectionString"
value="jdbc:oracle:thin:@myhost:1521:appdb2"/>
<config-property name="userName" value="scott"/>
@ <config-property name="password" value="tiger"/>
</connector-factory>
```

A.5 Troubleshooting the Oracle Application Server Adapter for Java Message Service (JMS)

JMS Provider Error

```
<JMSAdapter::Outbound>
ORABPEL-12165
ERRJMS_PROVIDER_ERR.
Could not produce message due to JMS provider error. Please examine the log
file to determine the problem.
.....
.....
.....
Caused by: javax.jms.JMSEException: MQJMS1013: operation invalid whilst
session is using asynchronous delivery.
```

Solution:

This exception occurs when the inbound JMS adapter and the outbound JMS adapter for MQ provider use same JNDI name. To avoid this exception, specify different JNDI names in WSDL files for inbound and outbound JMS adapter for MQ provider.

Another workaround is to specify the `UseMessageListener` property as `false` in the inbound WSDL file. For example:

```
UseMessageListener="false"
```

Index

A

- activationAgent property
 - fatalErrorFailoverProcess, 2-19
- ActivationSpec parameters, 2-22
- Adapter Configuration Wizard
 - configuring the database adapter, 4-23
 - configuring the file adapter, 4-25
 - design time, 4-37
 - starting, 1-1, 1-3, 2-5, 4-25
 - stored procedures, 4-100
- Adapter Configuration wizard
 - using with the AQ adapter, 3-6
 - using with the file adapter, 2-6, 6-5
 - using with the FTP adapter, 2-6
 - using with the MQSeries adapter, 6-5
- adapter services
 - defined in WSDL file, 4-73
- adapters
 - configuring, 1-1, 1-3
 - creating header variables, 3-25
 - definition, 1-1, 1-3
 - error handling, 2-18
 - in JDeveloper BPEL Designer, 1-1
 - in JDeveloper ESB Designer, 1-3
 - service names, 1-2, 1-4
- ADT payload types
 - AQ adapter, 3-4
- after-read strategy
 - database adapter, 4-34
- APIs, 4-105
- application.xml file
 - configuring OJMS, 5-18
 - configuring Tibco JMS, 5-20
- AQ adapter
 - ADT payload types, 3-4
 - AQ header properties, 3-3
 - correlation identifier, 3-2
 - dequeue mode, 3-3
 - enqueue-specific features, 3-2
 - features, 3-1
 - message priority, 3-2
 - multiconsumer queue, 3-2
 - payload schema, 3-5
 - use cases, 3-6
- AQ header properties

- AQ adapter, 3-3
- AQ headers, 3-25
- arrayIdentifierLength
 - construct, 7-53
- arrayLength
 - construct, 7-53
- arrays
 - native schema, 7-30
- arrayTerminatedBy
 - construct, 7-54
- assign
 - construct, 7-54
- audit link
 - instance execution process, 4-141

B

- batch processing
 - support with the file and FTP adapters, 2-11
- batching
 - definition, 2-15
- byteOrder
 - construct, 7-53

C

- cellSeparatedBy
 - construct, 7-54
- choiceCondition
 - construct, 7-54
- clauses
 - clauses to add to impact the sign position in
 - COBOL Copybook, 7-6
 - supported COBOL Copybook clauses, 7-4
- COBOL Copybook
 - clauses to add to impact the sign position, 7-6
 - definition, 7-3
 - multiple root levels, 7-14
 - Native Format Builder wizard support, 7-3
 - numeric types, 7-21
 - single root level, virtual decimal, fixed length
 - array, 7-17
 - supported clauses, 7-4
 - use cases, 7-14
 - user inputs, 7-4
 - variable length array, 7-20

- com.ibm.mq.jar, 6-34
- compiling
 - FulfillOrder process, 4-140
- complex structure
 - native schema, 7-11
- conditional processing
 - native schema, 7-36
- conditionValue
 - construct, 7-54
- config timeout parameter
 - configuring in the server.xml file, 2-17
- connection pooling
 - database adapter, 4-90
- constructs
 - arrayIdentifierLength, 7-53
 - arrayLength, 7-53
 - arrayTerminatedBy, 7-54
 - assign, 7-54
 - byteOrder, 7-53
 - cellSeparatedBy, 7-54
 - choiceCondition, 7-54
 - conditionValue, 7-54
 - dateFormat, 7-54
 - encoding, 7-53
 - headerLines, 7-53
 - headerLinesTerminatedBy, 7-53
 - identifierLength, 7-54
 - itemSeparatedBy, 7-54
 - leftsurroundedBy, 7-54
 - length, 7-54
 - listTerminatedBy, 7-54
 - lookAhead, 7-54
 - native schema, 7-53
 - paddedBy, 7-54
 - padStyle, 7-54
 - quotedBy, 7-54
 - rightsurroundedBy, 7-54
 - skip, 7-54
 - skipLines, 7-54
 - skipUntil, 7-54
 - standalone, 7-53
 - startsWith, 7-54
 - stream, 7-53
 - style, 7-54
 - surroundedBy, 7-54
 - terminatedBy, 7-54
 - uniqueMessageSeparator, 7-53
 - variable, 7-54
 - variables, 7-54
 - version, 7-53
 - xmlversion, 7-53
- control table
 - database adapter, 4-21
- copy rules
 - creating, 4-138, 4-139
- correlation identifier
 - AQ adapter, 3-2
- Correlation Schemas, 6-24
- creating relationships
 - database adapter, 4-29

- CSV
 - native schema, 7-8

D

- database adapter
 - advanced use cases for outbound invoke operations, 4-15
 - after-read strategy, 4-34
 - choosing a polling strategy, 4-90
 - connection pooling, 4-90
 - control table, 4-21
 - creating an invoke activity, 4-134
 - creating relationships, 4-29
 - creating the object model, 4-32
 - datatype conversions, 4-120, 4-121
 - defining keys, 4-28
 - defining the WHERE clause, 4-32
 - deleting the rows that were read, 4-35
 - deployment, 4-79
 - design overview, 4-7
 - existence checking, 4-90
 - features, 4-1
 - function return values, 4-122
 - generated XML schema, 4-78
 - importing and selecting tables, 4-28
 - inbound distributed polling, 4-91
 - last updated, 4-20
 - last-read ID, 4-19
 - locking, 4-91
 - logical delete, 4-17
 - mapping any relational schema to any XML schema types, 4-13
 - maxRaiseSize, 4-90
 - merge operations, 4-13
 - null values, 4-121
 - operation type, 4-26
 - performance, 4-88
 - physical delete, 4-16
 - polling strategies, 4-15
 - query by example operations, 4-14
 - REF CURSOR support, 4-122
 - relational types to XML schema types, 4-12
 - relational-to-XML mappings, 4-9, 4-71
 - relationship reading, 4-90
 - running the FulfillOrder process, 4-140
 - sequencing table, 4-19, 4-20, 4-36
 - SQL operations as Web services, 4-13
 - stored procedure design time WSDL and XSD generation, 4-111
 - stored procedures and functions, 4-100, 4-118
 - supported primitive datatypes, 4-113
 - TopLink Mapping Workbench project, 4-67
 - toplink_mappings.xml file, 4-8
 - update a field in the table, 4-35
 - use cases for outbound invoke operations, 4-14
 - use cases for Pure SQL, 4-15
 - validating, compiling, and deploying, 4-140
 - value binding, 4-118
- database connection

- configuring in the oc4j-ra.xml file, 4-7, 4-26
- database operations
 - DML operations, 4-13
- datatype conversions
 - database adapter, 4-120, 4-121
- dateFormat
 - construct, 7-54
- dates
 - native schema, 7-43
- DBActivationSpec, 4-76
- DBReadInteractionSpec, 4-75
- DBWriteInteractionSpec, 4-74
- debatching
 - definition, 2-3, 2-15
 - file adapter use case, 2-64
 - supported with the file and FTP adapters, 2-3
- defining keys
 - database adapter, 4-28
- deleting rows that were read
 - database adapter, 4-35
- delimited format
 - Native Format Builder wizard support, 7-3
- delimiters
 - file adapter use case, 2-64
- deploying
 - FulfillOrder process, 4-140
- deployment
 - of database adapter, 4-79
- dequeue mode
 - AQ adapter, 3-3
- Distribution List Support, 6-25
- DML operations
 - merge, 4-13
 - query by example, 4-14
 - with the database adapter, 4-13
- domain
 - passwords, 4-140
- DTD format
 - Native Format Builder wizard support, 7-3

E

- encoding
 - construct, 7-53
- enqueue-specific features
 - AQ adapter, 3-2
- error handling
 - fatalErrorFailoverProcess property, 2-19
 - inbound direction for file and FTP adapters, 2-18
 - outbound direction for file and FTP adapters, 2-37
 - rejectedMessageHandlers property, 2-18
 - uniqueMessageSeparator property, 2-20
- errors
 - default error directory, 2-21
- Esc key
 - stopping creation of XPath expressions, 4-138, 4-139
- existence checking
 - database adapter, 4-90

F

- fatalErrorFailoverProcess property
 - error handling, 2-19
- file adapter
 - ActivationSpec parameters, 2-22
 - architecture, 2-5
 - archiving successfully processed files, 2-11
 - batch processing, 2-11
 - batching multiple inbound messages, 2-15
 - batching multiple outbound files, 2-35
 - binary data format support, 2-2
 - Cobol Copybook format support, 2-2
 - debatching use case, 2-64
 - delimited format support, 2-2
 - dynamic outbound file names, 2-30, 2-34
 - features, 2-2
 - file delimiter use case, 2-64
 - file polling, 2-15
 - file read concepts, 2-7
 - file read operation, 2-3
 - file reading use case, 2-64
 - file size delivery limitations, 2-11
 - file write concepts, 2-25
 - file writing use case, 2-64
 - fixed positional format support, 2-2
 - fixed positional use case, 2-64
 - guaranteed delivery and recovery, 2-21
 - inbound direction, 2-7
 - inbound header WSDL file, 2-23
 - inbound WSDL file, 2-22
 - including and excluding files, 2-13
 - limitations on outbound file name lengths, 2-28
 - logical and physical directory paths, 2-9, 2-27
 - opaque support, 2-2
 - outbound file directory, 2-26
 - outbound file naming conventions, 2-32
 - outbound header WSDL file, 2-38
 - outbound WSDL file, 2-37
 - processing large files, 2-17
 - supported formats, 2-2
 - supported naming patterns, 2-12
 - synchronous reads using an invoke activity, 2-24
 - translating native data, 2-17
 - use cases, 2-63
 - write read operation, 2-3
 - XML format support, 2-2
- file names
 - limitations on file adapter outbound file name lengths, 2-28
- file polling, 2-15
- file reading
 - file adapter use case, 2-64
- file writing
 - file adapter use case, 2-64
- Filter-by Criteria, 6-26
- fixed length data
 - native schema, 7-22
- fixed length structure
 - native schema, 7-10
- fixed positional

- file adapter use case, 2-64
- fixed-length positional format
 - Native Format Builder wizard support, 7-3
- from Oracle BPEL Console, 2-21
- FTP adapter
 - ActivationSpec parameters, 2-22
 - architecture, 2-5
 - archiving successfully processed files, 2-11
 - batch processing, 2-11
 - batching multiple inbound messages, 2-15
 - batching multiple outbound files, 2-35
 - binary data format support, 2-2
 - Cobol Copybook format support, 2-2
 - creating an Oracle Wallet, 2-50
 - delimited format support, 2-2
 - dynamic outbound file names, 2-30, 2-34
 - features, 2-2
 - file inbound file directory, 2-26
 - file modification times, 2-42
 - file polling, 2-15
 - file read concepts, 2-7
 - file read operation, 2-3
 - file write concepts, 2-25
 - file write operation, 2-3
 - fixed positional format support, 2-2
 - guaranteed delivery and recovery, 2-21
 - inbound direction, 2-7, 2-40
 - inbound header WSDL file, 2-23
 - inbound WSDL file, 2-22
 - including and excluding files, 2-13
 - installing and configuring OpenSSL, 2-48
 - installing and configuring vsftpd, 2-49
 - logical and physical directory paths, 2-9, 2-27
 - opaque support, 2-2
 - outbound direction, 2-45
 - outbound file naming conventions, 2-32
 - outbound header WSDL file, 2-38
 - outbound WSDL file, 2-37
 - processing large files, 2-17
 - restrictions on use of RESTART and RECOVERY commands, 2-40
 - secure FTP overview, 2-47
 - serverType property for determining line separations during file transfers, 2-41
 - specifying connection information to an FTP server, 2-39
 - SSL, 2-46
 - supported formats, 2-2
 - supported naming patterns, 2-12
 - translating native data, 2-17
 - use cases, 2-63, 2-65
 - using secure FTP, 2-46
 - XML format support, 2-2
- FTP server
 - specifying configuration information in the oc4j-ra.xml file, 2-40
 - specifying connection information to, 2-39
- FulfillOrder process
 - deploying, 4-140
 - running, 4-140

- function return values
 - database adapter, 4-122

H

- header properties
 - JMS adapter, 5-5
- header variables
 - creating for an adapter, 3-25
 - specifying, 2-30, 2-34
- headerLines
 - construct, 7-53
- headerLinesTerminatedBy
 - construct, 7-53

I

- IBM Websphere JMS configuration, 5-22
- identifierLength
 - construct, 7-54
- importing tables
 - database adapter, 4-28
- inbound distributed polling
 - database adapter, 4-91
- Integration with CICS, 6-29
- invoke activities
 - creating, 4-134
- invoke activity
 - synchronous reads using the file adapter, 2-24
- itemSeparatedBy
 - construct, 7-54

J

- Java pattern letters
 - supported with file and FTP adapters, 2-33
- JDeveloper BPEL Designer
 - adapters, 1-1
- JDeveloper ESB Designer
 - adapters, 1-3
- JDK regular expressions
 - supported with the file and FTP adapters, 2-12, 2-14
- JMS adapter
 - concepts, 5-1
 - generated WSDL files, 5-15
 - header properties, 5-5
 - IBM Websphere JMS configuration, 5-22
 - OC4J JMS configuration, 5-20
 - oc4j-ra.xml file, 5-16
 - OJMS configuration, 5-18
 - point-to-point, 5-4
 - produce message procedure, 5-17
 - publish/subscribe, 5-4
 - Tibco JMS configuration, 5-20

L

- last updated
 - database adapter, 4-20
- last-read ID

- database adapter, 4-19
- leftsurroundedBy
 - construct, 7-54
- length
 - construct, 7-54
- lists
 - native schema, 7-28
- listTerminatedBy
 - construct, 7-54
- LocalBPELServer
 - using to deploy a process, 4-140
- locking
 - database adapter, 4-91
- logical delete
 - database adapter, 4-17
- logical directory paths, 2-9, 2-27
- lookAhead
 - construct, 7-54

M

- maxRaiseSize
 - database adapter, 4-90
- merge
 - DML operations, 4-13
- Message Delivery Failure Options, 6-26
- Message Grouping, 6-27
- message priority
 - AQ adapter, 3-2
- Message Properties
 - Message Expiry, 6-23
 - Message Format, 6-23
 - Message Persistence, 6-24
 - Message Priority, 6-23
 - Messages Types, 6-22
- message recovery, 2-21
- Message Segmentation, 6-26
- messages
 - file size delivery limitations with the file adapter, 2-11
- Messages Type
 - Normal Message, 6-22
 - Reply Message, 6-23
 - Report Message, 6-23
 - Request Message, 6-23
- Messaging Scenarios
 - Dequeue Message, 6-9
 - Request-Response Synchronous (Oracle BPEL Process Manager as Server), 6-16
 - Enqueue Message, 6-6
 - Request-Reply Asynchronous (Oracle BPEL Process Manager as Server), 6-19
 - Request-Response (Oracle BPEL Process Manager as a Client, 6-11
 - Request-Response Synchronous (Oracle Enterprise Service Bus as Server), 6-21
- MQSeries Adapter
 - Use Cases, 6-36
- MQSeries Adapter Configuration, 6-34
 - Adding com.ibm.mq.jar, 6-34

- oc4j-ra.xml, 6-35
- MQSeries Concepts, 6-1, 6-3
 - Application Data, 6-4
 - Cluster, 6-2
 - Enqueue/Dequeue, 6-2
 - Message, 6-1
 - Message Channel, 6-2
 - Message Group, 6-2
 - Message Header, 6-4
 - Message Queue, 6-1
 - Message Segment, 6-2
 - Messaging, 6-1
 - Optional Header, 6-4
 - Queue Manager, 6-2
 - Request/Response, 6-2
 - Transmission Queue, 6-2
- multiconsumer queue
 - AQ adapter, 3-2

N

- namespaces
 - schema must have a namespace, 2-18, 6-9
- naming patterns
 - supported with the file and FTP adapters, 2-12
- Native Format Builder wizard
 - COBOL Copybook format support, 7-3
 - creating native schema files, 7-1
 - delimited format support, 7-3
 - DTD format support, 7-3
 - fixed-length positional format, 7-3
 - overview of windows, 7-7
 - starting, 2-18, 6-9
 - supported formats, 7-2
- Native MQSeries Adapter, 6-4
 - Concepts, 6-6
 - Configuring, 6-34
 - Correlation Schemas, 6-24
 - Distribution List Support, 6-25
 - Filter-by Criteria, 6-26
 - Integration with CICS, 6-29
 - Integration with Oracle BPEL Process Manager, 6-5
 - Integration with Oracle Enterprise Service Bus, 6-5
 - Message Delivery Failure Options, 6-26
 - Message Grouping, 6-27
 - Message Properties, 6-22
 - Message Segmentation, 6-26
 - Messaging Scenarios, 6-6
 - Need, 6-4
 - Report Messages, 6-25
- native schema files
 - creating, 7-1
- native schemas
 - arrayIdentifierLength construct, 7-53
 - arrayLength construct, 7-53
 - arrays, 7-30
 - arrayTerminatedBy construct, 7-54
 - assign construct, 7-54

- byteOrder construct, 7-53
- cellSeparatedBy construct, 7-54
- choiceCondition construct, 7-54
- complex structure, 7-11
- conditional processing, 7-36
- conditionValue construct, 7-54
- constructs, 7-53
- CSV, 7-8
- dateFormat construct, 7-54
- dates, 7-43
- encoding construct, 7-53
- fixed length data, 7-22
- fixed length structure, 7-10
- headerLines construct, 7-53
- headerLinesTerminatedBy construct, 7-53
- identifierLength construct, 7-54
- itemSeparatedBy construct, 7-54
- leftsurroundedBy construct, 7-54
- length construct, 7-54
- lists, 7-28
- listTerminatedBy construct, 7-54
- lookAheadconstruct, 7-54
- paddedBy construct, 7-54
- padStyle construct, 7-54
- quotedBy construct, 7-54
- rightsurroundedBy construct, 7-54
- separated value file structure, 7-10
- skip construct, 7-54
- skipLines construct, 7-54
- skipUntil construct, 7-54
- standalone construct, 7-53
- startsWith construct, 7-54
- stream construct, 7-53
- style construct, 7-54
- surrounded data, 7-27
- surroundedBy construct, 7-54
- terminated data, 7-25
- terminatedBy construct, 7-54
- understanding, 7-8
- uniqueMessageSeparator construct, 7-53
- use cases, 7-8
- variable construct, 7-54
- variables, 7-45
- variables construct, 7-54
- version construct, 7-53
- xmlversion construct, 7-53
- null values
 - database adapter, 4-121

O

- object models
 - database adapter, 4-32
- object type inheritance, 4-116
- OC4J JMS configuration, 5-20
- oc4j-ra.xml, 6-35
 - channelName, 6-36
 - clientEncoding, 6-36
 - hostName, 6-35
 - hostOSType, 6-36

- password, 6-36
- portNumber, 6-35
- queueManagerName, 6-36
- sample, 6-36
- userID, 6-36
- oc4j-ra.xml file
 - configuring a database connection in, 4-7, 4-26
 - configuring an FTP server connection in, 2-40
 - JMS connection factory definitions, 5-16
 - location of, 2-40
- OJMS configuration, 5-18
- opaque format
 - supported with file and FTP adapters, 2-2
- OpenSSL
 - installing and configuring, 2-48
- Oracle Applications adapter, 1-2, 1-4
- Oracle BPEL Console
 - accessing, 4-140
 - manually performing message recovery, 2-21
 - running a process, 4-140
- Oracle Wallet
 - installing and configuring, 2-50
- OracleAS Adapter for AQ
 - See* AQ adapter, 3-1
- OracleAS Adapter for Files
 - See* file adapter, 2-1
- OracleAS Adapter for FTP
 - See* FTP adapter, 2-1
- OracleAS Adapter for JMS
 - See* JMS adapter, 5-1
- overloading, 4-105

P

- paddedBy
 - construct, 7-54
- padStyle
 - construct, 7-54
- passwords
 - domain, 4-140
- payload schema
 - AQ adapter, 3-5
- payloads
 - large, 4-90
- performance
 - database adapter, 4-88
- physical delete
 - database adapter, 4-16
- point-to-point
 - JMS adapter, 5-4
- polling strategies
 - database adapter, 4-15
- polling strategy
 - database adapter, 4-90
- primitive datatypes
 - supported, 4-113
- processing large files
 - with the file adapter, 2-17
- produce message procedure
 - JMS adapter, 5-17

- publish/subscribe
 - JMS adapter, 5-4

Q

- query by example
 - DML operations, 4-14
- quotedBy
 - construct, 7-54

R

- recovery
 - of messages from Oracle BPEL Console, 2-21
- REF CURSOR support
 - database adapter, 4-122
- regex constructs
 - supported, 2-12, 2-14
- rejectedMessageHandlers property
 - error handling, 2-18
- relational-to-XML mappings, 4-71
 - for database adapter, 4-9
- relationship reading
 - database adapter, 4-90
- Report Messages, 6-25
 - Confirmation on Arrival, 6-25
 - Confirmation on Delivery, 6-25
 - Exception Report, 6-25
 - Expiry Report, 6-25
 - Negative Action Notification, 6-26
 - Positive Action Notification, 6-25
- Response Fallback Queue Name, 6-18
- rightsSurroundedBy
 - construct, 7-54
- running
 - FulfillOrder, 4-140

S

- schemas
 - must have a namespace, 2-18, 6-9
- secure FTP
 - using with the FTP adapter, 2-46
- separated value file structure
 - native schema, 7-10
- sequencing table
 - database adapter, 4-19, 4-20
- sequencing table updates
 - database adapter, 4-36
- serverType property
 - determines line separations during file transfers, 2-41
- server.xml file
 - location of, 2-17
 - processing large files, 2-17
- service names
 - creating, 2-6, 6-5
 - in adapters, 1-2, 1-4
- skip
 - construct, 7-54
- skipLines

- construct, 7-54
- skipUntil
 - construct, 7-54
- SSL
 - creating an Oracle Wallet, 2-50
 - installing and configuring OpenSSL, 2-48
 - installing and configuring vsftpd, 2-49
 - secure FTP overview, 2-47
 - using secure FTP with the FTP adapter, 2-46
- standalone
 - construct, 7-53
- startsWith
 - construct, 7-54
- stored procedures and functions
 - creating a stored procedure use case, 4-127
 - database adapter, 4-100
 - design time WSDL and XSD generation, 4-111
 - invocation at run time, 4-121
 - overloaded procedures, 4-105
 - run time, 4-118
 - searching for, 4-102
- stream
 - construct, 7-53
- style
 - construct, 7-54
- surrounded data
 - native schema, 7-27
- surroundedBy
 - construct, 7-54
- synchronous reads
 - using the file adapter in an invoke activity, 2-24

T

- terminated data
 - native schema, 7-25
- terminatedBy
 - construct, 7-54
- Tibco JMS configuration, 5-20
- TopLink Mapping Workbench project, 4-67
- toplink_mappings.xml file, 4-8, 4-38, 4-67, 4-71
- translation
 - native data, 2-36
 - not required, 2-18, 6-9
 - of native data, 2-17
 - requires native schemas, 7-1
- translator
 - converts native data to XML and back, 2-2
- troubleshooting and workarounds
 - AQ adapter, A-25
 - database adapter, A-1
 - file adapter, A-23
 - FTP adapter, A-23
 - using stored procedures with the database adapter, A-19

U

- uniqueMessageSeparator property
 - error handling, 2-20

- uniqueMessageSeparator
 - construct, 7-53
- updating a field in a table
 - database adapter, 4-35
- use cases
 - creating a stored procedure, 4-127
 - using the AQ adapter, 3-6
 - using the File and FTP adapters, 2-63
 - using the Native Format Builder, 7-8
- user-defined types, 4-114, 4-116
- using the Native Format Translator for removing or adding namespaces to XML with no namespace, 7-14

V

- validating
 - FulfillOrder process, 4-140
- value binding
 - database adapter, 4-118
- variable
 - construct, 7-54
- variables
 - automatically creating, 4-136
 - construct, 7-54
 - native schema, 7-45
- version
 - construct, 7-53
- vsftpd server
 - installing and configuring, 2-49

W

- Web services
 - SQL operations as, 4-13
- WHERE clause
 - database adapter, 4-32
- WSDL file
 - defining the adapter service, 4-73
 - inbound direction for file and FTP adapters, 2-22
 - outbound direction for file and FTP adapters, 2-37
- WSDL files
 - JMS adapter, 5-15
 - specifying logical directory paths, 2-9, 2-27

X

- XML schema
 - generated by Adapter Configuration wizard, 4-78
- xmlversion
 - construct, 7-53
- XPath expressions
 - creating, 4-138, 4-139
- XSD attributes, 4-113