Oracle® Application Server

Adapter for Oracle Applications User's Guide

10*g* (10.1.3.4.0)

Part No. E10536-01

June 2008

ORACLE®

Oracle Application Server Adapter for Oracle Applications User's Guide, 10*g* (10.1.3.4.0)

Part No. E10536-01

Copyright © 2005, 2008, Oracle and/or its affiliates. All rights reserved.

Primary Author: David Weld, Melody Yang

Contributing Author: Santiago Bastidas, Neeraj Chauhan, Vimmika Dinesh, Neeraj Kumar, Narendra Parihar, Nagaraj Ravinuthala, Nadakuditi Ravindra,Veshaal Singh, Sheelah Vasudevan

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

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

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

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

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

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

This software and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third party content, products and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third party content, products or services.

Contents

Send Us Your Comments

Preface

1 Introduction to OracleAS Adapter for Oracle Applications

Overview of OracleAS Adapter for Oracle Applications.....1-1
New Features in This Release.....1-10
Understanding Applications Context.....1-11
Supporting for Multiple Organization Setups.....1-17
Supporting for Multiple Languages.....1-20
Understanding OracleAS Adapter for Oracle Applications Security.....1-24
Installing OracleAS Adapter for Oracle Applications.....1-30
Using the Oracle Applications Module Browser.....1-31
General Issues and Workarounds.....1-32

2 Using XML Gateway

Overview of XML Gateway.....2-1
Design-Time Tasks for XML Gateway Inbound Messaging.....2-6
Creating a New BPEL Project.....2-7
Creating a Partner Link.....2-10
Configuring the Invoke Activity.....2-21
Adding a Partner Link for the File Adapter.....2-27
Configuring the Receive Activity.....2-33
Configuring the Assign Activity.....2-36
Run-Time Tasks for XML Gateway Inbound Messaging.....2-39
Deploying the BPEL Process.....2-39

Testing the BPEL Process 2-41
Verifying Records in Oracle Applications 2-44
Design-Time Task for XML Gateway Outbound Messaging 2-45
Configuring XML Gateway for Outbound Messaging 2-46
Run-Time Task for XML Gateway Outbound Messaging 2-47
Testing the BPEL Process 2-47
Troubleshooting and Debugging 2-48

3 Using Business Events

Overview of Business Events 3-1
Business Events Concepts 3-3
Design-Time Tasks for Outbound Business Events 3-4
Creating a New BPEL Project 3-5
Creating a Partner Link 3-7
Configuring the Receive Activity 3-22
Adding a Partner Link for the File Adapter 3-25
Configuring the Invoke Activity 3-30
Configuring the Assign Activity 3-33
Run-Time Tasks for Outbound Business Events 3-36
Deploying the BPEL Process 3-36
Testing the BPEL Process 3-38
Troubleshooting and Debugging 3-52

4 Using Concurrent Programs

Overview of Concurrent Programs 4-1
Design-Time Tasks for Concurrent Programs 4-1
Creating a New BPEL Project 4-2
Adding a Partner Link 4-5
Configuring the Invoke Activity 4-15
Configuring the Transform Activity 4-23
Run-Time Tasks for Concurrent Programs 4-27
Deploying the BPEL Process 4-28
Testing the BPEL Process 4-29
Verifying Records in Oracle Applications 4-31

5 Using Tables and Views

Overview of Tables and Views 5-1
Design-Time Tasks for Interface Tables 5-3
Creating a New BPEL Project 5-3
Adding a Partner Link 5-6

Configuring the Invoke Activity....5-19
Configuring the Assign Activity....5-22
Run-Time Tasks for Interface Tables....5-23
Deploying the BPEL Process....5-23
Testing the BPEL Process....5-25
Design-Time Tasks for Views....5-27
Creating a New BPEL Project....5-28
Adding a Partner Link....5-31
Configuring the Invoke Activity....5-45
Configuring the Assign Activity....5-48
Run-Time Tasks for Views....5-49
Deploying the BPEL Process....5-49
Testing the BPEL Process....5-51
Troubleshooting and Debugging....5-53

6 Using PL/SQL APIs

Overview of PL/SQL APIs....6-1
Design-Time Tasks for PL/SQL APIs....6-2
Creating a New BPEL Project....6-2
Adding a Partner Link....6-5
Defining Wrapper APIs....6-17
Declaring Parameters with a DEFAULT Clause....6-20
Configuring the Invoke Activity....6-23
Configuring the Transform Activity....6-30
Run-Time Tasks for PL/SQL APIs....6-33
Deploying the BPEL Process....6-33
Testing the BPEL Process....6-35
Troubleshooting and Debugging....6-37

7 Using e-Commerce Gateway

Overview of e-Commerce Gateway Integration....7-1
Design-Time Tasks for e-Commerce Gateway....7-2
Creating a New BPEL Project....7-3
Adding a Partner Link....7-6
Configuring the Invoke Activity....7-16
Configuring the Transform Activity....7-24
Run-Time Tasks for e-Commerce Gateway....7-28
Deploying the BPEL Process....7-28
Testing the BPEL Process....7-30
Verifying Records in Oracle Applications....7-32

A Modifying Connection and Definition Files

WSDL Definition File A-1
Configuring Connection Information A-2
Oracle E-Business Suite Connectivity Using J2EE Data Source A-5

Index

Send Us Your Comments

Oracle Application Server Adapter for Oracle Applications User's Guide, 10*g* (10.1.3.4.0)

Part No. E10536-01

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document. Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

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

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

Note: Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the new Applications Release Online Documentation CD available on Oracle MetaLink and www.oracle.com. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: appsdoc_us@oracle.com

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

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

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

Preface

Intended Audience

Welcome to the *Oracle Application Server Adapter for Oracle Applications User's Guide*, 10*g* (10.1.3.4.0)

This documentation is written for the technical consultants, implementers and system integration consultants who use OracleAS Adapter for Oracle Applications.

This guide assumes you have a working knowledge of the following:

- The principles and customary practices of your business area.
- Oracle E-Business Suite.
- Oracle BPEL Process Manager.
- Oracle JDeveloper.
- Oracle Database, Oracle Application Server, and PL/SQL technology.
- EBS Integration Interfaces.
- Oracle integration technologies, including Web services, WSDL, XML Gateway, EDI Gateway, and the Business Event System.
- B2B, A2A and BP integrations.

If you have never used these products, Oracle suggests that you attend training classes available through Oracle University.

See Related Information Sources on page xi for more Oracle Applications product information.

TTY Relay Access to Oracle Support Services

To reach AT&T Customer Assistants, dial 711 or 1.800.855.2880. An AT&T Customer Assistant will relay information between the customer and Oracle Support Services at 1.800.223.1711. Complete instructions for using the AT&T relay services are available at http://www.consumer.att.com/relay/tty/standard2.html. After the AT&T Customer Assistant contacts Oracle Support Services, an Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process.

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Structure

1 Introduction to OracleAS Adapter for Oracle Applications
2 Using XML Gateway
3 Using Business Events
4 Using Concurrent Programs
5 Using Tables and Views
6 Using PL/SQL APIs
7 Using e-Commerce Gateway
A Modifying Connection and Definition Files

Related Information Sources

You can choose from many sources of information, including online documentation, training, and support services, to increase your knowledge and understanding of OracleAS Adapter for Oracle Applications.

Documentation

You may want to refer to other Oracle Application Server guides when you set up and use OracleAS Adapter for Oracle Applications. You can read the guides online by reading from the Oracle Application Server or Oracle Database Documentation Library CD included in your media pack, or on the Oracle Technology Network (OTN) [http://www.oracle.com/technology/documentation/appserver.html]. If you require printed guides, you can purchase them from the Oracle Store [http://oraclestore.oracle.com].

To download free release notes, installation documentation, white papers, or other collateral, please visit the main OTN page [http://www.oracle.com/technology/]. You must register online before using OTN; registration is free and can be done at the OTN registration page [http://www.oracle.com/technology/membership/].

Training

Oracle offers a complete set of training courses to help you and your staff master Oracle Application Server 10*g* and reach full productivity quickly. These courses are organized into functional learning paths, so you take only those courses appropriate to your job or area of responsibility.

You have a choice of educational environments. You can attend courses offered by Oracle University at any one of our many Education Centers, you can arrange for our trainers to teach at your facility, or you can use Oracle Learning Network (OLN), Oracle University's online education utility.

> **Tip:** For information about upcoming instructor-led training, please refer to Oracle University's course offerings [http://education.oracle.com/].

In addition, Oracle training professionals can tailor standard courses or develop custom courses to meet your needs. For example, you may want to use your organization's structure, terminology, and data as examples in a customized training session delivered at your own facility.

Support

From on-site support to central support, our team of experienced professionals provides the help and information you need to keep Oracle Application Server 10*g* working for you. This team includes your Technical Representative, Account Manager, and Oracle's large staff of consultants and support specialists, with expertise in your business area, managing an Oracle Database, and your hardware and software environment.

Do Not Use Database Tools to Modify Oracle Applications Data

Oracle STRONGLY RECOMMENDS that you never use SQL*Plus, Oracle Data Browser, database triggers, or any other tool to modify Oracle Applications data unless otherwise instructed.

Oracle provides powerful tools you can use to create, store, change, retrieve, and maintain information in an Oracle database. But if you use Oracle tools such as SQL*Plus to modify Oracle Applications data, you risk destroying the integrity of your data and you lose the ability to audit changes to your data.

Because Oracle Applications tables are interrelated, any change you make using an Oracle Applications form can update many tables at once. But when you modify Oracle Applications data using anything other than Oracle Applications, you may change a row in one table without making corresponding changes in related tables. If your tables get out of synchronization with each other, you risk retrieving erroneous information and you risk unpredictable results throughout Oracle Applications.

When you use Oracle Applications to modify your data, Oracle Applications automatically checks that your changes are valid. Oracle Applications also keeps track of who changes information. If you enter information into database tables using database tools, you may store invalid information. You also lose the ability to track who has changed your information because SQL*Plus and other database tools do not keep a record of changes.

1

Introduction to OracleAS Adapter for Oracle Applications

This chapter covers the following topics:

- Overview of OracleAS Adapter for Oracle Applications
- New Features in This Release
- Understanding Applications Context
- Understanding OracleAS Adapter for Oracle Applications Security
- Installing OracleAS Adapter for Oracle Applications
- Using the Oracle Applications Module Browser
- General Issues and Workarounds

Overview of OracleAS Adapter for Oracle Applications

Oracle Applications is a set of integrated business applications that runs entirely on the Internet. Oracle Applications offers you the following:

- Reduced costs
- Increased revenue across front-office and back-office functions
- Access to current, accurate, and consistent data

The applications in Oracle Applications are built on a unified information architecture that consolidates data from Oracle and non-Oracle applications and enables a consistent definition of customers, suppliers, partners, and employees across the entire enterprise. This results in a suite of applications that can give you information, such as current performance metrics, financial ratios, profit and loss summaries. To connect Oracle Applications to non-Oracle applications, you use OracleAS Adapter for Oracle Applications.

OracleAS Adapter for Oracle Applications provides comprehensive, bidirectional, multimodal, synchronous, and asynchronous connectivity to Oracle Applications. The Adapter supports for all modules of Oracle Applications in Release 12 and Release 11*i* including selecting custom integration interface types based on the version of Oracle E-Business Suite.

> **Important:** Please note that the support for various versions of Oracle E-Business Suite has the following conditions:
>
> - OracleAS Adapter for Oracle Applications supports only those versions of Oracle E-Business Suite Release 11*i* which work with OWF.G.Rollup 7 applied.
> - OracleAS Adapter for Oracle Applications version 10.1.3.3 supports Oracle E-Business Suite Release 12.0.
> - To enable the "Native E-Business Suite Connectivity using J2EE Data Sources" feature, the minimum requirement for Oracle E-Business Suite Release 11*i* is FND Rollup 6 and for Oracle E-Business Suite Release 12 is 12.0.4 release.
>
> See "Oracle Application Sever Adapter for Oracle Applications Documentation Update, Release 10*g*" Oracle*MetaLink* Document 464164.1 for details.

Features

OracleAS Adapter for Oracle Applications includes the following features:

- It supports open standards, including J2EE Connector Architecture (J2CA), Extensible Markup Language (XML), Web Service Invocation Framework (WSIF), Web Service Inspection Language (WSIL), and Web Service Definition Language (WSDL).
- It uses a JDeveloper based design-time tool for dynamically browsing the Oracle Applications interface and configuring the adapter metadata.
- It integrates applications based on open standards, such as IFX, OAG, RosettaNet, and UCCnet by interfacing with XML Gateway.
- It generates adapter metadata as WSDL files with J2CA extension.

  > **Note:** See *Oracle Application Server Adapter Concepts* on OTN for more information.

- It supports multiple languages and multiple organization setups based on the

concept of applications context.

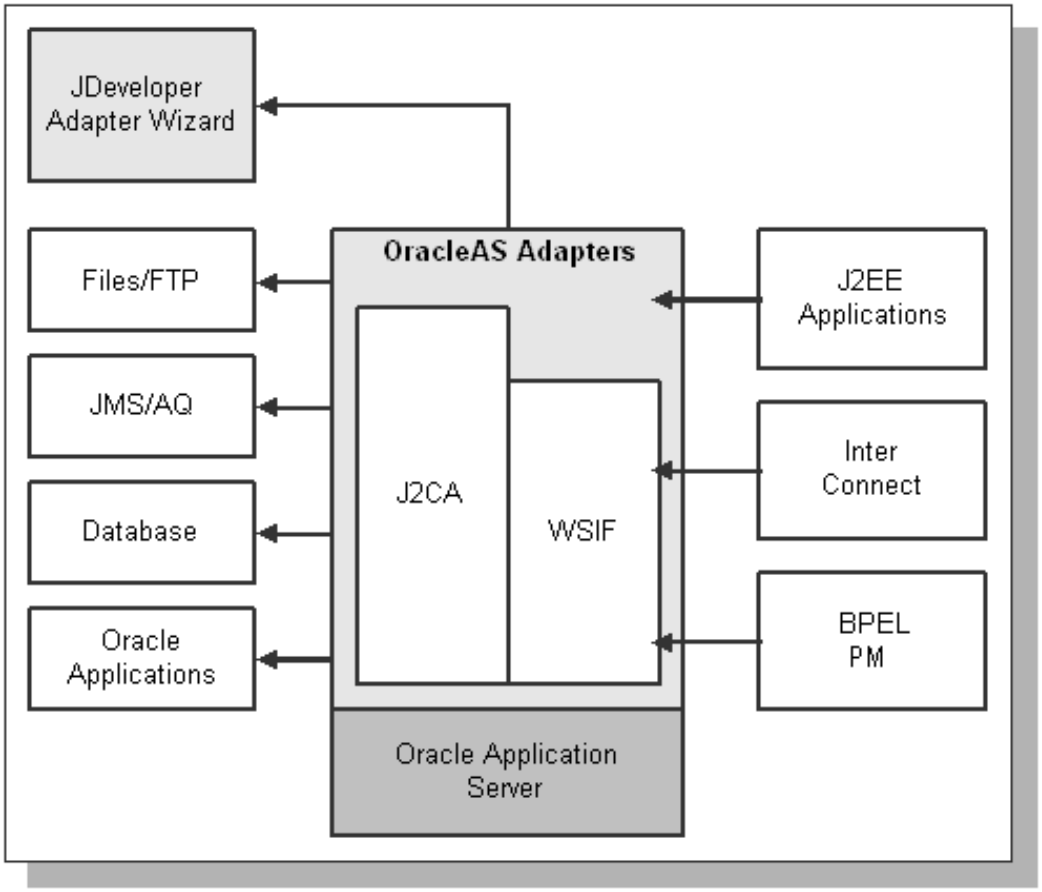
- It leverages and supports Oracle User Management function security to allow authorized users to access and execute APIs that they are exposed through the BPEL process to update Oracle Applications.

> **Note:** This overview includes details about features and capabilities that are new in the current release of OracleAS Adapter for Oracle Applications. For more information, see New Features in This Release, page 1-10.

Architecture

OracleAS Adapter for Oracle Applications is based on J2CA 1.0 standards and deployed as a resource adapter in the same Oracle Application Server Containers for J2EE (OC4J) container as BPEL Process Manager. The architecture of OracleAS Adapter for Oracle Applications is similar to the architecture of technology adapters.

OracleAS Adapter for Oracle Applications Architecture



Integration Interface Types

OracleAS Adapter for Oracle Applications acts as a highly flexible integration interface for Oracle Applications. The adapter supports the following interface types for integrating with Oracle Applications:

- **Oracle XML Gateway**

  XML Gateway enables bidirectional integration with Oracle Applications. It helps you to insert and retrieve data from Oracle Applications. XML Gateway is a higher-level interface that exposes OAGIS-formatted XML documents for commonly used Oracle Application business objects and business interfaces. XML Gateway integrates with interface tables, *Oracle Workflow Business Event System* (BES), and interface views to insert and retrieve data from Oracle Applications. It maps the underlying table data to XML and back.

- **Business events**

  A business event is an occurrence in an internet application that might be

significant to other objects in a system or to external agents. An example of a business event can be the creation of a new sales order or changes to an existing order.

Oracle Workflow uses the Business Event System that leverages the Oracle Advanced Queuing (AQ) infrastructure to communicate and manage business events between systems. The Business Event System consists of an Event Manager and workflow process event activities. The Event Manager lets you register subscriptions to significant events; event activities representing business events within workflow processes let you model complex business flows or logics within workflow processes.

When a local event occurs, the subscribing code is executed in the same transaction as the code that raised the event. Subscription processing can include executing custom code on the event information, sending event information to a workflow process, and sending event information to other queues or systems.

- **Concurrent programs**

  Concurrent programs enable you to move data from interface tables to base tables.

- **Interface tables**

  Interface tables enable you to insert or update data into Oracle Applications. The associated concurrent program should be running to move the data from the interface tables to base tables.

- **Interface views**

  Interface views help you to retrieve data from Oracle Applications using the application tables.

- **PL/SQL APIs**

  These APIs enable you to insert and update data in Oracle Applications using PL/SQL.

- **Oracle e-Commerce (EDI) Gateway**

  Oracle e-Commerce Gateway provides a common, standards-based approach for Electronic Data Interchange (EDI) integration between Oracle Applications and third party applications.

Please note that OracleAS Adapter for Oracle Applications also supports the following integration interface types that are exposed by the Oracle Applications Module Browser, not by Oracle Integration Repository:

- Customized XML Gateway maps
- Customized PL/SQL APIs

Note: Business events integration interface type is also exposed by Oracle Applications Module Browser, not by Oracle Integration Repository.

Support for Oracle Integration Repository

Oracle Integration Repository, an integral part of Oracle E-Business Suite, is a prebuilt catalog of information about the numerous public integration interfaces delivered with Oracle applications, known as *business interfaces*. It provides a comprehensive view of the interface mechanisms available for Oracle E-Business Suite's business interfaces. These interfaces are exposed because their definitions were annotated at design time as required by Oracle Integration Repository.

Oracle Integration Repository can only provide information about an integration interface that has been specifically annotated by the developer to make it public. OracleAS Adapter for Oracle Applications takes advantage of the annotations that have already been created to make the following business interface types visible in the Oracle Applications Module Browser:

- XML Gateway message maps
- PL/SQL APIs
- Concurrent programs
- Open Interface tables
- Interface views
- e-Commerce Gateway EDI messages

These business interfaces are exposed as Web services, and are available for process orchestration through the Oracle BPEL Process Manager.

For more information about Oracle Integration Repository, see *Oracle Integration Repository User's Guide*. These guide is part of the Oracle Applications documentation library. Oracle Applications documentation can be accessed with the following link:

http://www.oracle.com/technology/documentation/applications.html

Support for Custom Integration Interfaces in Various Versions of Oracle E-Business Suite

To extend the support for Oracle E-Business Suite, OracleAS Adapter for Oracle Applications enhances the capability of implicitly utilizing Oracle Integration Repository to support custom interfaces, such as customized PL/SQL APIs, with reference to the following versions of Oracle E-Business Suite:

- Release 12, page 1-7
- Release 11*i*10, page 1-8
- Pre-Release 11*i*10, page 1-9

Important: Please note that the support for various versions of Oracle E-Business Suite has the following conditions:

- OracleAS Adapter for Oracle Applications supports only those versions of Oracle E-Business Suite Release 11*i* which work with OWF.G.Rollup 7 applied.
- OracleAS Adapter for Oracle Applications version 10.1.3.3 supports Oracle E-Business Suite Release 12.0.
- To enable the "Native E-Business Suite Connectivity using J2EE Data Sources" feature, the minimum requirement for Oracle E-Business Suite Release 11*i* is FND Rollup 6 and for Oracle E-Business Suite Release 12 is 12.0.4 release.

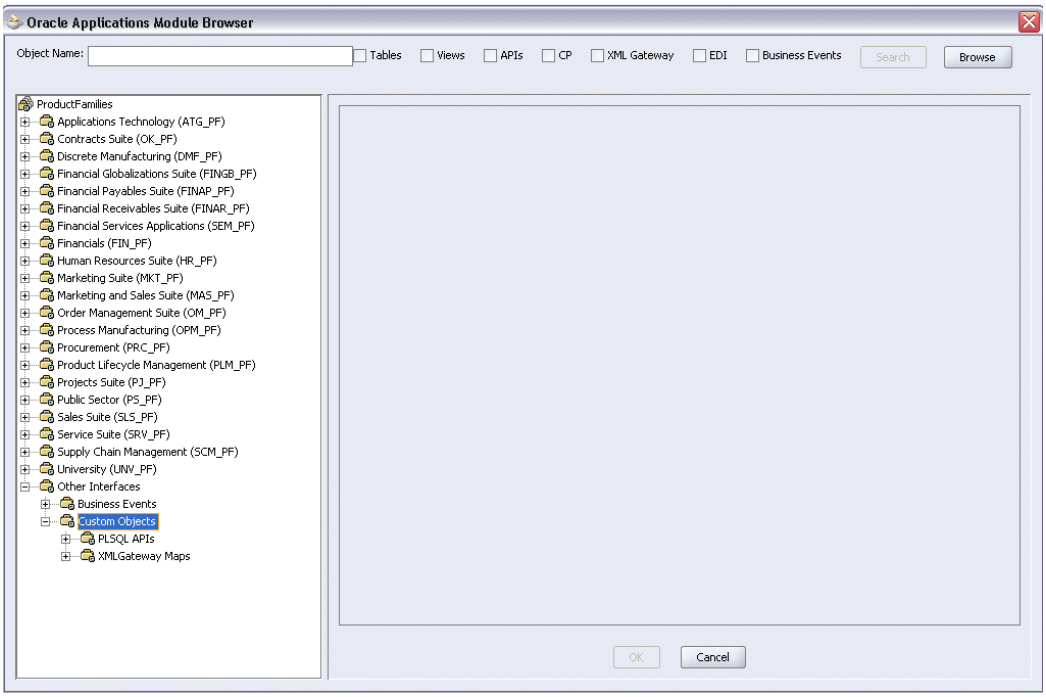
  See "Oracle Application Sever Adapter for Oracle Applications Documentation Update, Release 10*g*" Oracle*MetaLink* Document 464164.1 for details.

From the business service creation and runtime perspectives, OracleAS Adapter for Oracle Applications treats customized PL/SQL APIs as part of the seeded PL/SQL APIs. The only difference is that the customized PL/SQL APIs can be exposed by the Oracle Applications Module Browser during the design time.

Support for Oracle E-Business Suite Release 12

From Release 12, Oracle Integration Repository is shipped as part of the E-Business Suite which enables OracleAS Adapter for Oracle Applications to directly connect to the live database of Oracle Integration Repository querying for the public interfaces and then displaying the list of customized PL/SQL APIs under the `Other Interfaces` node in the Oracle Applications Module Browser.

Supporting Custom Integration Interfaces in Release 12



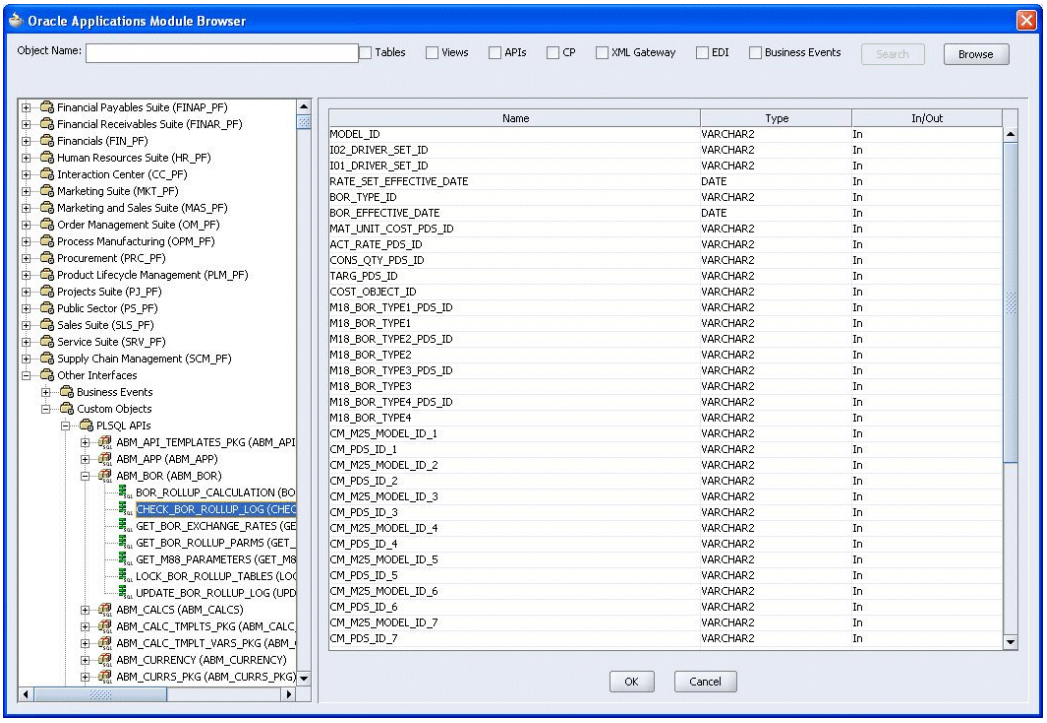
Please note that OracleAS Adapter for Oracle Applications allows you to extract the Integration Repository data file from the live database you connect to Oracle Applications and create a local copy of the Integration Repository in your workplace. Next time when you look for public interfaces, the system can retrieve data from the cache backend connection in your workplace.

For detailed information about connecting to Oracle E-Business Suite Release 12, please refer to the Creating a Partner Link or Adding a Partner Link design-time task for each integration interface.

Support for Oracle E-Business Suite Release 11*i*10

To support the Release 11*i*10 version of Oracle E-Business Suite, OracleAS Adapter for Oracle Applications connects to the live database of Oracle Integration Repository and stores the data file within the Adapter for query. At the design time, OracleAS Adapter for Oracle Applications queries public interfaces from the native XML data file of the Integration Repository located in the Adapter and displays the list of custom integration interfaces under the `Other Interfaces` node in the Oracle Applications Module Browser.

Supporting Custom Integration Interfaces in Release 11i10



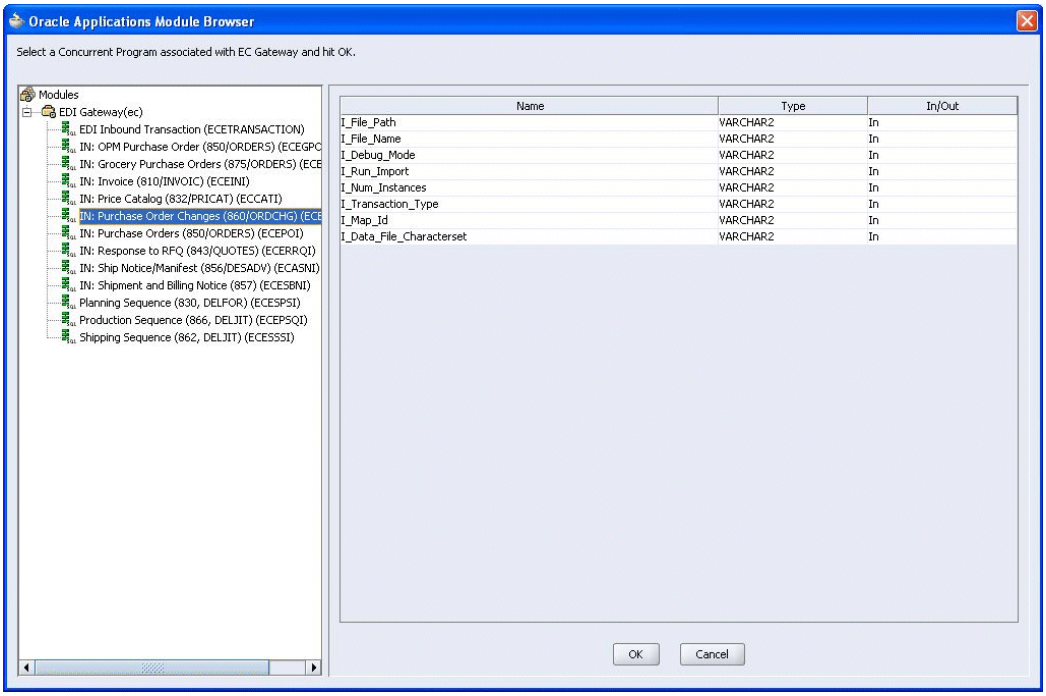
Support for Oracle E-Business Suite Pre-Release 11*i*10

To support the pre-Release 11*i*10 versions of Oracle E-Business Suite, OracleAS Adapter for Oracle Applications connects to the live database of Oracle Integration Repository for all interface types. Since there is no differentiation between public, private, and customized PL/SQL APIs in the pre-Release 11*i*10 versions of Oracle E-Business Suite, OracleAS Adapter for Oracle Applications displays them all under the node of each module through Oracle Applications Module Browser.

Before making a selection from the browser for the pre-Release 11*i*10, you must select an interface type you want to use in the Adapter Configuration Wizard. All interfaces of your selected type will be displayed in the browser.

For example, you will find a list of concurrent programs associated with e-Commerce (EDI) Gateway displayed in the Oracle Application Module Browser as follows if the EDI Gateway interface type is selected.

Supporting Custom Integration Interfaces in pre-Release 11i10



When you make a selection through the module browser at design time, OracleAS Adapter for Oracle Applications validates your selected API against the database. If it exists in the database for a particular version of your instance, then the associated WSDL file will be generated successfully.

New Features in This Release

This section describes the new features that have been added in OracleAS Adapter for Oracle Applications 10*g* (10.1.3.4.0).

Support for Function Security

Security is the most critical feature that is designed to guard application content from unauthorized access. By leveraging Oracle User Management function security, OracleAS Adapter for Oracle Applications now provides a security feature which only allows users with authorized privileges to execute APIs that they are exposed through the BPEL process to update Oracle Applications. This feature protects application programming interfaces (APIs) from unauthorized access or execution without security checks.

Please note that OracleAS Adapter for Oracle Applications provides this security support as an optional feature. If you want all login users to access and execute APIs without security checks, you can turn the security feature off using the "EBS Adapter for BPEL, Function Security Enabled" (EBS_ADAPTER_FUNCTION_SEC_ENABLED)

profile option. If the function security feature is enabled, all API calls for PL/SQL APIs, Oracle e-Commerce Gateway, and concurrent programs will be checked for user security before they are invoked.

Native E-Business Suite Connectivity Using J2EE Data Sources

OracleAS Adapter for Oracle Applications now uses a new mechanism to authenticate users at run time and get the connection to Oracle Applications databases through the use of J2EE data sources. Since J2EE data sources are defined in OC4J container that runs the BPEL processes, this approach is native to Oracle E-Business Suite in defining the connection pool to access the application database.

With this new mechanism, account details information including application login user name and password that was required as part of the configuration for database connection is now added together with the dbc file location as input parameters during the J2EE data source creation.

> **Note:** To have this feature available, you must apply necessary patches to enable the connectivity between Oracle E-Business Suite and an external Oracle Application Server at run time for Oracle E-Business Suite Release 12 and Release 11*i*10. See "Oracle Application Sever Adapter for Oracle Applications Documentation Update, Release 10*g*" Oracle*MetaLink* Document 464164.1 for details.

Understanding Applications Context

Applications context is required for secured transaction processing into and out of Oracle Applications.

Applications context is a combination of **Organization ID**, **Username** and **Responsibility**. To establish applications context, the Organization ID is implicitly derived from the Oracle Applications setup data.

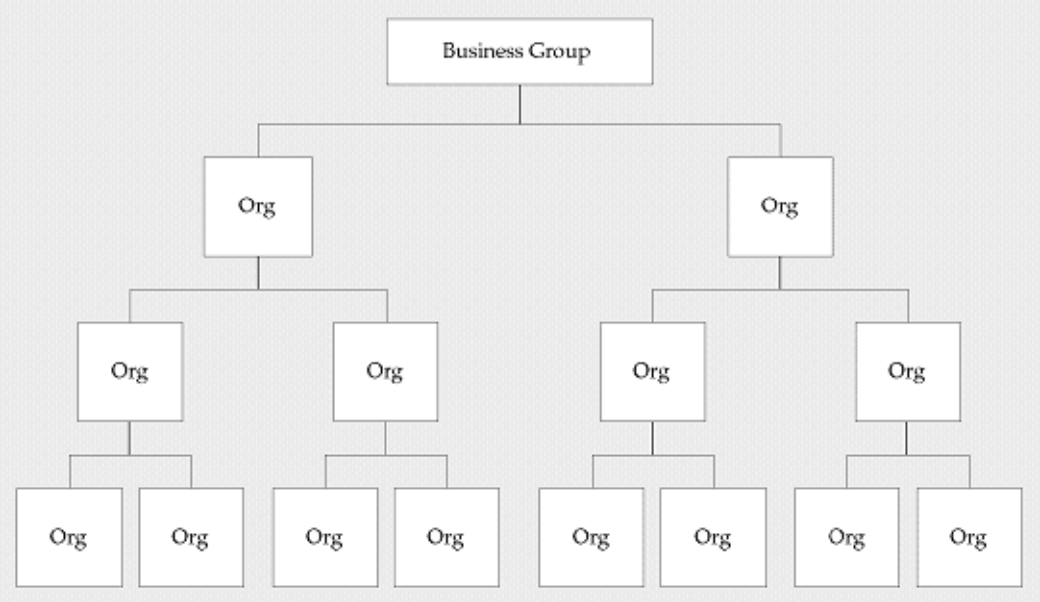
To understand applications context, you need to understand first how Organization ID and multiple organizations are related.

Application Context in Multiple Organizations

Multiple organizations can be sets of books, business groups, legal entities, operating units, or inventory organizations. You can define multiple organizations and the relationships between them in a single installation of Oracle Applications.

Multilevel organization hierarchies can be defined with a business group at the top of each hierarchy. When you define new organizations, they are automatically assigned to the business group associated with your current session. Each organization is part of a business group. The business group is usually the top box on an enterprise organization chart.

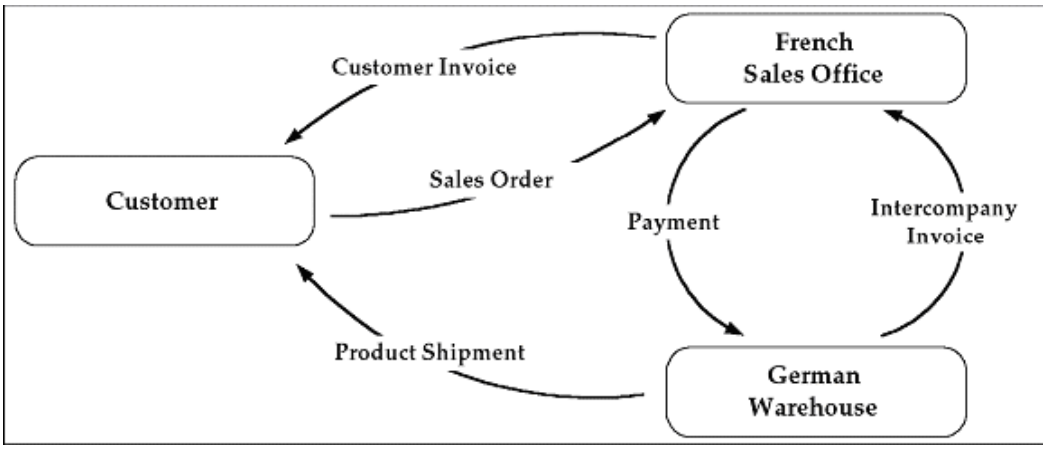
Business Group Hierarchy



Example of a Multiple-Organization Setup

Using the accounting, distribution, and materials management functions in Oracle Applications, you define the relationships among inventory organizations, operating units, legal entities, and sets of books to create a multilevel company structure, as shown in the following diagram.

A Multiple-Organization Transaction

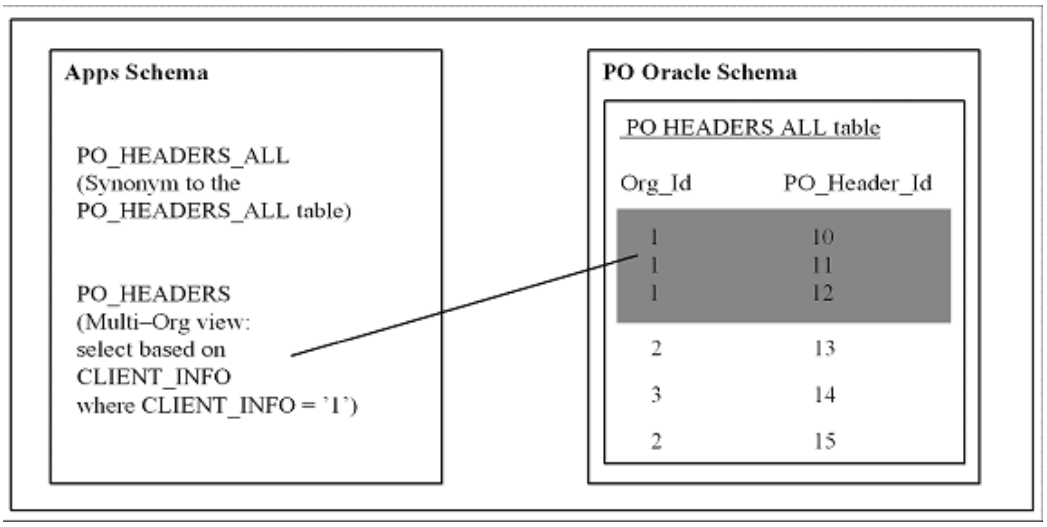


Consider two different organizations in your company: One is a French sales office and the other is a German warehouse. There is a sales order transaction with the customer, and this illustrates how the entire Order-to-Deliver process would work:

1. The customer places a sales order with the French sales office.
2. The German warehouse delivers the product shipment to the customer.
3. The German warehouse issues an inter-company invoice to the French sales office.
4. The French sales office makes the inter-company payment to the German warehouse.
5. The French sales office sends the customer invoice to the customer.
6. The customer makes payment to the French sales office.

The database architecture is the same for a multiple-organization and non-multiple-organization installation, and uses the standard install tools feature that automatically creates synonyms in the APPS schema for each base product table and defines these synonyms with the same name as the base product tables. For example, the PO Oracle schema has a table named PO_HEADERS_ALL and the APPS schema has a corresponding synonym of the same name, PO_HEADERS_ALL. The PO_HEADERS_ALL synonym can be used to access unpartitioned data.

Schema Synonyms



Multi-Organization Access Control (MOAC) Security by Operating Units

While setting up the system profile values, the username and responsibility are tied up with the organization or operating units.

Multiple-Organization System Profiles

System Profile Values

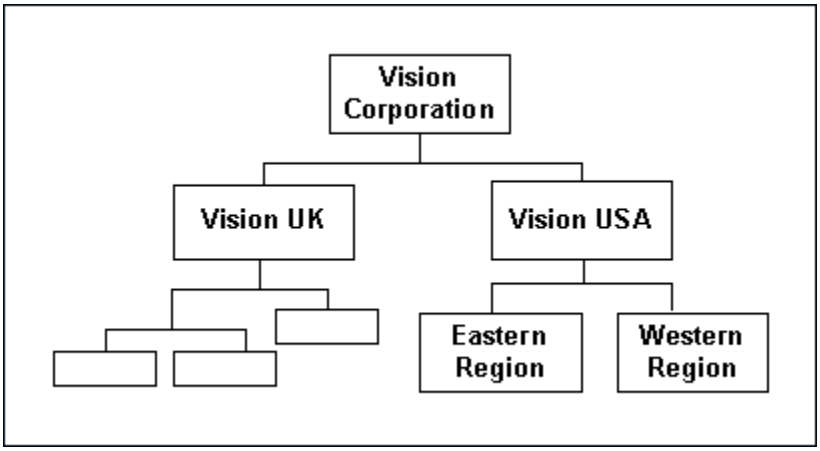
Profile Option Name	Site	Application	Responsibility	User
			Purchasing, Vision Operati	
MO: Default Operating Unit				
MO: Distributed Environment	No			
MO: Operating Unit	Vision Operations		Vision Operations	
MO: Security Profile				
MO: Top Reporting Level	Operating Unit			

To have a secured way for users to only access or report on data for the operating units they have access to, OracleAS Adapter for Oracle Applications uses the MOAC security feature to determine the operating unit access and derive the Organization ID based on relevant profile values.

With MOAC, a system administrator can predefine the scope of access privileges as a security profile, and then use the profile option *MO: Security Profile* to associate the security profile with a responsibility. By using this approach, multiple operating units are associated with a security profile and that security profile is then assigned to a responsibility. Therefore, through the access control of security profiles, users can access data in multiple operating units without changing responsibility.

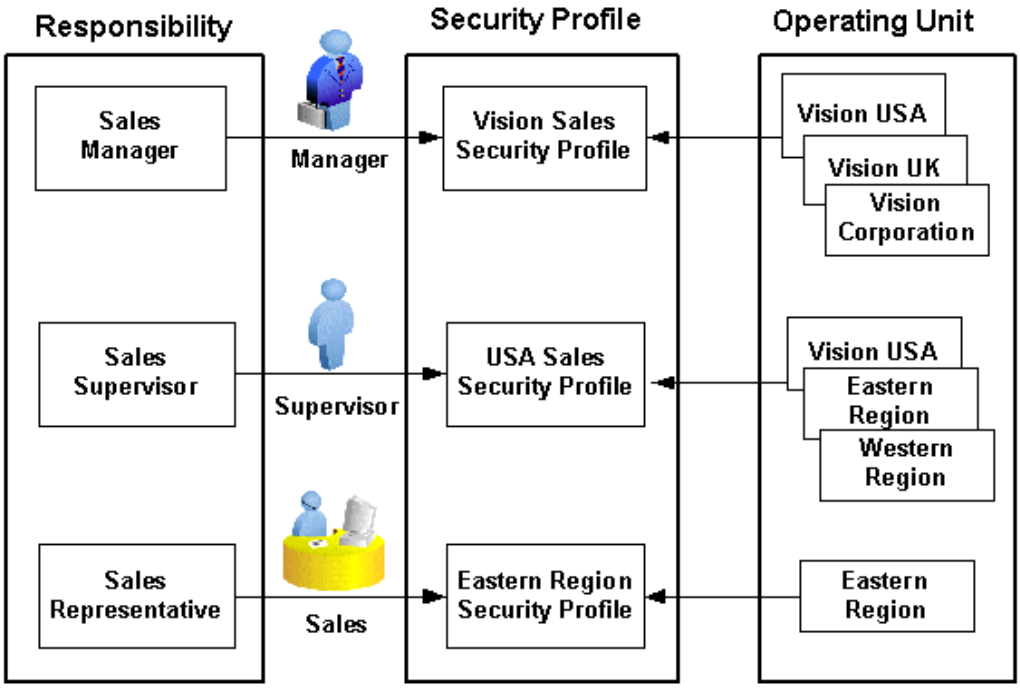
Security profiles are defined based on organization hierarchies. For example, a sales company consists of USA and UK operating units; the USA operating unit has Western Region Sales and East Region Sales. Sales managers are responsible for both USA and UK sales; supervisors are responsible for either USA or UK, and sales representatives are only responsible for their designated sales regions. The Sales organization hierarchy can be illustrated as follows:

Sales Organization Hierarchy



To secure sales data within the company, relevant operating units can be associated with predefined security profiles. For example, all sales data access privileges are grouped into the Vision Sales security profile; USA Sales security profile is for USA related data, and regional security profiles are for designated regional data. The system administrator can associate these security profiles containing multiple operating units with users through appropriate *responsibilities*. Therefore, sales supervisors can easily access sales data in the Eastern or Western region without changing their responsibilities. The following diagram illustrates the relationship between security profiles, responsibilities, and operating units for this sales company:

Relationship Diagram Between Security Profiles, Responsibilities, and Operating Units

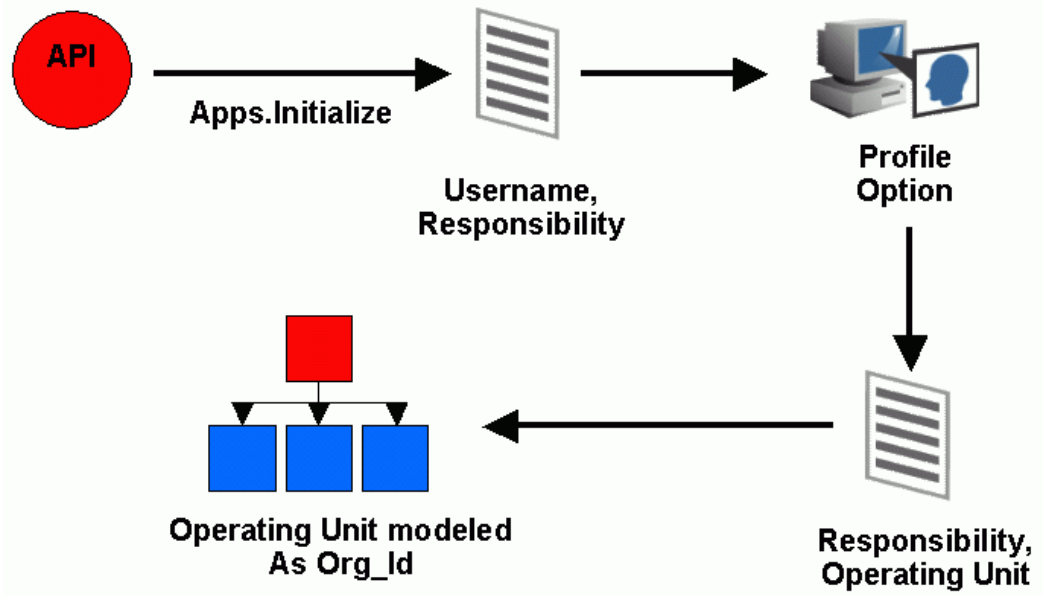


Responsibility Determines Operating Units

Because responsibilities are associated with security profiles that linked to operating units, your responsibility is the key in determining which operating units you will have the access privileges.

The following diagram illustrates how Oracle Applications use the profile options in a multi-organization environment:

Building Applications Context for Multiple Organizations



1. When the system integrator runs, the process achieves the integration with Oracle Applications using PL/SQL APIs.
2. The Apps.Initialize process takes the parameters of Username and Responsibility.
3. With these parameters, a lookup on all System Profile Values assigned to that responsibility is done to determine the Operating Unit within a multi-organization environment.
4. The Operating Unit is modeled as Organization ID derived from the security profile value.
5. The data is read and written into the Oracle Applications with the parameters of Username, Responsibility and Organization ID.

Based on the concept of applications context supported in a multi-organization environment, OracleAS Adapter for Oracle Applications provides the following features:

- Support for Multiple Organization Setups, page 1-17
- Support for Multiple Languages, page 1-20

Supporting for Multiple Organization Setups

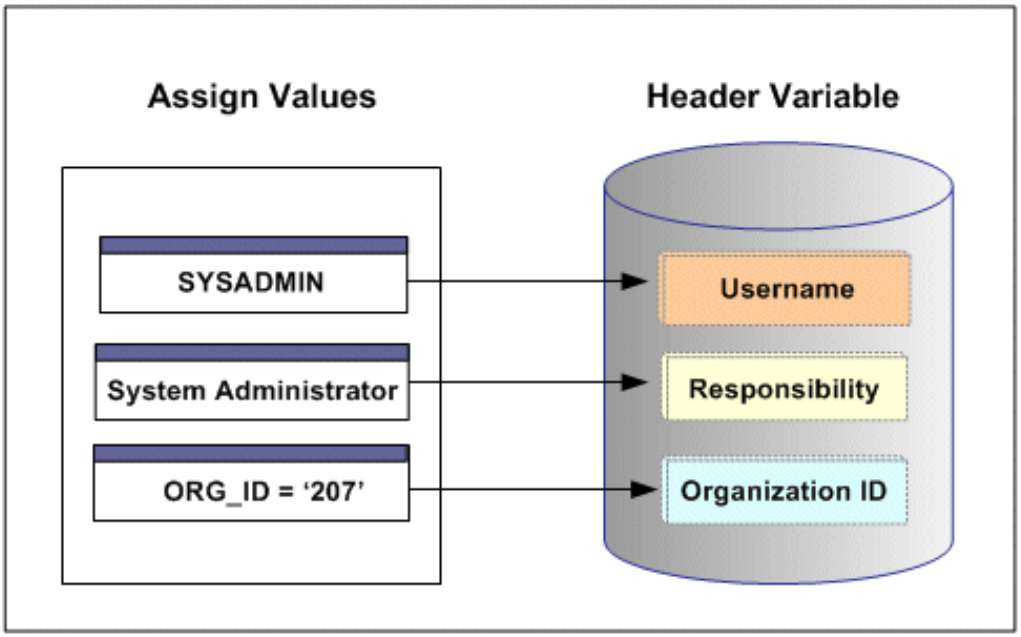
Instead of implicitly deriving organization information from a profile value during the Oracle Applications setups, OracleAS Adapter for Oracle Applications provides a

mechanism which allows Organization ID information can be directly entered through the header variable creation during the design time to support multiple organization setups.

OracleAS Adapter for Oracle Applications uses the header variable to include *Username*, *Responsibility*, and *Organization ID*, the three essential elements for applications context. Once you declare the header variable and assign appropriate values to each parameter contained in the header for a business activity through a PL/SQL API or an interface which requires the applications context to be set, these values in the header will be passed and used as an input to the rest of activities in the BPEL process.

> **Note:** Integration interface types that require applications context to be set are PL/SQL APIs, concurrent programs, and EDI programs.

Creating and Assigning Header Variable



The advantage of having this header variable mechanism in supporting multiple organization setups is that with only one single BPEL process, organization information can be easily placed into multiple organizations within the E-Business Suite if the Organization ID value has been specified in the header. While in the past, since Organization ID is implicitly derived from the profile value based on an application logon user's username and responsibility; therefore, only that associated organization for the invocation of the deployed BPEL process can be inserted.

With the example described earlier in the Multiple Organization Setup section, when a change order is placed within the French sales office, a sales manager from the French office logs on to the system to update the order which invokes a PL/SQL API for that

change. If the Organization ID contained in the header variable has been assigned with a value, such as `207` for the French sales office, the Organization ID associated with the sales manager will be set to French sales office for the invocation of the API.

> **Note:** For Oracle E-Business Suite Release 12, *Organization ID* parameter is automatically included in the header variable along with *Username* and *Responsibility*. For Release 11.5.10, you must apply back port patch 4549743 to Oracle Applications instance in order to have *Organization ID* displayed in the header.

Design-Time Tasks for Organization ID Support in Multiple Organization Setups

OracleAS Adapter for Oracle Applications uses the following procedures to complete the design-time tasks to support Organization ID in multiple organization setups:

1. Create a BPEL project, page 6-2
2. Add a Partner Link, page 6-5
3. Configure an Invoke Activity and Create the Header Variable

   This activity involves the following tasks:

   - Configure an Invoke Activity in the General tab, page 6-23

     Configure an **Invoke** activity by linking the activity to the partner link you just created. This opens the General tab in the Edit Invoke dialog box with Partner Link and Operation information populated. You will create Input Variable and Output Variable for the Invoke activity.

   - Create the Header Variable in the Adapters tab, page 6-26

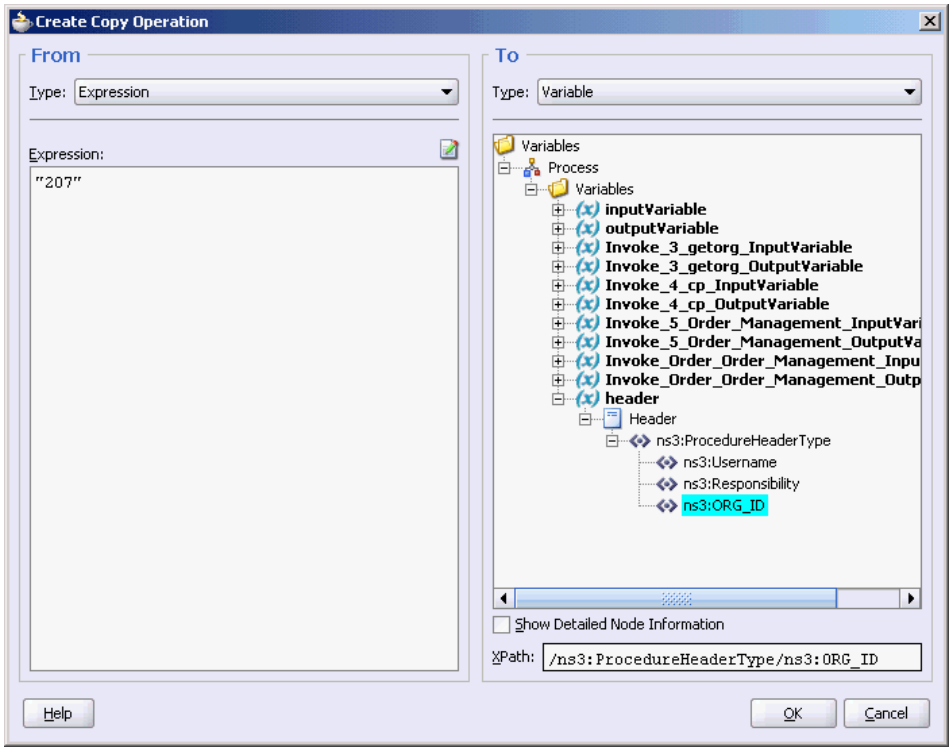
     Create the header variable used in applications context for Oracle Applications to identify the application user and the user's associated organization information.

4. Assign Organization ID Using an Assign Activity

   After creating the header variable, you need to configure an **Assign** activity by placing it before the **Invoke** activity.

   1. Select **Copy Operation** tab in the Assign dialog box and select **Copy Operation...** from the Create drop-down list.
   2. In the From group, enter an ORG_ID value, such as `'207'`, in the Expression field. In the To group, locate the variable header where you declare the variable and assign the Organization ID to ORG_ID parameter of the header.

Assigning Value to ORG_ID



Supporting for Multiple Languages

By leveraging the Multiple Language Support (MLS) feature from Oracle E-Business Suite, OracleAS Adapter for Oracle Applications allows a logon user's default language to be dynamically displayed at runtime when the system deploys a BPEL process through an API requested by the user. When an application user retrieves data from the system for a transaction or receives error messages while executing APIs, the user will find the data or error messages displayed in his or her default language. Additionally, if the user has dates set up based on a region, after data retrieval, that user will find the dates returned with the format and time zone information corresponding to the default information specified in his or her preference page.

To understand how OracleAS Adapter for Oracle Applications supports the MLS feature, you need to first understand how data is retrieved from the application database.

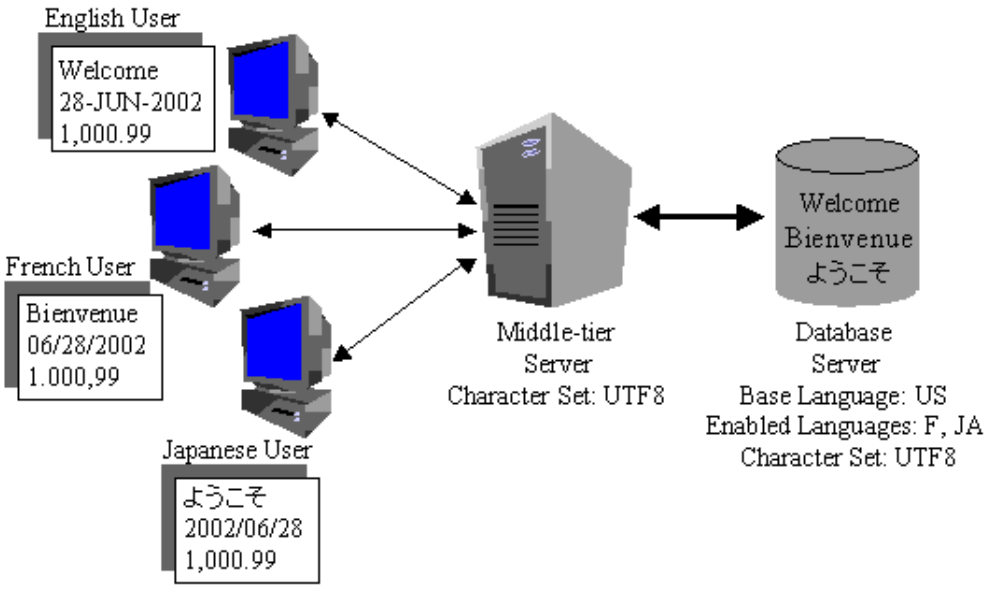
Many of the APIs that OracleAS Adapter for Oracle Applications invokes query database views and these views present data in the default language of the applications user that is used to invoke the APIs. In other words, these APIs exposed within OracleAS Adapter for Oracle Applications are language aware. When an application user requests a transaction through the execution of APIs, data queries from the database views must have been initialized with the default language specified by the

user in the preference page.

To identify the default language used in the database session for data query and retrieval, OracleAS Adapter for Oracle Applications will first examine the *ICX: Language* profile value at all levels including user, responsibility, application, and site. If it is not set at any of those levels, OracleAS Adapter for Oracle Applications then takes `NLS_LANGUAGE` parameter from the database instance National Language Support (NLS) parameters. The NLS parameters initialized in the session are:

- `NLS_LANGUAGE`
- `NLS_SORT`
- `NLS_DATE_FORMAT`
- `NLS_DATE_LANGUAGE`
- `NLS_NUMERIC_CHARACTERS`
- `NLS_TERRITORY`

Dynamically Displaying Language Based on User's Default Language



For example, when a user with a default language Japanese logs into the system and performs a transaction through the execution of APIs that the user defined in the Partner link of the BPEL process, OracleAS Adapter for Oracle Applications uses the *username* of the logon Japanese user and *responsibility* to identify the default language code, such as `JA` for Japanese, used in the database session. Consequently, the NLS context parameters are set in Japanese. That user will therefore see all queried data or error messages displayed in Japanese which corresponds to the user's default language

set in the General Preference page of Oracle Applications.

> **Note:** The default language set in the General Preference page updates the *ICX: Language* profile option.

Please refer to the Set Preferences section, Getting Started with Oracle Applications chapter, *Oracle Applications User's Guide* for the information on how to set the default language used by the applications user.

Design-Time Tasks for MLS Support

To implement the MLS support feature, perform the following design-time tasks before you configure an Invoke activity so that the variable can be passed to the activity:

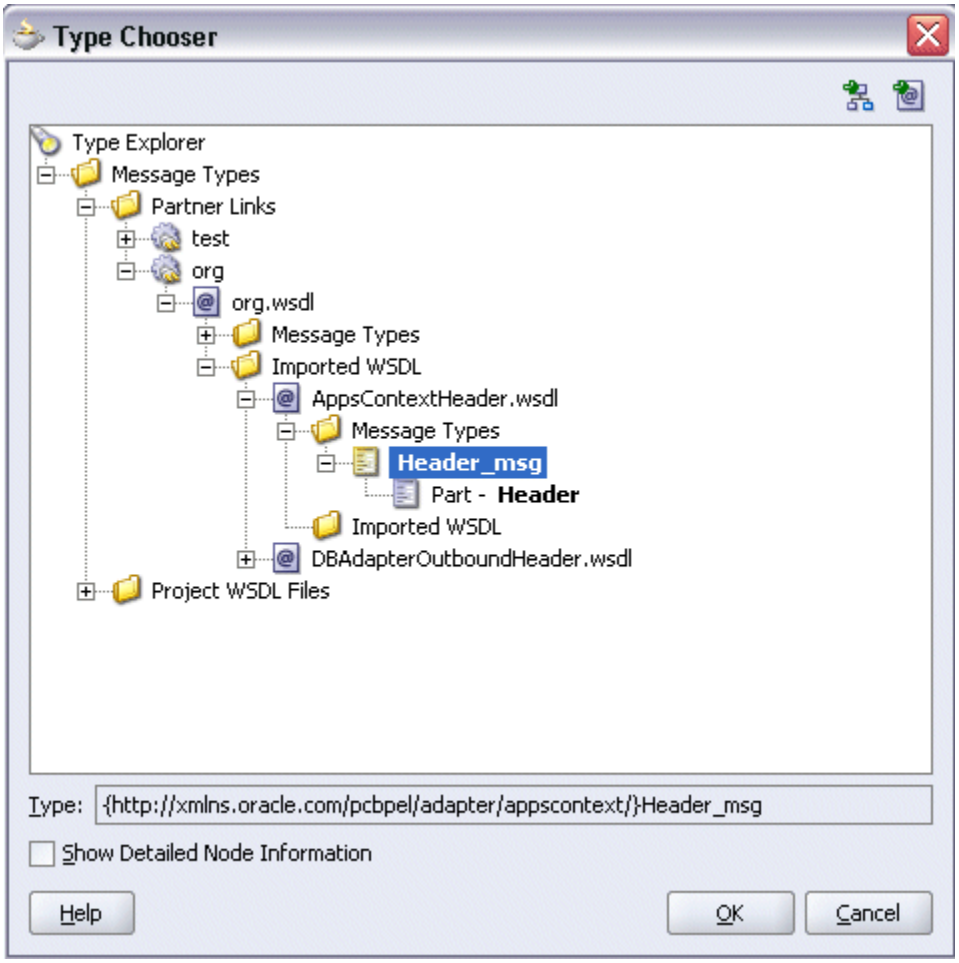
1. Declare the Header Variable (Optional), page 1-22
2. Assign the Variable Using the Assign Activity, page 1-23

Declaring the Header Variable (Optional)

You can optionally define the *username* and *responsibility* to be used in the execution of the API by setting a header variable `{http://xmlns.oracle.com/pcbpel/adapter/appscontext/}Header_msg`. This declaration provides context information for Oracle Applications to identify the application user that will be executing the API.

> **Note:** Since the header variable declaration provides context information for Oracle Applications to identify the application user, OracleAS Adapter for Oracle Applications not only uses this variable declaration to support the MLS feature, but also to support other features that utilize the concept of applications context including the Organization ID support in multiple organization setups.

Declaring Header Variable

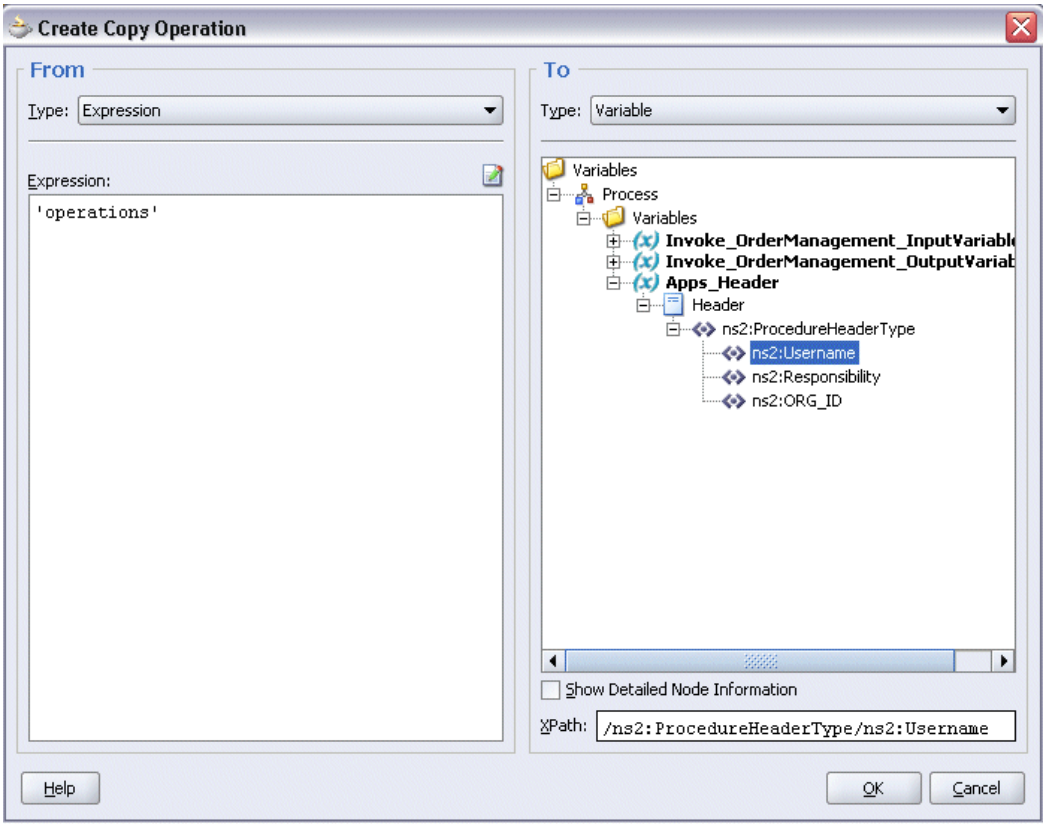


Because the declaration of this header variable is optional, if you do not declare the variable, the default *username* is `SYSADMIN` and the default *responsibility* is `System Administrator`.

Assigning the Variable Using the Assign Activity

After declaring the header variable, you must assign the variable value using an **Assign** activity. For example, in the From group, enter an username, such as `'Operation'`, in the Expression field. In the To group, locate the variable header where you declare the variable and assign the value to username variable.

Assigning a Variable Value



The username defined in the header variable will be used to derive the NLS variables which will be set in the context of the session that executes the API you defined in the Partner link of the BPEL process.

Understanding OracleAS Adapter for Oracle Applications Security

Security is the most critical feature that is designed to guard application content from unauthorized access. By leveraging Oracle User Management function security, Oracle Application Server Adapter for Oracle Applications provides a security feature which only allows users with authorized privileges to execute APIs that they are exposed through the BPEL process to update Oracle Applications. This protects application programming interfaces (APIs) from unauthorized access or execution without security checks.

Please note that Oracle Application Server Adapter for Oracle Applications provides this security support as an optional feature. If you want all login users to access and execute APIs without security checks, you can turn the security feature off using the "EBS Adapter for BPEL, Function Security Enabled" (EBS_ADAPTER_FUNCTION_SEC_ENABLED) profile option.

- If it is set to 'Y', then the function security feature is enabled and all API calls for PL/SQL APIs, Oracle e-Commerce Gateway, and concurrent programs will be checked for user security before they are invoked.
- If it is set to 'N' (default value), then the function security feature is disabled. No security check is implemented during the invocation of all API calls.

> **Note:** To have this function security feature available, appropriate patches need to be applied to your environment. See "Oracle Application Sever Adapter for Oracle Applications Documentation Update, Release 10*g*" Oracle*MetaLink* Document 464164.1 for details.

This section includes the following topics:

- Function Security for OracleAS Adapter for Oracle Applications, page 1-25
- Creating Security Grants, page 1-27

Function Security for OracleAS Adapter for Oracle Applications

Function security is the basic access control in Oracle Applications. It restricts user access to individual menus and menu options within the system regardless of which application data in the row. Since APIs are stored procedures that enable you to insert and update data in Oracle Applications, when having the function security layer enforced on the access to an API, it actually implicitly restricts the data access to the application.

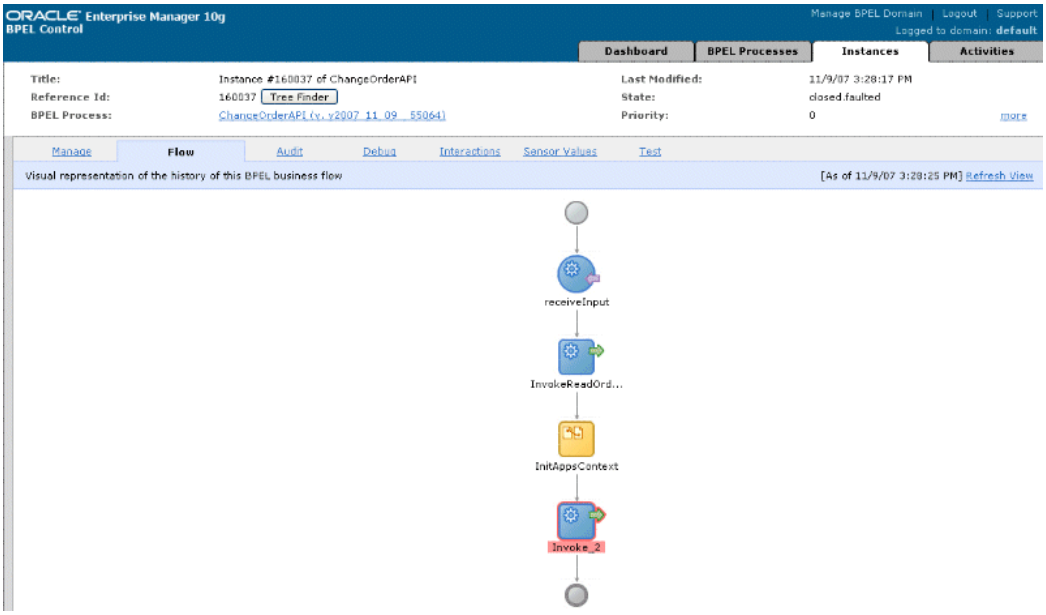
To allow appropriate users with right privileges to execute APIs, OracleAS Adapter for Oracle Applications leverages Oracle User Management Role-Based Access Control security (RBAC) to reinforce the function security through user roles and whether a user can access an API is determined by the roles granted to the user. A role can be configured to consolidate the responsibilities, permissions, permission sets, and function security policies that users require to perform a specific function. This simplifies mass updates of user permissions because changes can be done through roles which will inherit the new sets of permissions automatically. Based on the job functions, each role can be assigned a specific permission or permission set if needed. For example, a procurement organization may include 'Buyer', 'Purchasing Manager', and 'Purchasing Support' roles. The 'Purchasing Manager' role would include a permission set that contains all Purchase Order (PO) Creation, PO Change, and Contract PO related APIs allowing the manager role to perform a job function while the Buyer or Support role may not have the access privileges.

In OracleAS Adapter for Oracle Applications, all annotated APIs resided in Oracle Integration Repository are registered on the FND_FORM_FUNCTIONS table so that the function security (FND_FORM_FUNCTIONS) can be applied. This allows the creation of a secured function for each API.

Important: For Oracle E-Business Suite Release 12, all annotated APIs in Oracle Integration Repository are registered on the FND_FORM_FUNCTIONS table. However, if it is for the Release 11*i*, you have to apply a function security patch that populates this information on that table. Refer to "Oracle Application Sever Adapter for Oracle Applications Documentation Update, Release 10*g*" Oracle *MetaLink* Document 464164.1 for details.

By leveraging the concept of permission sets, OracleAS Adapter for Oracle Applications allows related APIs to be grouped and sequenced under one permission set; each permission set can be associated with a function role and then assigned to users through security grants. When a user logs on to the E-Business Suite and tries to access an API exposed through the BPEL process, if the security feature is enabled, the function security API will be invoked to validate whether the user is authorized to have the execution privileges on the API.

For example, if a user does not have the access privileges for a PL/SQL API exposed through a BPEL process, the execution of that BPEL process will fail while trying to invoke the PL/SQL API as shown in the following diagram:



Without the authorized privileges, the Function Security Validation Exception message will be raised indicating that the user does not have the privilege for a specific PL/SQL API.

For more information on Function Security and RBAC security models, see *Oracle Applications System Administrator's Guide - Security* for details.

Creating Security Grants

To secure the API invocation only to a user with appropriate execution privileges, OracleAS Adapter for Oracle Applications uses the following steps to create security grants to users through user roles:

1. Creating a Permission Set, page 1-27
2. Creating a User Role, page 1-29
3. Granting a Permission Set to a User Through a Role, page 1-30

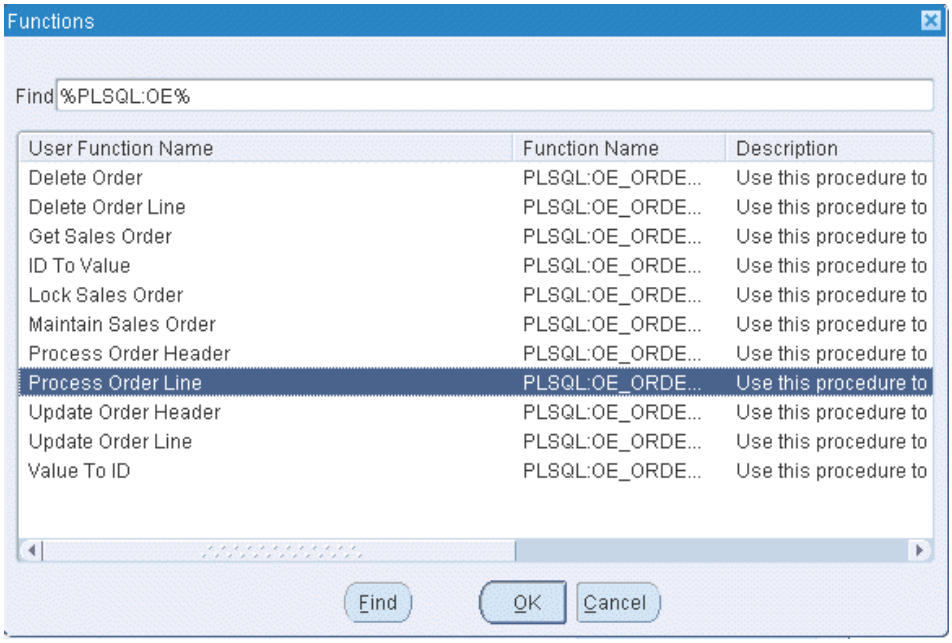
Creating a Permission Set

Use the following steps to create a permission set:

1. Log in to Oracle E-Business Suite using the System Administrator responsibility.
2. Select Application: Menu from the Navigator to access the Menus window.
3. Enter the following menu information:
   - Menu: Enter an appropriate menu name (such as 'OE_PROCESS_LINE_PS')
   - User Menu Name: Enter an appropriate user menu name (such as 'Order Manager Process Line Permission Set')
   - Menu Type: Permission Set
   - Description: Enter description information for this menu
4. Add all the functions that you want to group on this Permission Set by entering values for Seq and Function.
   1. Enter the Seq field.
   2. In the Function column, search for the functions you want to assign to this permission set.

      Select an appropriate function name by performing a search in the Functions window. For example the syntax for searching public PL/SQL APIs is: `PLSQL:<package name>:<procedure name>`. You can enter `%PLSQL:OE%` in the Find field and click **Find** to execute the search.

Searching for Functions

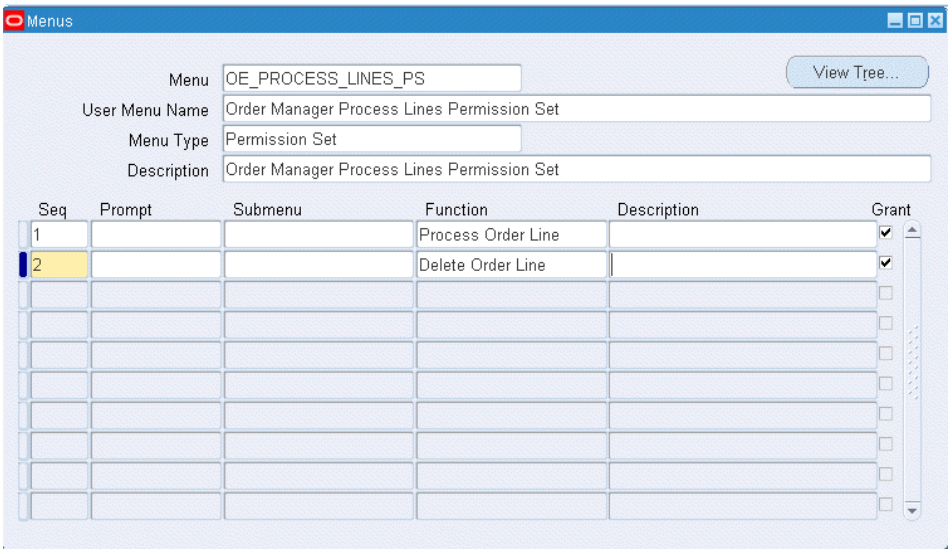


Based on your BPEL process, select appropriate functions and then grant the permissions to the APIs that you will be invoking from the BPEL process.

For example, for a sales order line change BPEL process, you select sales order line change related functions contained in the order change PL/SQL API and group them as a permission set, and then grant the permission set to an appropriate user through a role.

See: Creating a User Role, page 1-29.

Permission Set Menu



5. Save the Permission Set.

Creating a User Role

Permission sets are granted through user roles. Therefore, you must first create a role and then assign the role to a user.

Use the following steps to create a user role:

1. Log in to Oracle E-Business Suite using the User Management responsibility.
2. Select Roles & Role Inheritance from the Navigator to access the Roles & Role Inheritance page.
3. Click **Create Role** to access the Create Role page.
4. Enter the following information to create a role:
   - Category: Select Miscellaneous from the drop-down list
   - Role Code: Enter an appropriate role code (such as 'EBS_ADAPTER_ROLE')
   - Display Name: Enter appropriate information for the display name (such as 'EBS Adapter Role')
   - Description: Enter appropriate information for the description (such as 'EBS Adapter Role')

- Application: Select an appropriate application (such as 'Application Object Library')
- Active Date: Enter an appropriate date which is earlier than or equal to today's date so that the role can become valid right away

5. Save the information and click **Create Grant**.
6. Enter the following information in the Create Grant: Define Grant page:
   - Name: Enter an appropriate name (such as 'EBS_ADAPTER_GRANT')
   - Description: Enter description information for this grant
7. Click **Next**.
8. In the Create Grant: Define Object Parameters and Select Set page, select the Permission Set you created earlier in the Creating a Permission Set section, page 1-27 and click **Next**.
9. Click **Finish**.

Granting a Permission Set to a User Through a Role

Use the following steps to grant a permission set to a user through a role:

1. Log in to Oracle E-Business Suite using the User Management responsibility.
2. Select Users from the Navigator to access the User Maintenance page.
3. Search for the user you want to assign the role and click **Go**.
4. Select the **Update** icon next to the user name that you want to assign the role.
5. In the Update User page, click **Assign Roles** to have the Search window populated which allows you to search for the role that you created earlier.
6. Select the role (such as 'EBS_ADAPTER_ROLE') and save your update.

Installing OracleAS Adapter for Oracle Applications

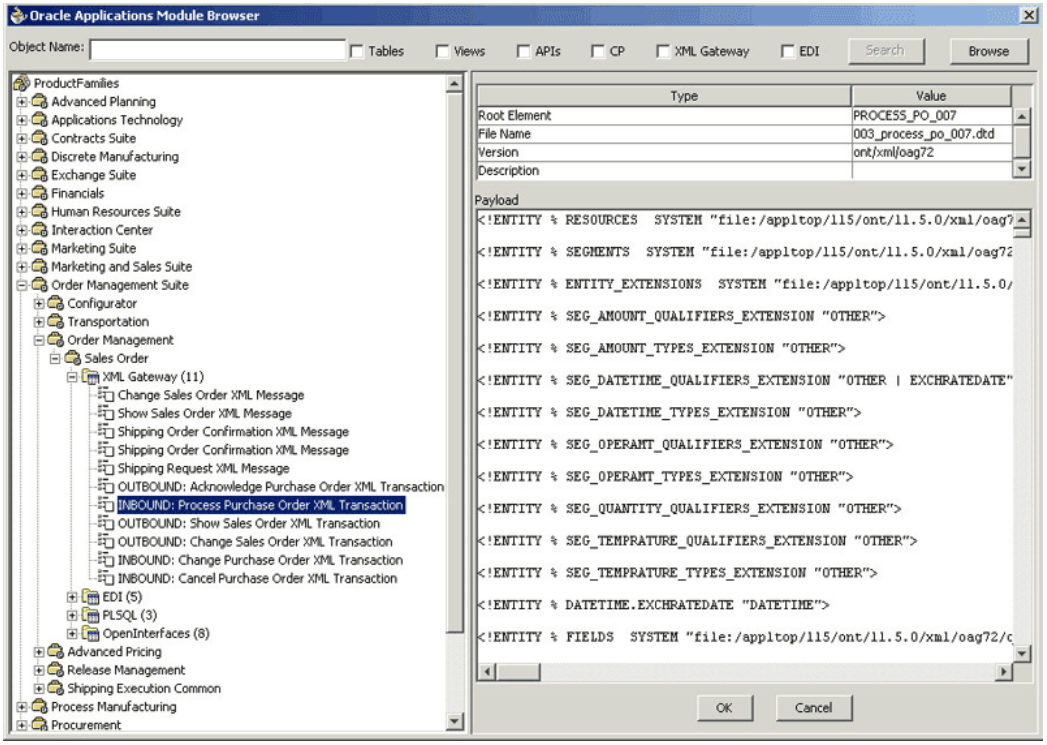
The installation of the Oracle Application Server Adapter for Oracle Applications happens out-of-the-box with the Oracle BPEL Process Manager (PM) product. OracleAS Adapter for Oracle Applications is deployed using the Oracle BPEL PM in Oracle JDeveloper.

Note: Refer to *Oracle Application Server Integration Business Activity Monitoring User's Guide* for more details about installing Oracle BPEL Process Manager. Refer to the section "Notes on Installing Oracle BPEL Process Manager."

Using the Oracle Applications Module Browser

In addition to the interfaces that are made available through Oracle Integration Repository, OracleAS Adapter for Oracle Applications enables you to use business events, customized PL/SQL APIs, customized XML Gateway maps, and selected concurrent programs, all of which you can explore using the Oracle Applications Module Browser.

Oracle Applications Module Browser



The Oracle Applications Module Browser is a key component of OracleAS Adapter for Oracle Applications. You use the Module Browser to select the interface needed to define a partner link. The Module Browser combines interface data from Oracle Integration Repository with information about the additional interfaces supported by OracleAS Adapter, organized in a tree hierarchy as follows:

```
ProductFamilies
 |-[product_family]
 |   |-[product]
 |       |-[business_entity]
 |           |-XML Gateway ([n])
 |           |-EDI ([n])
 |           |-PLSQL ([n])
 |           |   |-[package_name]
 |           |-OpenInterfaces ([n])
 |              |-[OpenInterface_name]
 |                 |-Tables ([n])
 |                 |-Views ([n])
 |                 |-ConcurrentPrograms ([n])
 |-Other Interfaces
    |-Business Events
    |-Custom Objects
       |-PLSQL APIs
       |   |-[package_name]
       |-XMLGateway Maps
          |-Inbound
          |-Outbound
```

Note: The items under Other Interfaces, as well as certain PL/SQL APIs and concurrent programs under the *[product family]* hierarchy, are available through OracleAS Adapter for Oracle Applications, but not through Oracle Integration Repository.

The Oracle Integration Repository interface data populates the ***[product_family]*** sections, grouped according to the products and business entities to which they belong. Each interface type heading is followed by a number ***[n]*** indicating how many of that type are listed in that section.

Business events appear under Other Interfaces. Customized XML Gateway maps appear under **Other Interfaces > Custom Objects,** categorized as either inbound or outbound.

Customized PL/SQL APIs appear in two places:

- Procedures within a package that's already exposed via Oracle Integration Repository appear under the package name within a product family hierarchy.
- Procedures within a completely new package appear under the package name, under **Other Interfaces > Custom Objects**.

General Issues and Workarounds

This section describes the following issues and workarounds:

- **WSDL Context Information Default Values**

  In the current release, BPEL Process manager generates WSDL code containing context information, including the username and responsibility of an Oracle

Applications user who has sufficient privileges (based on the applications context of Organization ID, Username and Responsibility) to run the program.

By default the **Username** value is set to `SYSADMIN` and the **Responsibility** value is set to `SYSTEM ADMINISTRATOR`. To change these values, you must manually edit the WSDL file.

- **Correlation ID Defaults to BPEL for XML Gateway Transactions**

  The Adapter Configuration wizard of OracleAS Adapter for Oracle Applications does not specify a correlation ID for XML Gateway transactions for inbound or outbound interfaces. Instead, a default correlation ID of `BPEL` is automatically set in the WSDL file. To make this configuration work, you must configure Oracle Applications to set the same correlation ID value of `BPEL` for the corresponding XML Gateway transactions.

  If you want the Adapter to use a different correlation ID than the default, you need to configure a correlation ID in Oracle Applications, then edit the `Correlation="BPEL"` line contained in the `<jca:operation>` section of the adapter service WSDL. Replace `BPEL` with the string value of the correlation ID you specified in Oracle Applications.

- **Workaround for Stored Procedures Using Complex Types and the DEFAULT Clause**

  When working with stored procedures for which the Adapter Configuration wizard must generate wrapper SQL stored procedures, there is a current limitation on DEFAULT clauses not being carried over to the generated wrapper stored procedures.

  As a workaround, perform the following steps one time only for a given stored procedure:

  1. Open the generated wrapper SQL script.
  2. Copy all default clauses from the base-stored procedure into the corresponding wrapper.
  3. Use SQL*Plus to reload the wrapper SQL script into the database.
  4. Edit the generated XSD. If a parameter has a DEFAULT clause, its corresponding element in the XSD must have the extra attribute: `db:default="true"`

     For example, with the following SQL:

     ```
     FINANCE$INVOICE(isTrue INTEGER DEFAULT 1, value NUMBER DEFAULT 0)
     ```

     The elements in the XSD for `isTrue` and `value` must have the new attribute:

     ```
     <element name="ISTRUE" ... db:default="true" .../>
     <element name="VALUE" ... db:default="true" .../>
     ```

- **One-time Workaround for Concurrent Programs and E-Commerce Gateway Interfaces**

  When working with Concurrent Programs and E-Commerce Gateway interfaces, you must perform the following workaround exactly once for a given E-Business Suite instance.

  > **Note:** This is to work around the known issue with the Adapter Configuration wizard being unable to preserve DEFAULT clauses for PL/SQL wrappers that it generates underneath the covers.

  Load the following SQL file into the apps schema (using SQL*Plus) before launching the Oracle Applications adapter of the Adapter Configuration wizard to create services for either Concurrent Programs or E-Commerce Gateway Interfaces.

  ```
  ORACLE_HOME
  \bpel\samples\tutorials\150.AppsAdapter\OrderImportConcurrentProgram
  \bpel\XX_BPEL_FND_REQUEST_SUBMIT_REQUEST.sql
  ```

- **Cannot Create a Partner Link If the Underlying API Has Been Recreated**

  The generation of a wrapper for an API that was recreated with the same name, but with a different set of parameters, will fail.

  > **Note:** This can happen for both packaged procedures and top-level or root procedures that require generated wrappers.

  The following example illustrates the problem:

  1. Create the initial API that, in this case, is defined at the top level:

     ```
     SQL> create procedure test (a number, b varchar2, c BOOLEAN)
     ```

     The `BOOLEAN` parameter indicates that a wrapper is necessary.

  2. Use the database adapter for stored procedures in the Adapter Configuration wizard to generate and load the wrapper for this API.

  3. Drop the API, then recreate it with a different set of parameters:

     ```
     SQL> drop procedure test
     SQL> create procedure test (a number, b varchar2, c number, d
     BOOLEAN)
     ```

  4. An attempt to generate a partner link for this API using the Adapter Configuration wizard will fail with the following message:

     ```
     The wrapper procedure, TOPLEVEL$TEST, could not be found
     ```

  5. As a workround, exit JDeveloper BPEL Designer and restart it after recreating the stored procedure, but before attempting to create the second partner link.

2

Using XML Gateway

This chapter covers the following topics:

- Overview of XML Gateway
- Design-Time Tasks for XML Gateway Inbound Messaging
- Creating a New BPEL Project
- Creating a Partner Link
- Configuring the Invoke Activity
- Adding a Partner Link for the File Adapter
- Configuring the Receive Activity
- Configuring the Assign Activity
- Run-Time Tasks for XML Gateway Inbound Messaging
- Deploying the BPEL Process
- Testing the BPEL Process
- Verifying Records in Oracle Applications
- Design-Time Task for XML Gateway Outbound Messaging
- Configuring XML Gateway for Outbound Messaging
- Run-Time Task for XML Gateway Outbound Messaging
- Testing the BPEL Process
- Troubleshooting and Debugging

Overview of XML Gateway

The OracleAS Adapter for Applications provides a bridge between Oracle Applications and third party applications. Inbound and outbound XML data is exchanged between Oracle Applications and third party applications through the XML Gateway.

Oracle XML Gateway provides a common, standards-based approach for XML integration between Oracle Applications and third party applications, both inside and outside your enterprise. XML is key to an integration solution, as it standardizes the way in which data is searched, exchanged, and presented thereby enabling interoperability throughout the supply chain.

Oracle XML Gateway is a set of services that allows easy integration with Oracle Applications to support XML messaging. Oracle Applications utilize the Oracle Workflow Business Event System to support event-based XML message creation and consumption.

OracleAS Adapter for Applications can be configured to use XML Gateway to interact with third party applications. The tight integration provided by open interface tables is not suitable for those scenarios where trading partners change frequently. XML Gateway is an ideal solution when you need to interact with third party applications that use open standards. Moreover, it is also suitable for scenarios where trading partners change frequently.

Standards-Based Messaging

As a provider of broad based business application solutions to support all industries, Oracle XML Gateway supports all Document Type Definition (DTD) based XML standards. The majority of the Oracle prebuilt messages delivered with Oracle Applications are premapped using the Open Application Group (OAG) standard. Any Oracle prebuilt message map may be remapped to your standard of choice using the XML Gateway Message Designer.

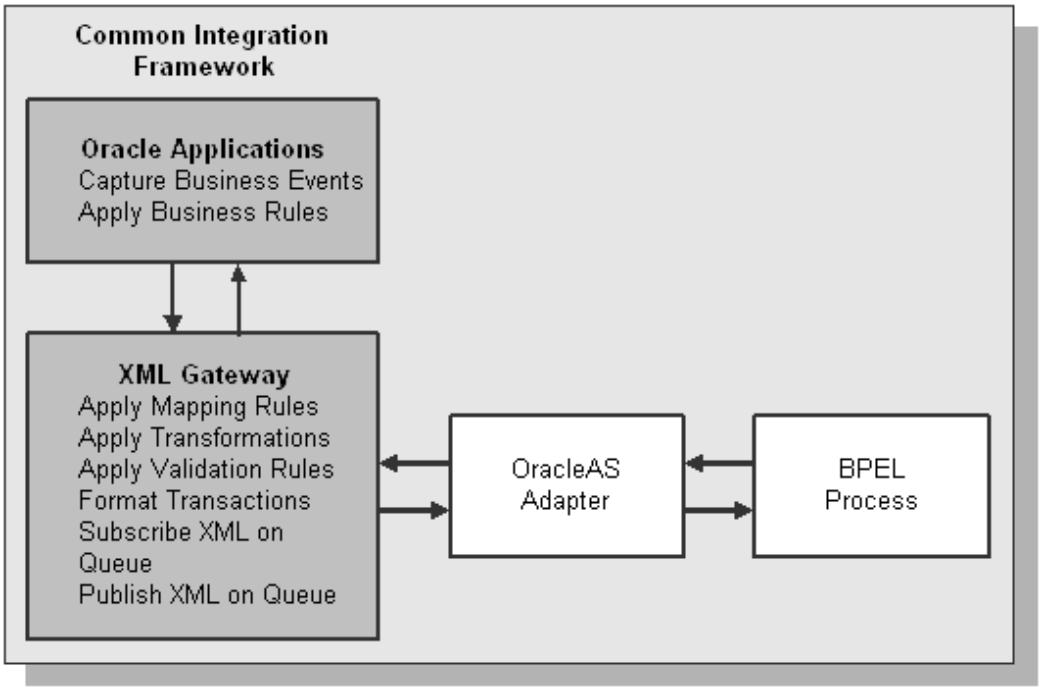
Integration Architecture

XML Gateway provides an application integration infrastructure that is flexible enough to accommodate the integration requirements of any application that needs to integrate with Oracle Applications. XML Gateway enables you to create an efficient and responsive supply chain that links all customers, factories, warehouses, distributors, carriers, and other trading partners. All these entities can seamlessly operate as a single enterprise.

Oracle XML Gateway supports both Business-to-Business (B2B) and Application-to-Application (A2A) initiatives. B2B initiatives include communicating business documents and participating in industry exchanges. An example of an A2A initiative is data integration with legacy and disparate systems.

XML Gateway enables bidirectional integration with Oracle Applications by allowing you to insert and retrieve data from Oracle Applications. The Oracle Applications adapter for XML Gateway supports two operations, namely, enqueue and dequeue. The enqueue operation is used to put messages in the queue. This is used to insert inbound data into Oracle Applications. The dequeue operation polls for messages from the queue. This is used to retrieve outbound data from Oracle Applications. Only one operation can be defined for each adapter service.

XML Gateway Integration Architecture



Message Queues

The XML Gateway uses queues specifically at two points in the process as well as employing a general error queue. The first point is at the transport agent level between the transport agent module and the XML Gateway. The second point is at the transaction level between base Oracle Applications products and the XML Gateway.

Inbound Queues

Inbound message queues are used for XML messages inbound into Oracle Applications. Inbound message queues are positioned between the Transport Agent and the Oracle Workflow Business Event System.

The messages must be formatted according to the XML Gateway envelope message format. The envelope message format is discussed under XML Gateway Envelope, page 2-4. Oracle Workflow Business Event System copies the inbound messages to the proper inbound Transaction Queue.

Outbound Queues

Outbound message queues are used for XML messages outbound from Oracle Applications. The outbound Message Queue is positioned between the XML Gateway and the Transport Agent.

The XML Gateway creates XML messages, then enqueues them on this queue. The

Transport Agent dequeues the message and delivers it to the Trading Partner.

XML Gateway Envelope

In addition to the business document such as a purchase order or invoice in the XML Payload, a set of message attributes are also transmitted. Collectively, these attributes are called the XML Gateway envelope. The following table describes some of these attributes.

Envelope Attributes

Attribute	Description
MESSAGE_TYPE	Payload message format. This defaults to `XML`. Oracle XML Gateway currently supports only `XML`.
MESSAGE_STANDARD	Message format standard as displayed in the Define Transactions form and entered in the Define XML Standards form. This defaults to `OAG`. The message standard entered for an inbound XML document must be the same as the message standard in the trading partner setup.
TRANSACTION_TYPE	External Transaction Type for the business document from the Trading Partner table. The transaction type for an inbound XML document must be the same as the transaction type defined in the Trading Partner form.
TRANSACTION_SUBTYPE	External Transaction Subtype for the business document from the Trading Partner table. The transaction subtype for an inbound XML document must be the same as the transaction subtype defined in the Trading Partner form.
DOCUMENT_NUMBER	The document identifier used to identify the transaction, such as a purchase order or invoice number. This field is not used by the XML Gateway, but it may be passed on inbound messages.
PROTOCOL_TYPE	Transmission Protocol as defined in the Trading Partner table.
PROTOCOL_ADDRESS	Transmission address as defined in the Trading Partner table.

Attribute	Description
USERNAME	USERNAME as defined in the Trading Partner table.
PASSWORD	The password associated with the USERNAME defined in the Trading Partner table.
PARTY_SITE_ID	The party site identifier for an inbound XML document must be the same as the Source Trading Partner location defined in the Trading Partner form.
ATTRIBUTE3	For outbound messages, this field has the value from the Destination Trading Partner Location Code in the Trading Partner table. For inbound messages, the presence of this value generates another XML message that is sent to the trading partner identified in the Destination Trading Partner Location Code in the Trading Partner table. This value must be recognized by the hub to forward the XML message to the final recipient of the XML Message. **Note:** For more information, see *Destination Trading Partner Location Code* in the *Oracle XML Gateway User's Guide*. This guide is a part of the Oracle Applications documentation library. Oracle Applications documentation can be accessed from the following link: http://www.oracle.com/technology/documentation/applications.html
PAYLOAD	The XML message.

Parameters defined by the Application

The following parameters may be defined by the base application:

- `ATTRIBUTE1`
- `ATTRIBUTE2`
- `ATTRIBUTE4`
- `ATTRIBUTE5`

Parameters Not Used

The following parameters are not used:

- `PARTYID`
- `PARTYTYPE`

> **Note:** See *Oracle XML Gateway User's Guide* for details on the XML Gateway Execution Engine, Trading Partner validation, and so on. This guide is a part of the Oracle Applications documentation library. Oracle Applications documentation can be accessed from the following link:
>
> http://www.oracle.com/technology/documentation/applications.html

Design-Time Tasks for XML Gateway Inbound Messaging

OracleAS adapter for Oracle Applications is deployed using the BPEL Process Manager (PM) in Oracle JDeveloper. The BPEL PM creates the WSDL interfaces for the XML Gateway message map.

This section describes configuring the OracleAS Adapter for Oracle Applications to use XML Gateway. It describes the tasks required to configure OracleAS Adapter for Oracle Applications using the Adapter Configuration Wizard in Oracle JDeveloper.

Prerequisites to Configure XML Gateway Inbound

You need to populate certain variables in the BPEL PM in order to provide context information for Oracle Applications. The `MESSAGE_TYPE`, `MESSAGE_STANDARD`, `TRANSACTION_TYPE`, `TRANSACTION_SUBTYPE`, and `PARTY_SITE_ID` are the mandatory header variables that you need to populate for the XML transaction to complete successfully. Refer to Configuring the Assign Activity, page 2-36 for more information.

You also need to configure and schedule two listeners on the Oracle Applications side. These are the ECX Inbound Agent Listener and the ECX Transaction Agent Listener. Use the following steps to configure these listeners in Oracle Applications:

1. Log in to Oracle Applications with the responsibility of Workflow Administrator.
2. The Navigator page . Click the **Workflow Administrator Web Applications** link.
3. Click the **Workflow Manager** link under Oracle Applications Manager.
4. Click the status icon next to **Agent Listeners**.
5. Configure and schedule the **ECX Inbound Agent Listener** and the **ECX Transaction Agent Listener**. Select the listener, and select Start from the **Actions** box. Click **Go**.

Following is a list of the procedures required to accomplish the design-time tasks.

1. Create a new BPEL project, page 2-7.
2. Create a partner link, page 2-10.
3. Configure the Invoke activity, page 2-21.
4. Add a partner link for the file adapter, page 2-27.
5. Configure the Receive activity, page 2-33.
6. Configure the Assign activity, page 2-36.

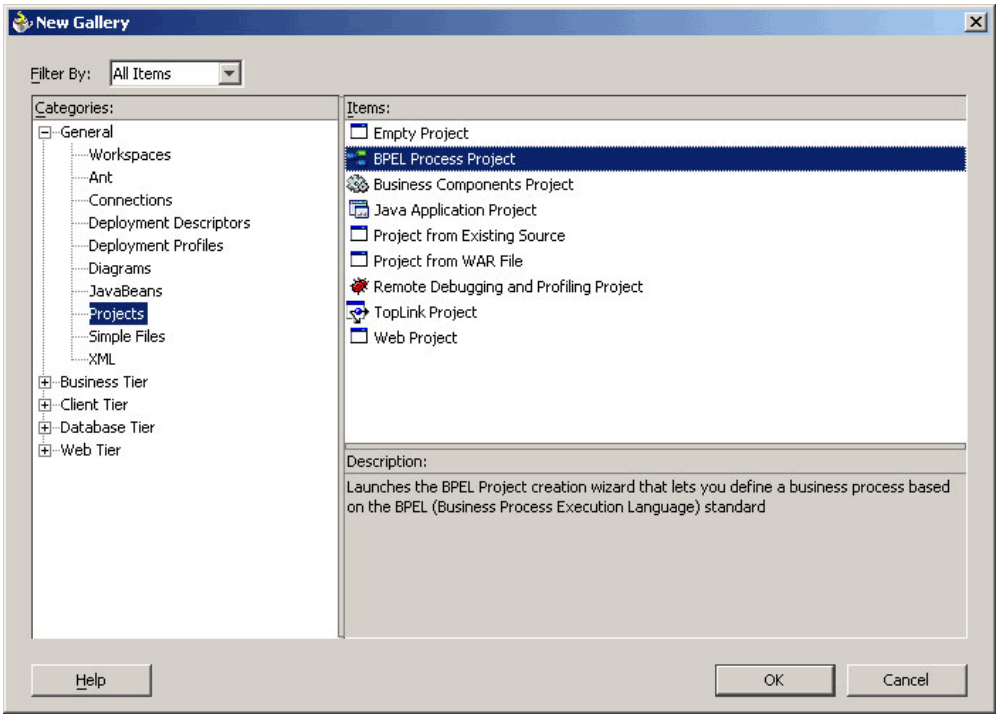
Creating a New BPEL Project

To create a new BPEL project:

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

   The New Gallery dialog box appears.
3. Select **All Items** 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.

Creating a New BPEL Process Project

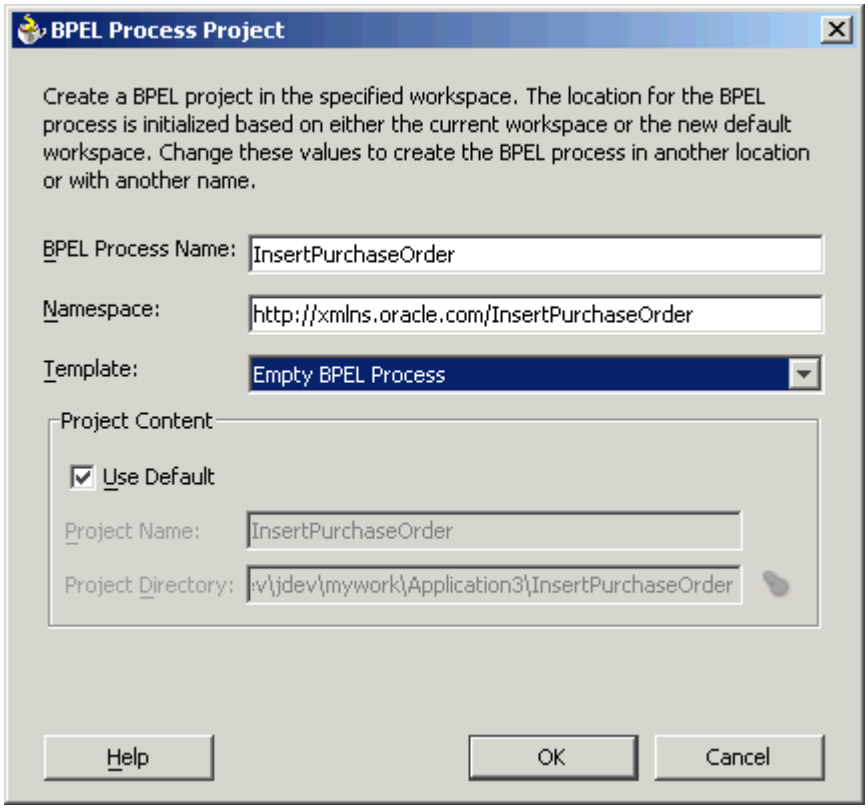


6. Click **OK**. The BPEL Process Project dialog box appears.

7. In the **BPEL Process Name** field, enter a descriptive name. For example, `InsertPurchaseOrder`.

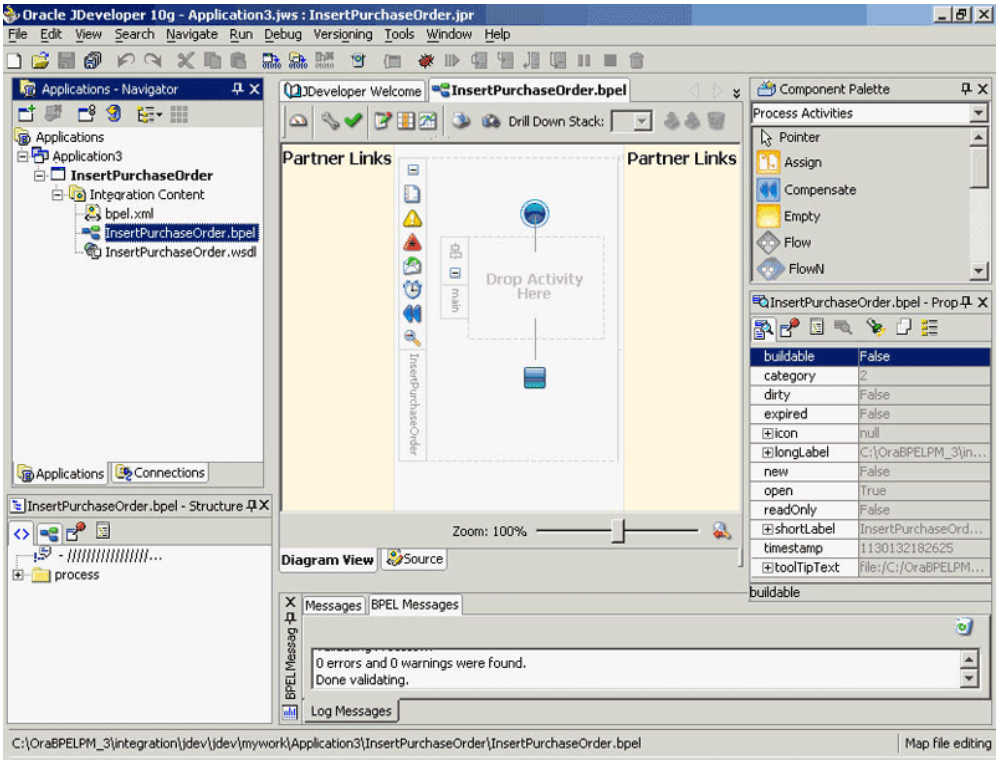
8. From the **Template** box, select **Empty BPEL Process.** Keep the default selection for **Use Default** under Project Content.

Specifying a Name for the New BPEL Process Project



9. Click **OK**. A new BPEL process, with the required source files including `bpel.xml`, `InsertPurchaseOrder.bpel`, and `InsertPurchaseOrder.wsdl`, is created.

New BPEL Process Project



Creating 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.

To create a partner link:

1. Drag and drop **PartnerLink**, from the Component Palette, into the border area of the process diagram. The Create Partner Link dialog box .

2. Click the **Define Adapter Service** icon in WSDL Settings. The Adapter Configuration Wizard .

3. Click **Next**. The Adapter Type dialog box .

4. Select **Oracle Applications.**

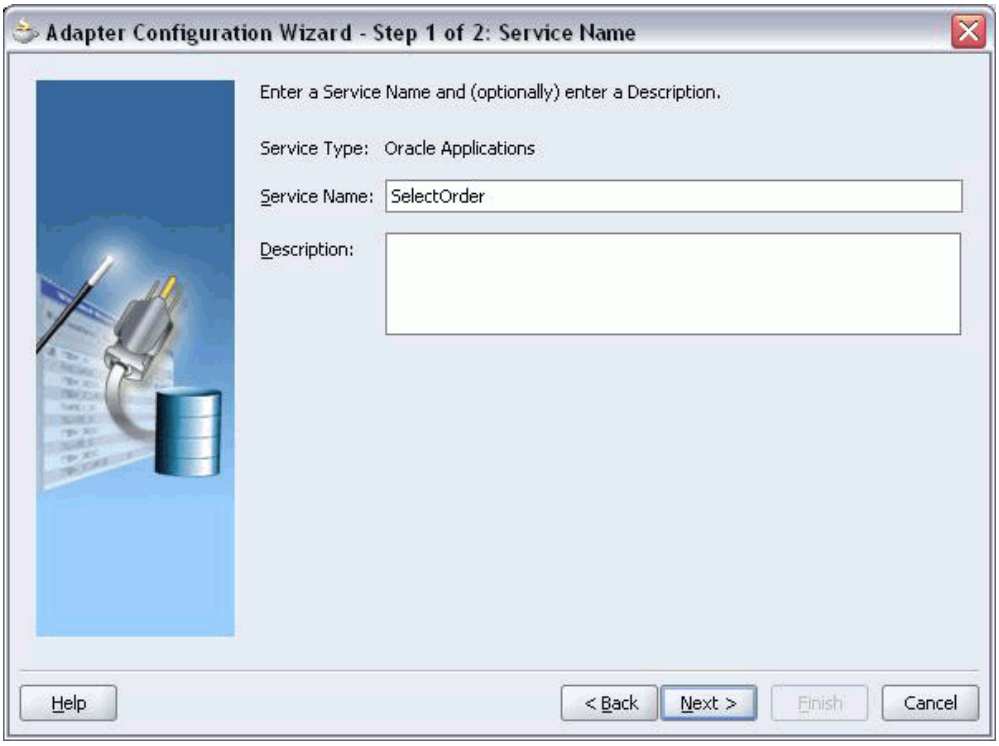
Selecting OracleAS Adapter for Oracle Applications

5. Click **Next.** The Service Name dialog box appears. Enter the following information:

   1. In the **Service Name** field, enter a service name.

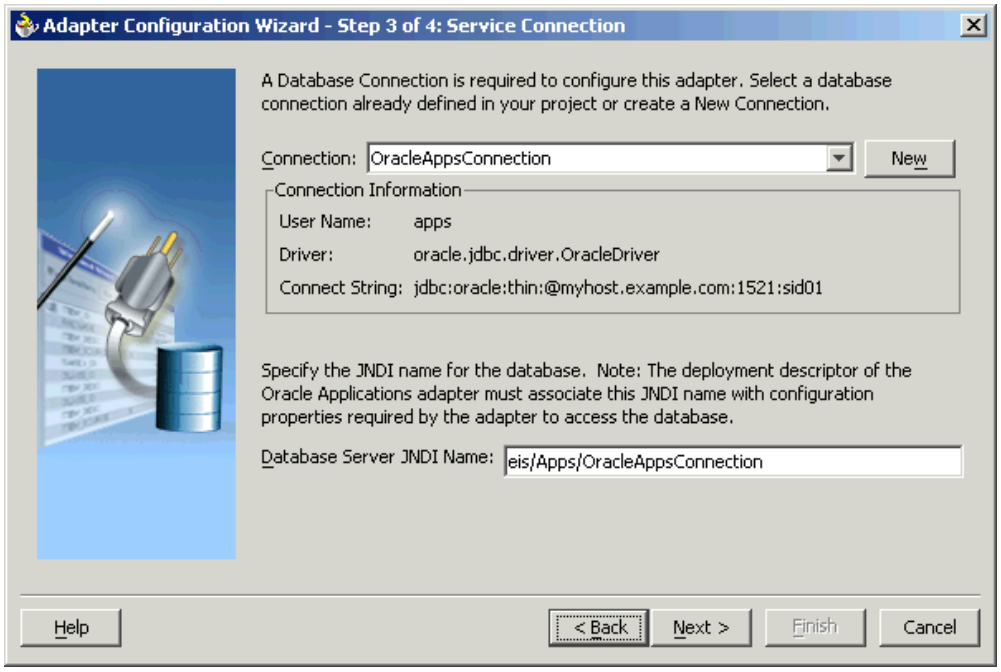
   2. In the **Description** field, enter a description for the service. This is an optional field.

Specifying the Service Name



6. Click **Next**. The Service Connection dialog box appears.

Creating a New Database Connection



7. Click **New** to define a database connection. The Create Database Connection Wizard appears. Alternatively, you can select an existing database connection from the Connection list.

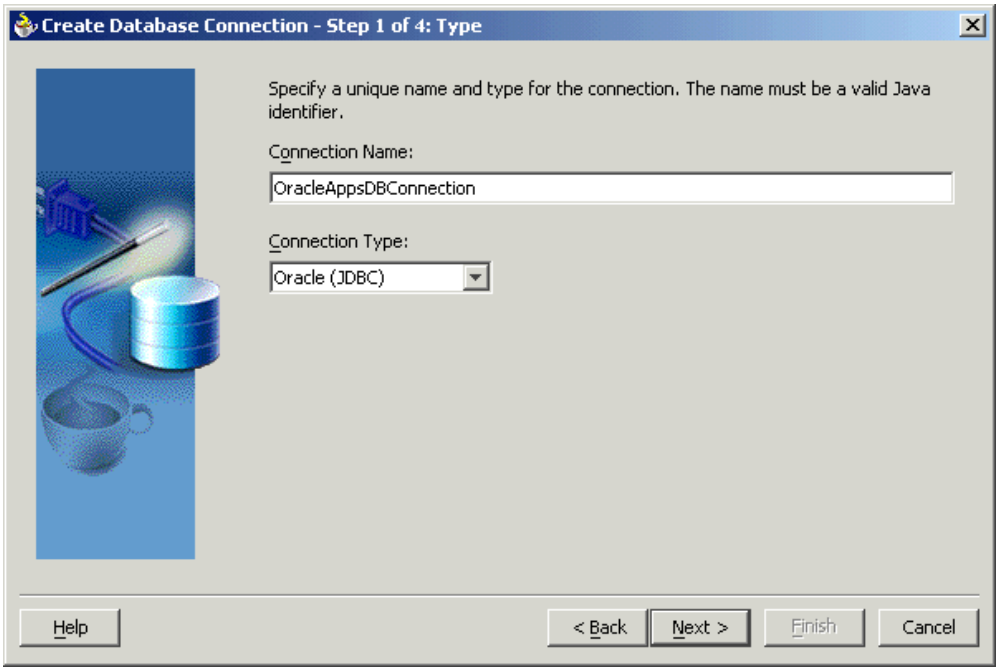
   Note: You need to connect to the database where Oracle Applications is running.

8. Enter the following information in the Type dialog box:

   1. In the **Connection Name** field, specify a unique name for the database connection.

   2. From the **Connection Type** box, select the type of connection for your database connection.

Specifying the Connection Name and Type of Connection



9. Click **Next**. The Authentication dialog box appears.

10. Enter information in the following fields:

    1. In the **UserName** field, specify a unique name for the database connection.

    2. In the **Password** field, specify a password for the database connection.

11. Click **Next**. The Connection dialog box appears.

12. Enter information in the following fields:

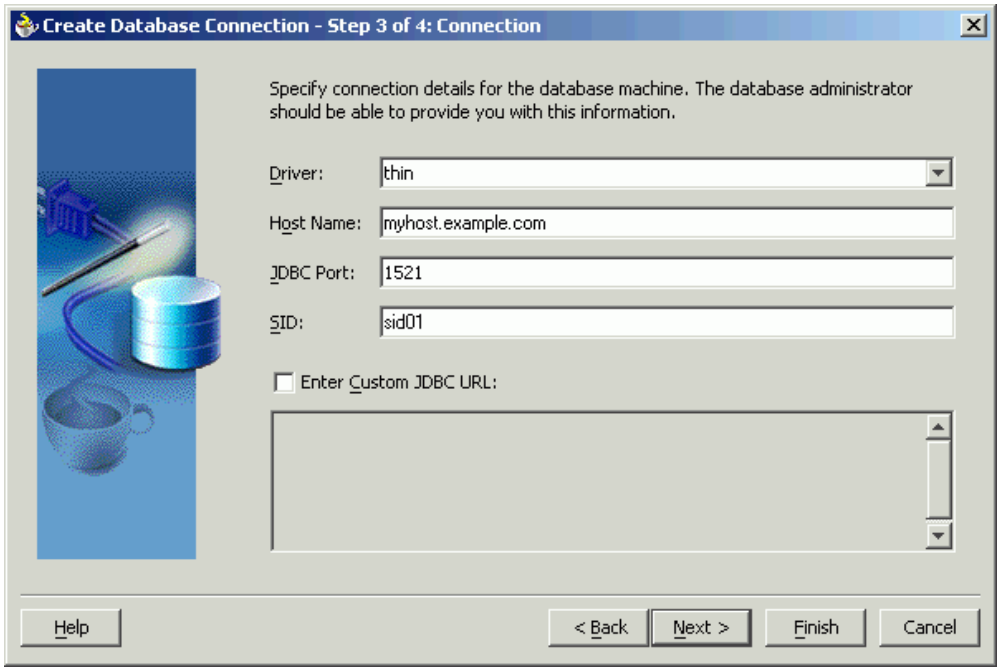
    1. From the **Driver** list, select **Thin**.

    2. In the **Host Name** field, specify the host name for the database connection.

    3. In the **JDBC Port** field, specify the port number for the database connection.

    4. In the **SID** field, specify a unique SID value for the database connection.

Specifying the New Database Connection Information



13. Click **Next**. The Test dialog appears.

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

15. Click **Next**. The Service Connection dialog box appears, providing a summary of the database connection.

16. The JNDI (Java Naming and Directory Interface) name corresponding to the database connection appears automatically in the **Database Server JNDI Name** field. Alternatively, you can specify a JNDI name.

    Note: When you specify a JNDI name, the deployment descriptor of the Oracle Applications adapter must associate this JNDI name with configuration properties required by the adapter to access the database.

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 for production.

Note: For more information about JNDI concepts, see *Oracle*

Application Server Adapter Concepts.

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

    Once you have completed creating a new connection for the service, you can add an XML Gateway map by browsing through the maps available in Oracle Applications.

18. Click **Next** in the Service Connection dialog box.

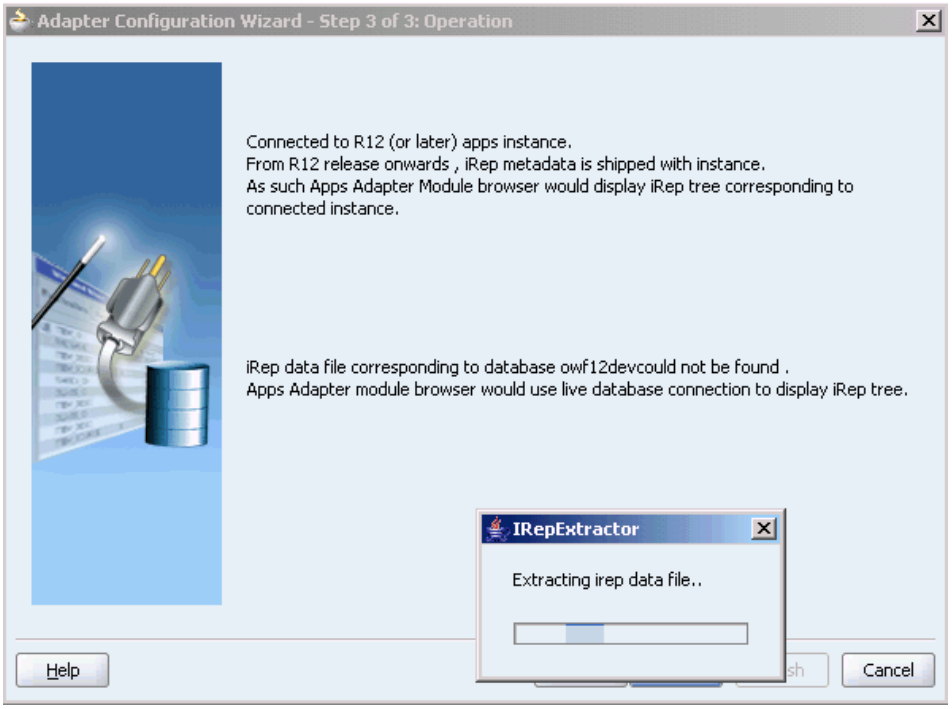
    For Oracle E-Business Suite Release 12:

    If you are connecting to Oracle E-Business Suite Release 12, then the **IREP File not present** dialog box appears indicating that OracleAS Adapter could not find the Oracle Integration Repository data file corresponding to the database you are connecting to Oracle Applications in your workspace. Absence of the data file would make browsing or searching of Integration Repository tree considerably slow. You can choose to extract the data file and create a local copy of the Integration Repository data file. Once it is created successfully, OracleAS Adapter will pick it up automatically next time and retrieve data from your local Integration Repository.

    You can select one of the following options:

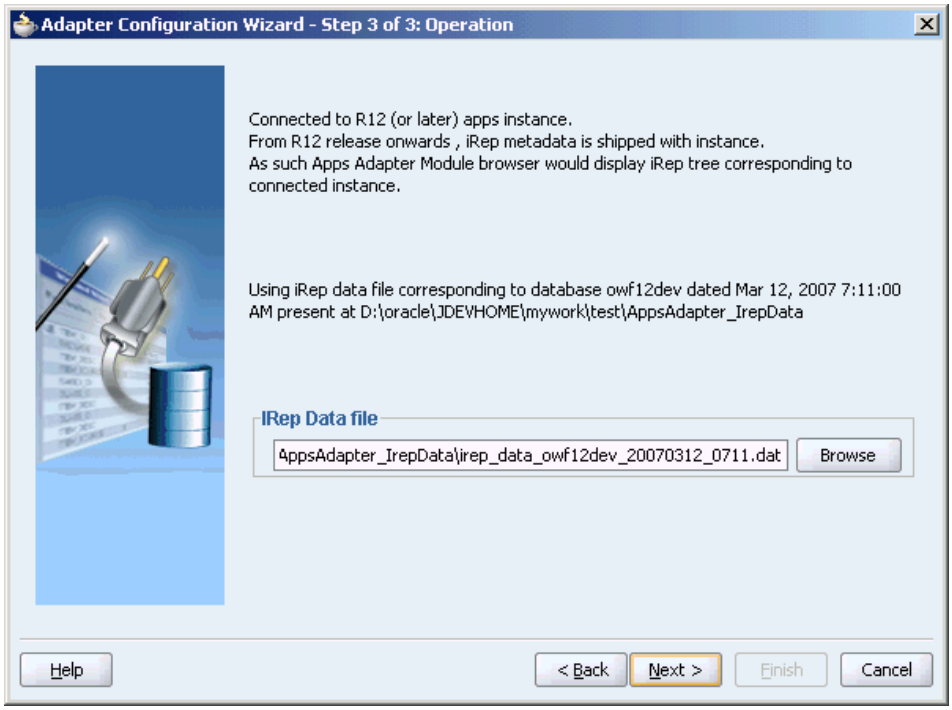
    - Click **Yes** to extract the Integration Repository data file.

Extracting Integration Repository Data File



After the system successfully creates a local copy of the Integration Repository data file, next time when you connect to the database, you will find the **IRep Data File** field appears in the Operation dialog box indicating where your local copy exists with the creation date and time as part of the file name.

Using the Local Integration Repository Data File



- Click **No** to query the Integration Repository data file from the live database you are connecting to display the Integration Repository tree.

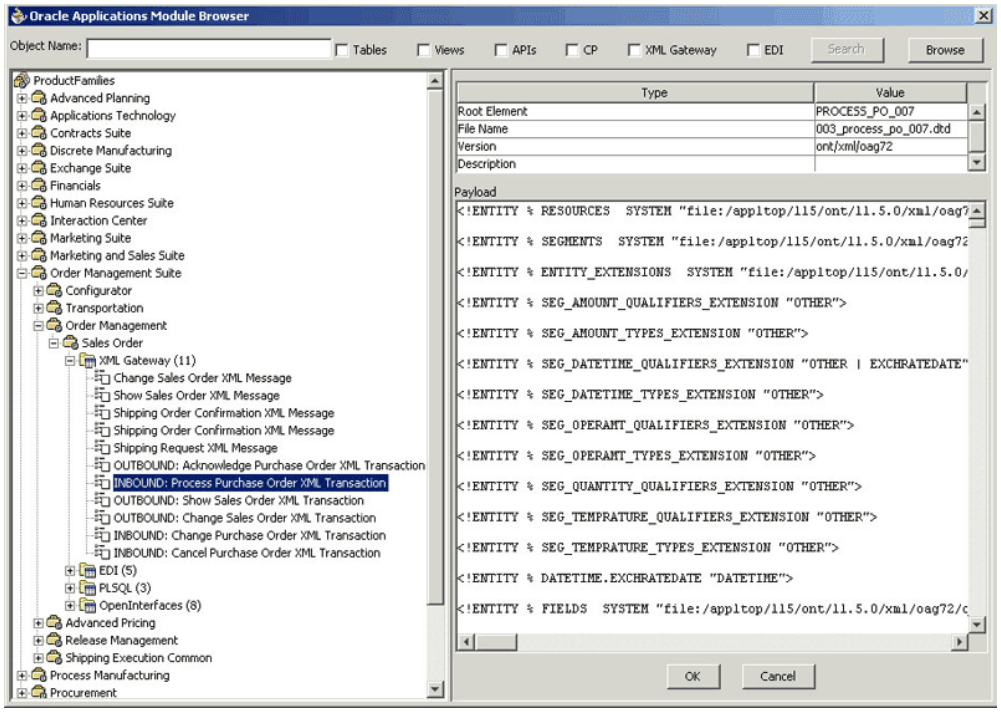
  Note: It is highly recommended that you create a local copy of the Integration Repository data file so that OracleAS Adapter will query the data next time from the local copy in your workspace to enhance the performance.

For Oracle E-Business Suite pre-Release 11.5.10:

If you are connecting to a pre-11.5.10 Oracle Applications instance, you must select the interface type in the Adapter Configuration Wizard. Select **XML Gateway** to proceed. Select **Enqueue** or **Dequeue** depending respectively on whether data is inbound into Oracle Applications or outbound from Oracle Applications. Next, choose the DTD file that is converted into an XSD schema file.

19. Click **Get Object** to open the Oracle Applications Module Browser.

Specifying the XML Gateway Message Map



Oracle Applications Module Browser includes the various product families that are available in Oracle Applications. For example, the Marketing Suite or the Order Management Suite are product families in Oracle Applications. The product families contain the individual products. For example, the Order Management Suite contains the Order Management product. The product contains the business entities associated with the product. For example, the Order Management product contains the Sales Order business entity.

Business entities contain the various application modules that are exposed for integration. These modules are grouped according to the interface they provide. XML Gateway message maps can be found under the XML Gateway category.

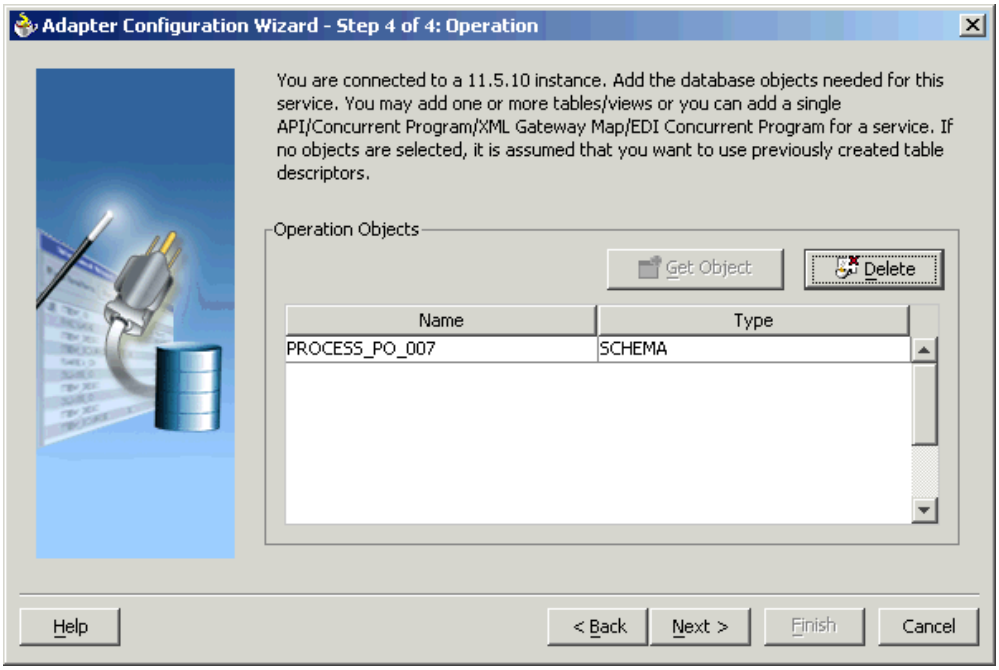
20. Select an inbound or outbound XML Gateway message map. You can select only one XML Gateway message map for each adapter service. Click **OK** to generate the XML schema.

    - You can also search for an XML Gateway message map by entering the name or part of the name for the message map in the **Object Name** field. Select the **XML Gateway** check box and click **Search**.

    - The custom message maps that you might have saved can be

found in the **Others** category.

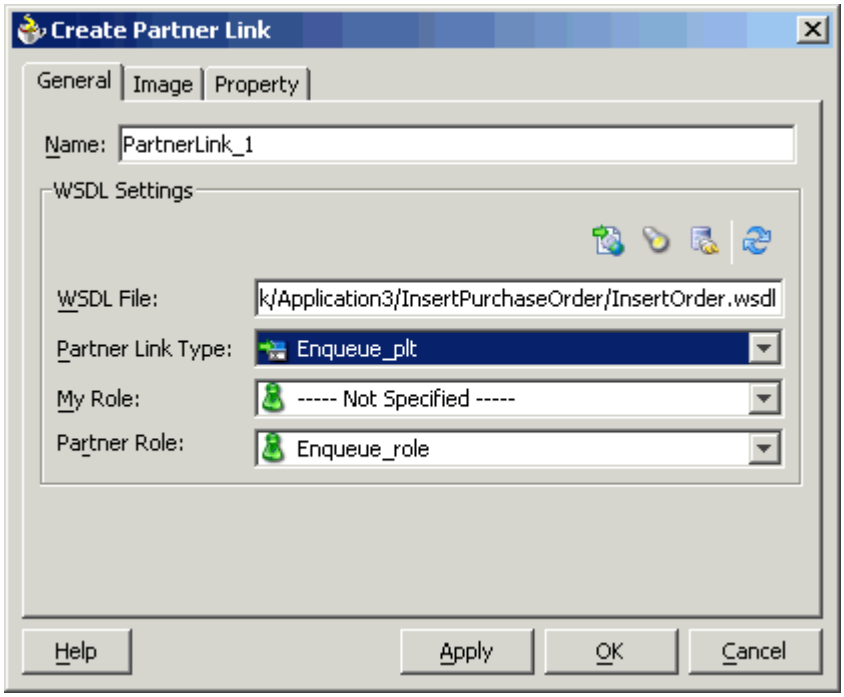
Adding the XML Schema



21. Click **Next.**

22. Click **Finish.** The wizard generates the WSDL file corresponding to the XML schema. This WSDL file is now available for the partner link.

Completing the Partner Link Configuration



23. Click **OK.** The partner link is created with the required WSDL settings.

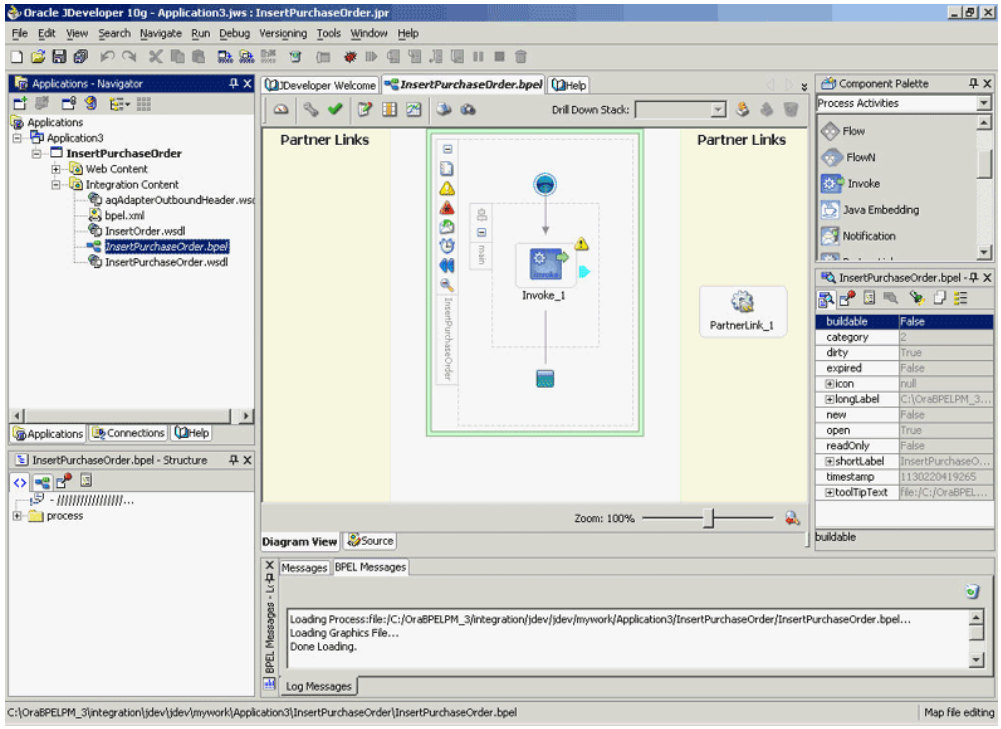
Configuring the Invoke Activity

After adding and configuring the partner link, the next task is to configure the BPEL process itself. You can start by configuring the **Invoke** process activity to enqueue the XML Gateway inbound messages.

To configure the Invoke activity:

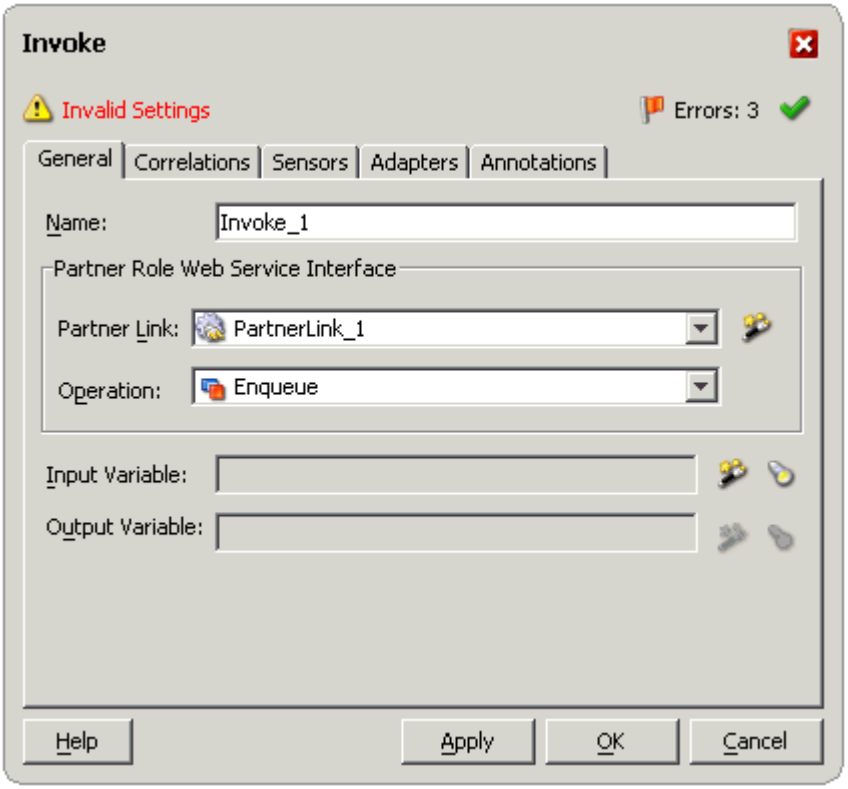
1. Drag and drop the **Invoke** activity into the process map window.

Adding the Invoke Activity



2. Double-click **Invoke** in the process map to open the Invoke dialog box. The **General** tab is selected by default.

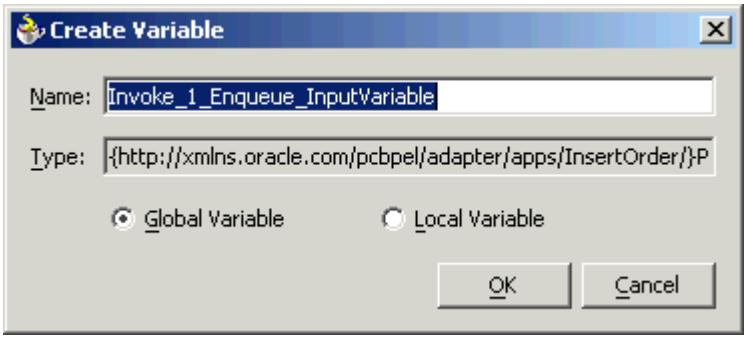
Configuring the Invoke Activity



3. In the **Partner Link** box, select the partner link to invoke. This is the partner link that you configured in the previous section. The **Operation** is automatically selected, depending on the message map that you chose when configuring the partner link. If you selected an inbound message map, then the Enqueue operation is selected.

4. Click the **Create** icon next to the **Input Variable** field. Enter a descriptive name for the variable in the Create Variable dialog box that appears. You can also accept the default name. Click **OK.**

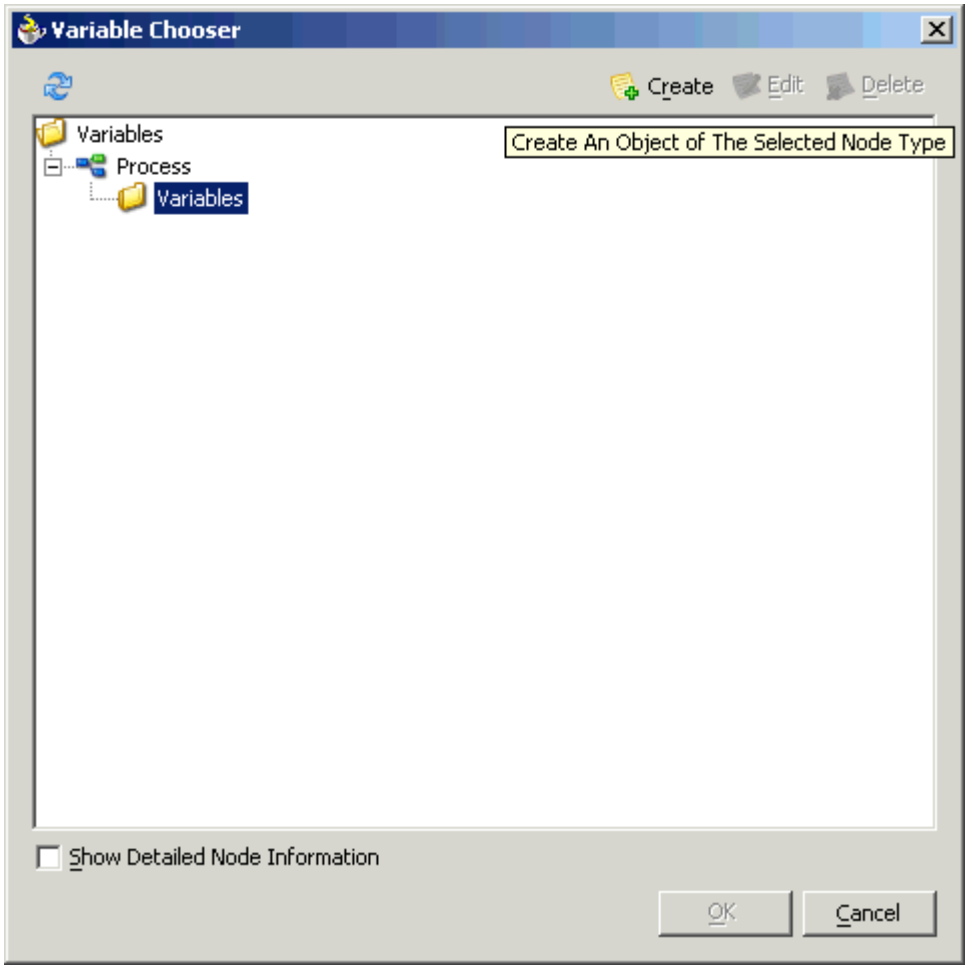
Creating the Input Variable



5. Click the **Adapters** tab in the Invoke dialog box. You need to specify the **Input Header Variable** here. Click the **Browse** icon next to the **Input Header Variable** field.

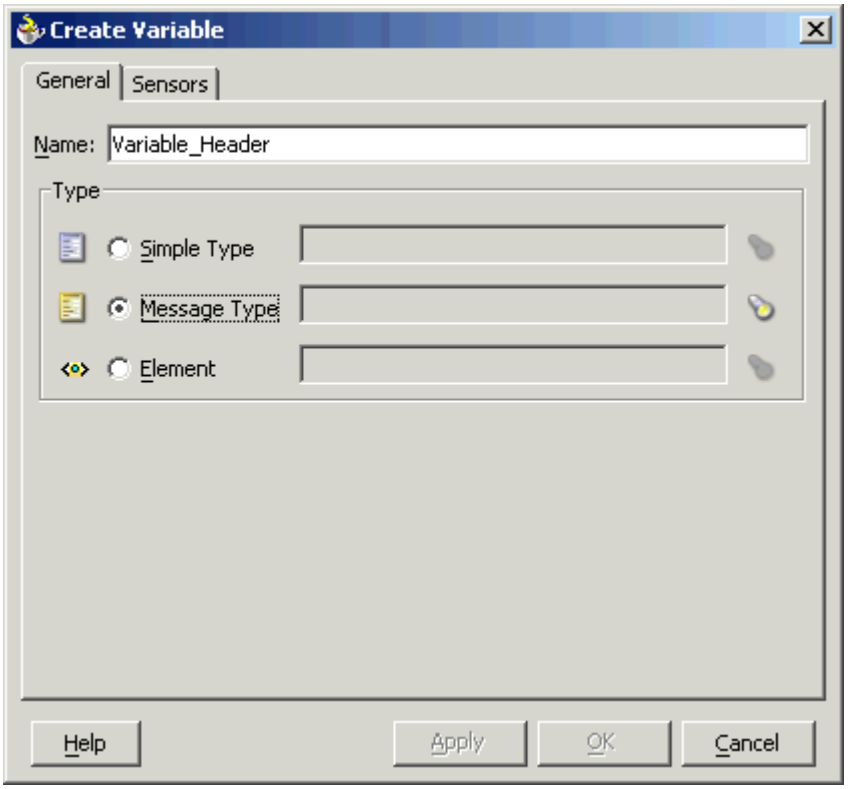
6. In the Variable Chooser dialog box that appears, select the Variables node and click **Create.**

Specifying the Input Header Variable



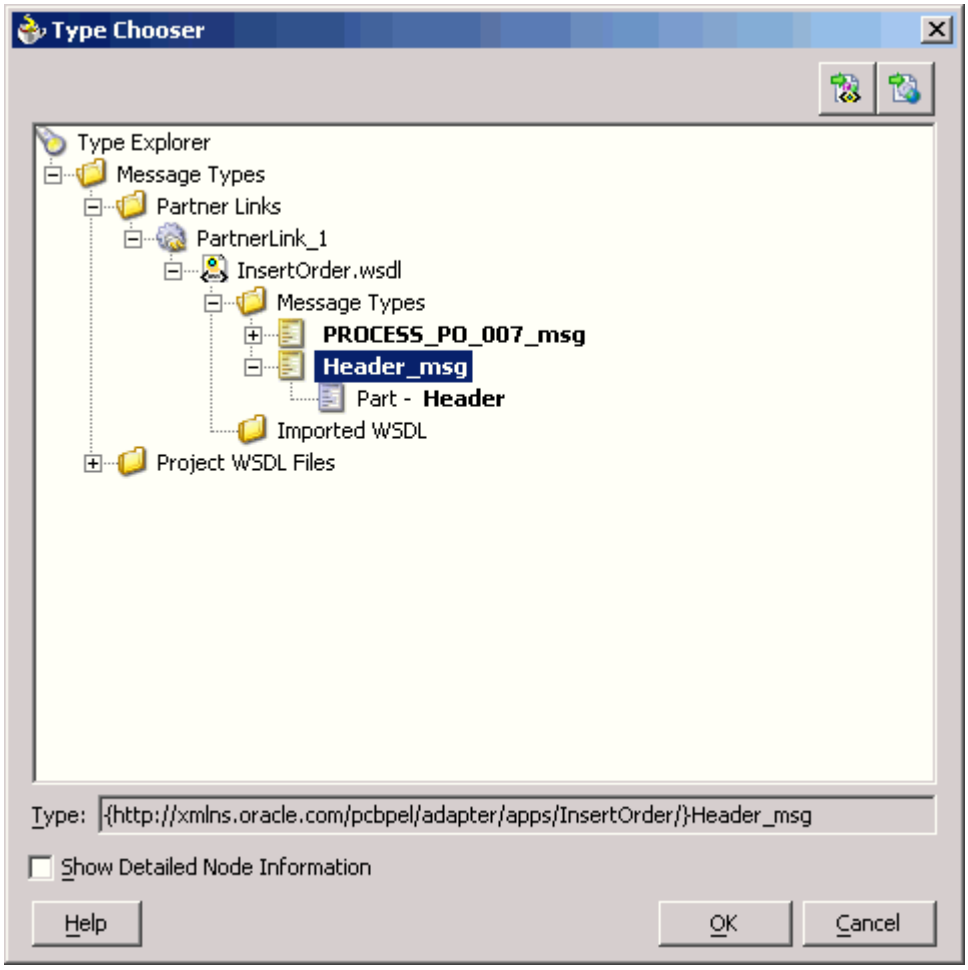
7. The **Create Variable** dialog box appears. Type a descriptive name for the variable in the **Name** field. Select the **Message Type** option and click the **Browse** icon next to it.

Creating the Input Header Variable



8. In the Type Chooser dialog box that appears, select the appropriate message type and click **OK.**

Specifying the Message Type



9. In the Create Variable dialog box, click **OK**. In the Variable Chooser dialog box, click **OK**.

10. In the Invoke dialog box, click **Apply**, then **OK.**

> **Note:** If you have configured a partner link for outbound messages from Oracle Applications, then you need to configure a **Receive** activity in place of the **Invoke** activity. The **Receive** activity is used to dequeue the XML Gateway outbound messages.

Adding a Partner Link for the File Adapter

If you are configuring an inbound message, then you would need to add another partner link for the file adapter. This allows the inbound message to pick up an XML

file received from the third party application. The data is inserted into Oracle Applications through the partner link that was configured earlier.

To add a partner link for the file adapter:

1. Drag and drop **PartnerLink** into the border area of the process diagram. The Create Partner Link dialog box appears.

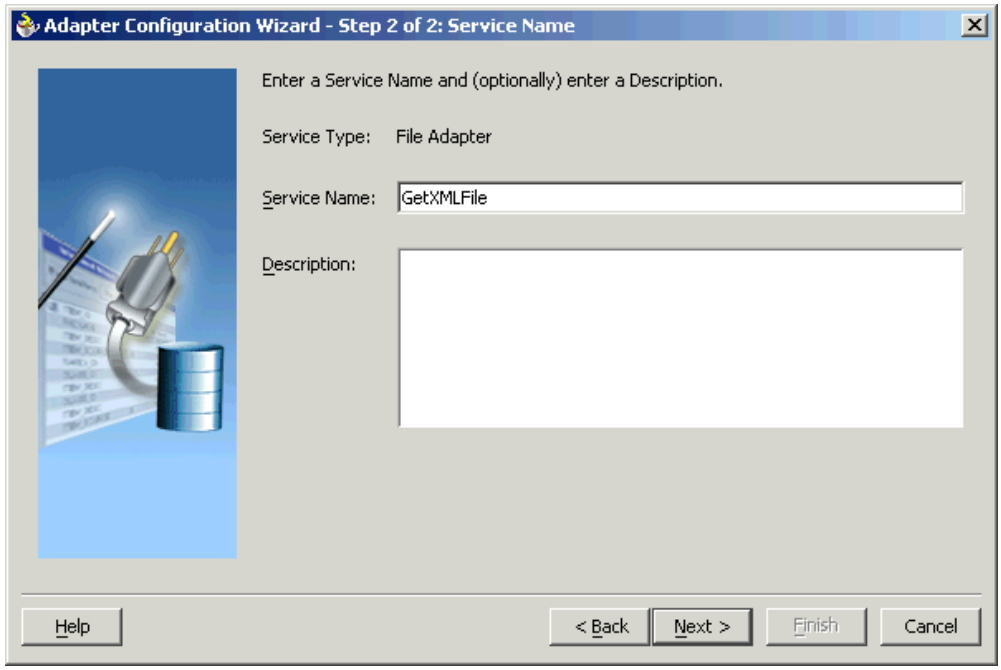
2. Click the **Define Adapter Service** icon in the WSDL Settings section. The Adapter Configuration Wizard appears.

3. Click **Next**. The Adapter Type dialog box appears.

4. Select **File Adapter** and click **Next.** The Service Name dialog box appears.

5. Enter a name for the file adapter service in the **Service Name** field. You can also enter an optional description. Click **Next.**

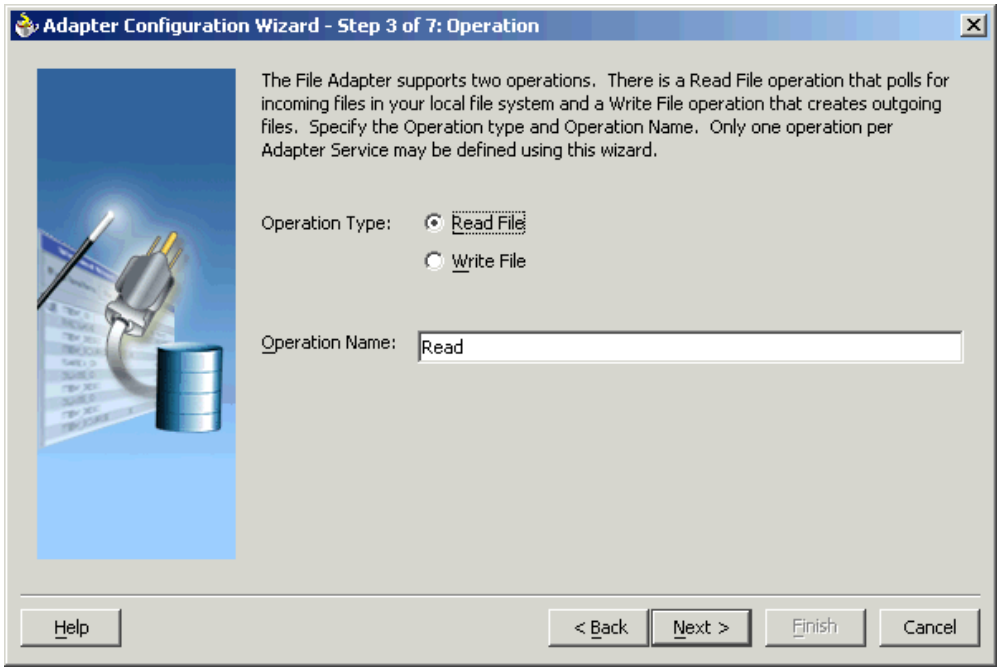
Specifying the Service Name



The Operation dialog box appears.

6. Select the **Operation Type.** The **Operation Type** specifies whether you are polling for files or creating outgoing files. Select **Read File** if you need to read files for inbound operations. Enter a name for the Operation in the **Operation Name** field. You can also use the default name. Click **Next.**

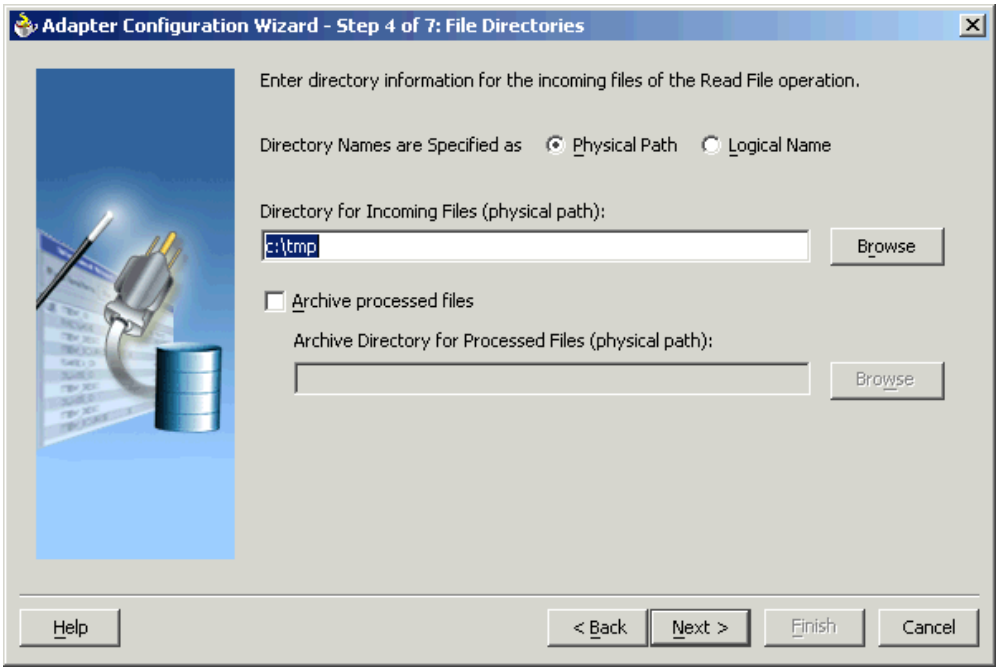
Specifying the Operation Name and Type



Note: If you are configuring the BPEL Process for outbound XML messages from Oracle Applications, you must choose **Write File** in place of **Read File**..

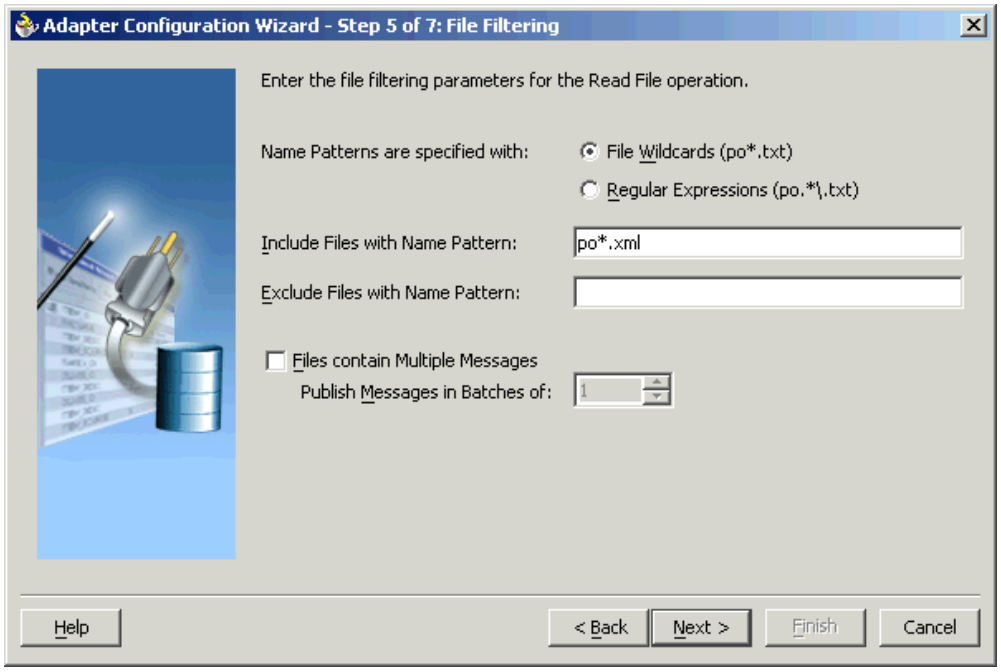
7. The File Directories dialog box appears. Enter the physical path for the XML files. You can also choose the location by using **Browse.** Click **Next.**

Specifying the Physical Path for the XML Files



8. The File Filtering dialog box appears. This enables you to specify the pattern using which you want to match the XML files that need to be picked. Specify a pattern and click **Next.**

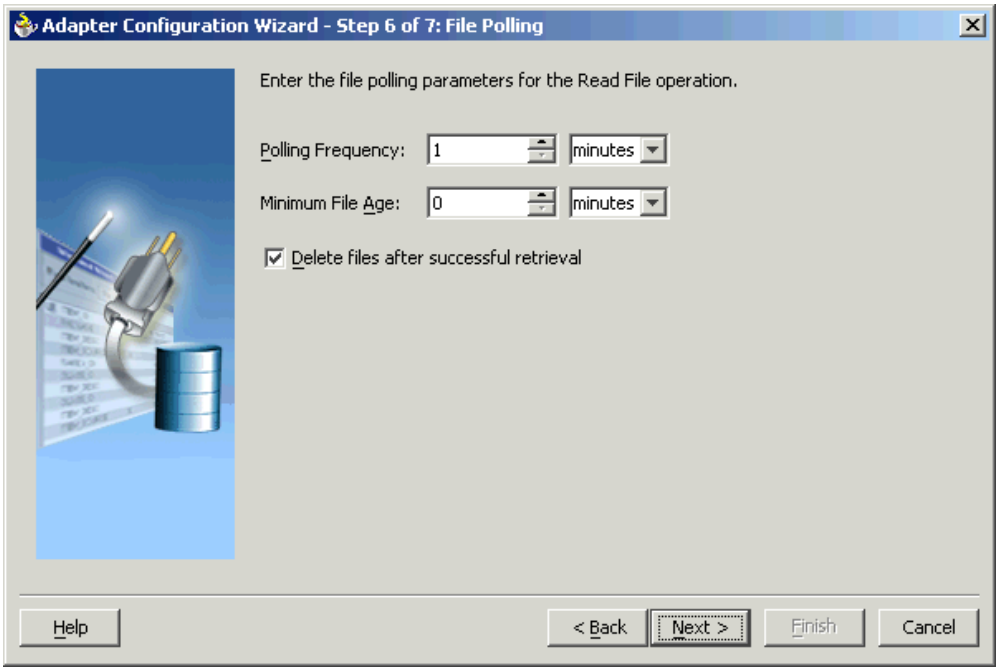
Specifying File Filtering Parameters for the Read Operation



The File Polling dialog box appears.

9. You must specify the frequency at which the directory needs to be polled for the XML files. Specify a **Polling Frequency** and click **Next.**

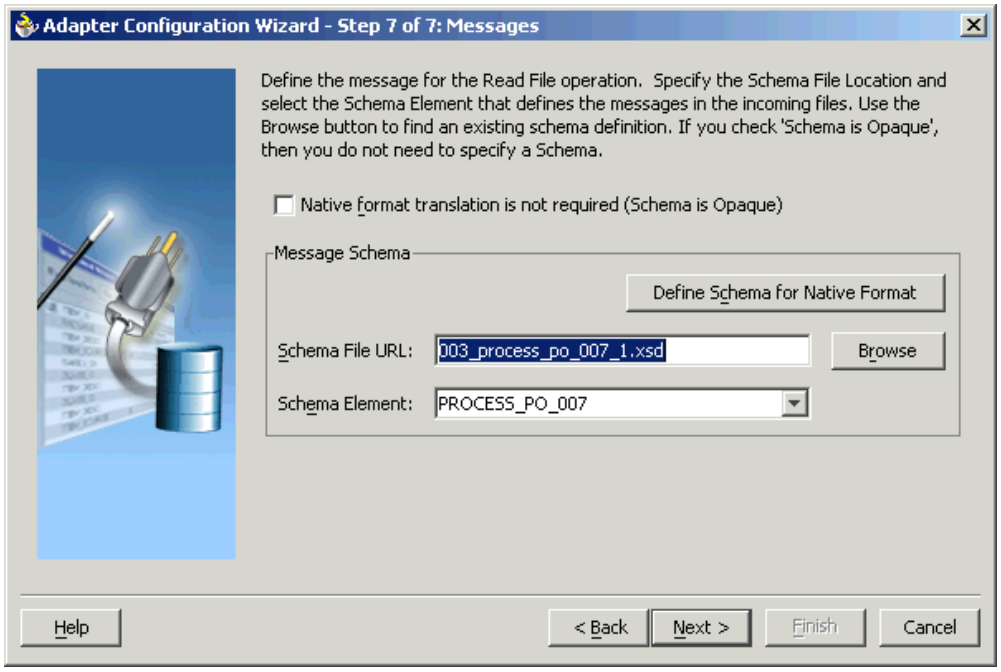
Specifying the Polling Frequency



The Messages dialog box appears.

10. You must specify the schema file location and select the schema element that defines the messages in the incoming files. You can use **Browse** to select the schema file location. Enter the required information, then click **Next.**

Specifying the Schema File Location and Schema Element



11. Click **Finish** to finish creating the File Adapter service. Click **OK** in the Create Partner Link dialog box to create the partner link for the File Adapter service.

    Note: The File Adapter is just one of the ways to read or write XML messages. The XML messages can also be exchanged using an FTP Adapter, SMTP Adapter, or through another BPEL process.

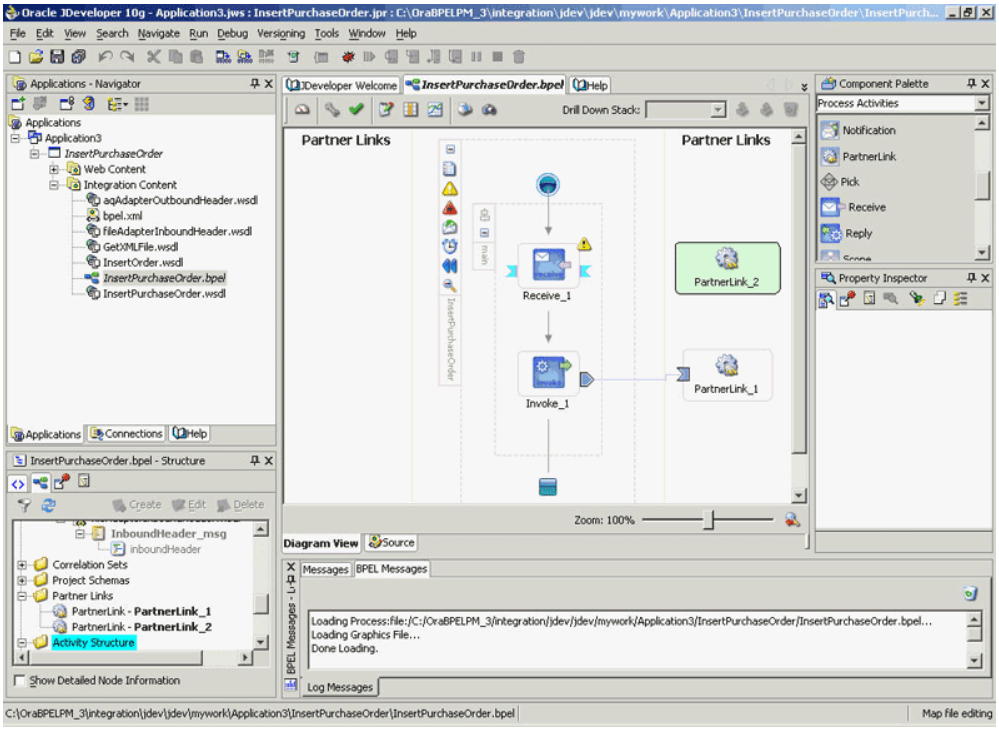
Configuring the Receive Activity

The next task is to configure a Receive activity to receive XML data from the partner link that you configured for the file adapter service.

To configure the Receive activity:

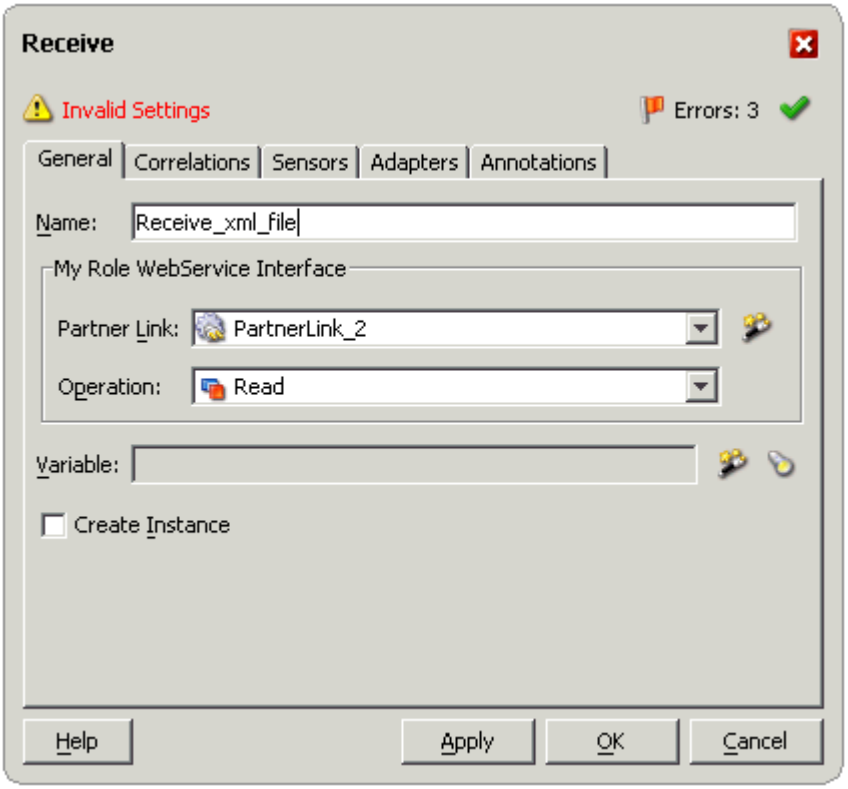
1. Drag and drop **Receive** into the process map window. The **Receive** activity should be placed in between **Start** and **Invoke.**

Adding the Receive Activity



2. Double-click **Receive** in the process map to open the Receive dialog box. The General tab is selected by default.

Configuring the Receive Activity

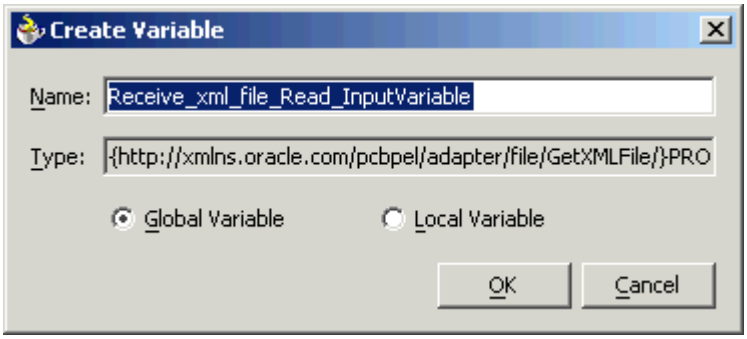


3. In the **Partner Link** box, choose the partner link corresponding to the file adapter service. The **Operation** is selected by default depending on the **Partner Link** that you select.

4. Click the **Create** icon next to the **Variable** field. The **Create Variable** dialog box appears.

5. Enter a **Name** for the variable. You can also accept the default name. Click **OK.**

Creating the Receive Variable



6. Click **Apply** in the Receive dialog box, and then click **OK.** This associates the **Receive** activity with the partner link configured for the file adapter service.

   Note: If you are configuring the process for outbound XML messages from Oracle Applications, you must link the file adapter partner link to the **Invoke** activity. The Invoke process activity invokes the file adapter interface to write the XML message to an XML file.

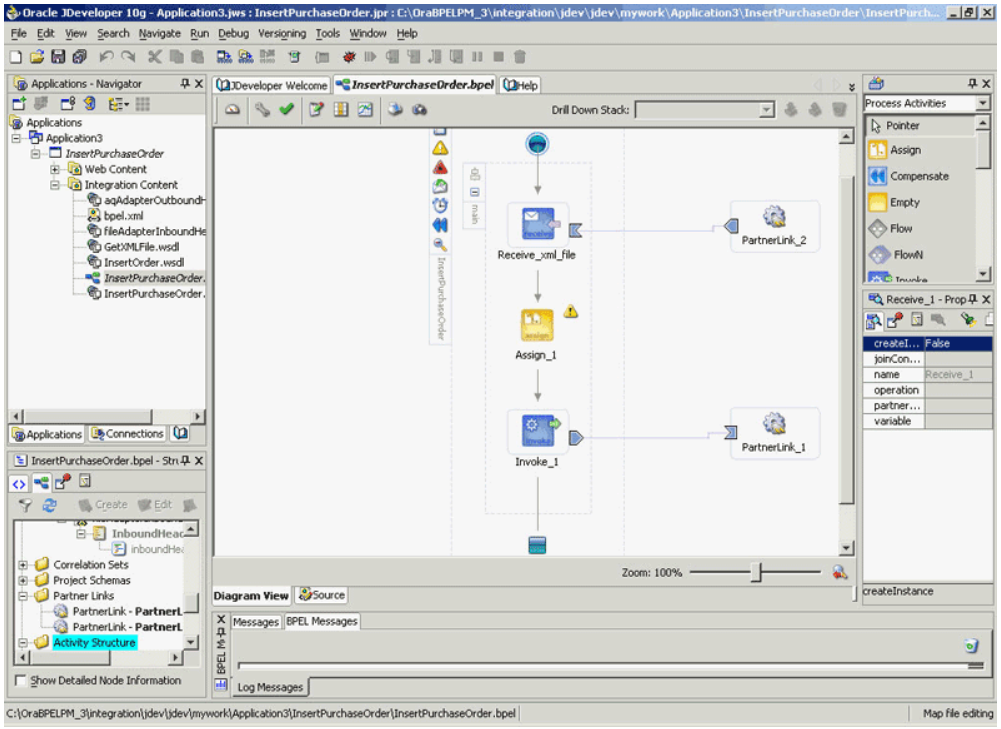
Configuring the Assign Activity

The next task is to add an Assign activity to the process map. This is used to populate parameters that provide context information to Oracle Applications.

To configure the Assign activity:

1. Drag and drop the **Assign** activity into the process map. The Assign activity needs to be dropped in between the **Receive** and **Invoke** activities.

Adding the Assign Activity

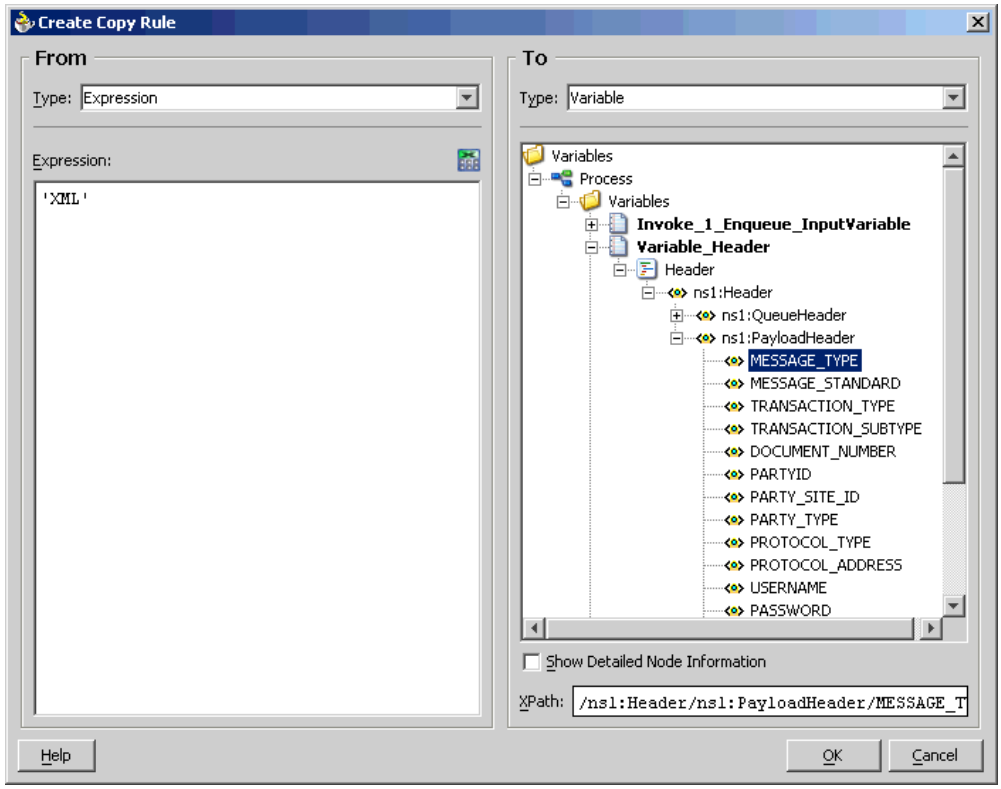


2. You now need to configure the **Assign** activity. Double-click the **Assign** activity in the process map.

3. The Assign dialog box appears. The **Copy Rules** tab appears by default. Click **Create**.

4. The Create Copy Rule dialog box appears. In the To group, expand the **Variables** node by clicking the plus sign next to it. Expand in turn the **Variable_Header, Header,ns1:Header,** and **ns1:PayloadHeader** child nodes.

Specifying Context Information for Oracle Applications



5. Select the `MESSAGE_TYPE` variable. In the From group, select `Expression` as the **Type.** Enter a message type in the **Expression** field. For example, the message type for XML Gateway is `'XML'`. Click **OK.**

6. Repeat steps 3 and 4, then select the `MESSAGE_STANDARD` variable. `MESSAGE_STANDARD` specifies the XML message standard that you are using. Enter an **Expression** for the message standard, for example, `'OAG'`. Click **OK.**

7. Repeat steps 3 and 4, then select the `TRANSACTION_TYPE` variable. This variable defines the type of transaction that you are performing, say a purchase order operation. Enter an **Expression** for the transaction type, for example, `'PO'`. Click **OK.**

8. Repeat steps 3 and 4, then select the `TRANSACTION_SUBTYPE` variable. This variable defines the subtype for the transaction that you are performing. Enter an **Expression** for the transaction subtype, for example, `'Process'`. This means that you are processing a Purchase Order transaction type. Click **OK.**

9. Repeat steps 3 and 4, then select the `PARTY_SITE_ID` variable. The party site ID identifies the trading partner. Enter an **Expression** for the party site ID. Click **OK.**

Note: `MESSAGE_TYPE`, `MESSAGE_STANDARD`, `TRANSACTION_TYPE`, `TRANSACTION_SUBTYPE`, and `PARTY_SITE_ID` are the mandatory header variables that you need to populate for the XML transaction to complete successfully.

Run-Time Tasks for XML Gateway Inbound Messaging

After designing the BPEL process, the next step is to deploy, run and monitor it.

1. Deploy the BPEL process., page 2-39
2. Test the BPEL process., page 2-41
3. Verify records in Oracle Applications., page 2-44

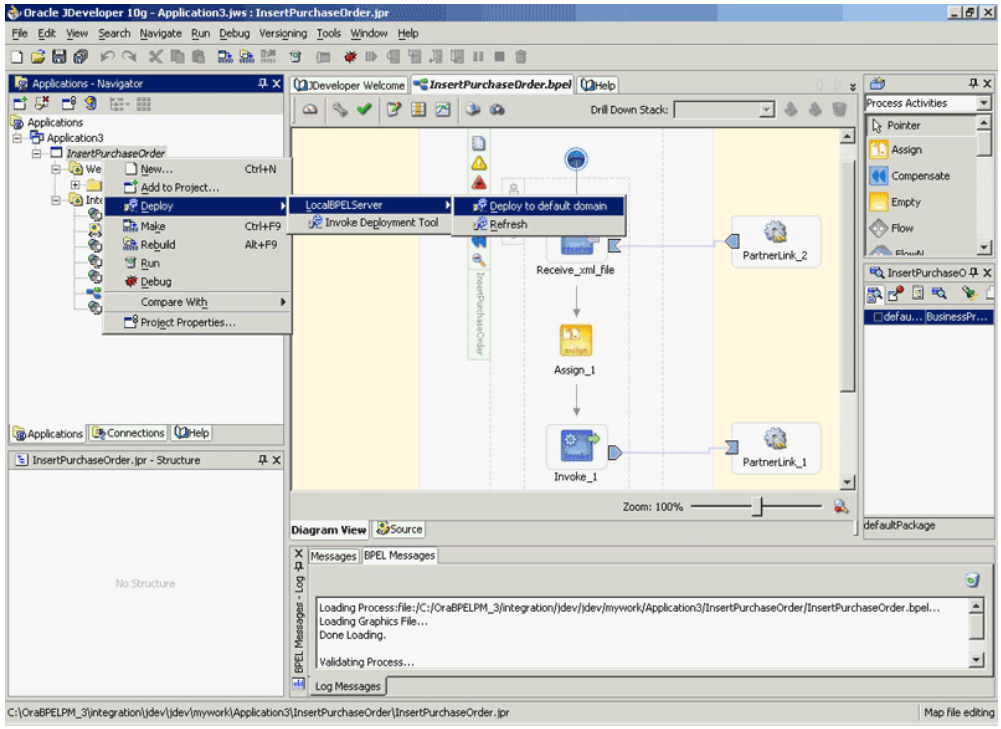
Deploying the BPEL Process

You need to deploy the BPEL process before you can run it. The BPEL process is first compiled and then deployed to the BPEL server.

To deploy the BPEL process:

1. Select the BPEL project in the Applications window.
2. Right-click the project name. Select **Deploy** from the menu that appears.
3. If you are deploying the process on the local BPEL server, select **Local BPEL Server,** then **Deploy to Default Domain.**

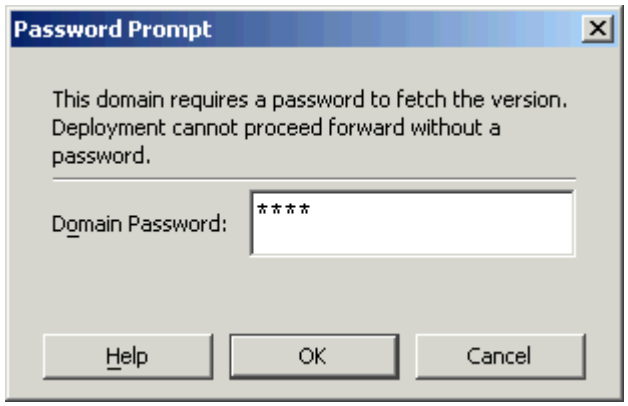
Deploying the BPEL Process



Note: You can select **Invoke Deployment Tool** if you want to deploy to a different BPEL server.

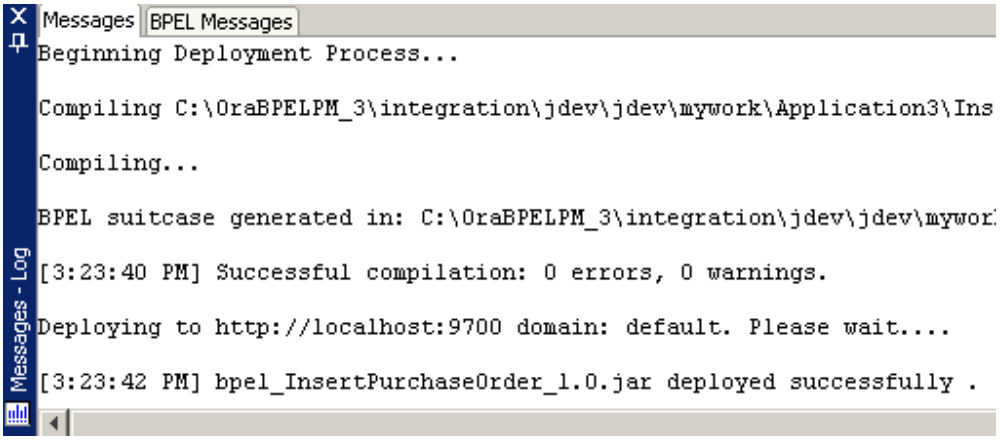
4. The Password Prompt dialog box appears. Enter the password for the default domain in the **Domain Password** field. Click **OK.**

Specifying the Domain Password



5. The BPEL process is compiled and deployed. You can check the progress in the Messages window.

Messages Window



Testing the BPEL Process

Once the BPEL process is deployed, it can be seen in the BPEL console. You can manage and monitor the process from the BPEL console. You can also test the process and the integration interface by manually initiating the process.

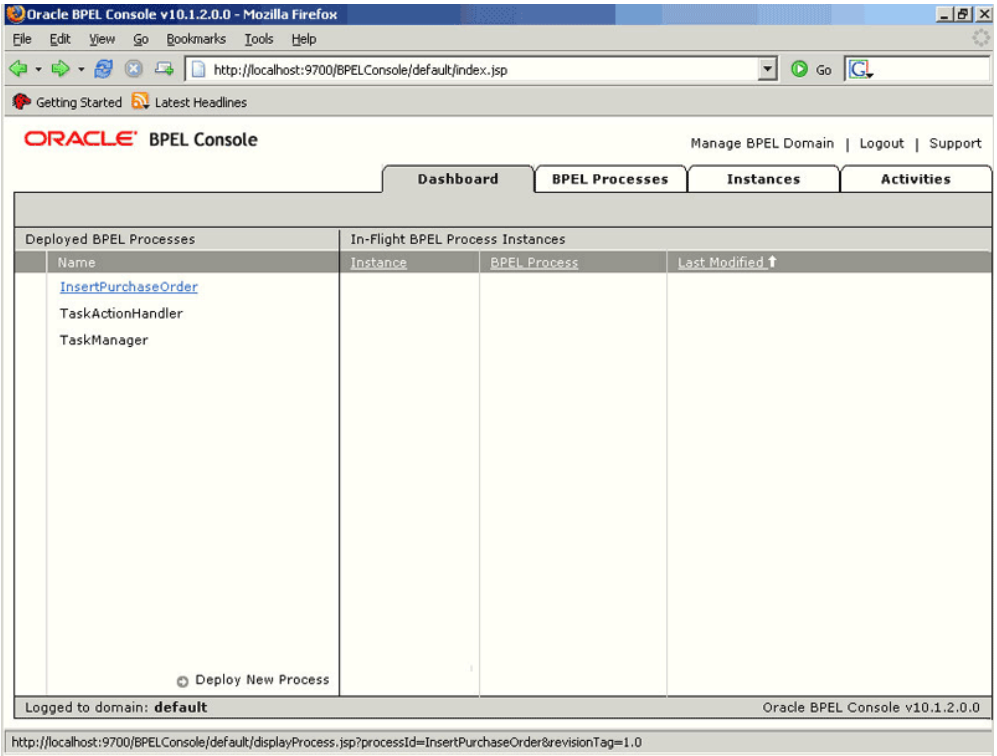
To test the BPEL process:

1. To open the BPEL console, click **Start**, and then choose **Programs**. In the Programs menu, select **Oracle -** *oracle_home*,**Oracle BPEL Process Manager 10.1.3,** and then select **BPEL Console.**

2. The BPEL console login screen appears. Select **Default** in the **Domain** box. Enter the password for the default domain in the **Password** field. Click **Login.**

3. Oracle BPEL console appears. The list of deployed processes is shown under Deployed BPEL Processes.

Deployed BPEL Processes

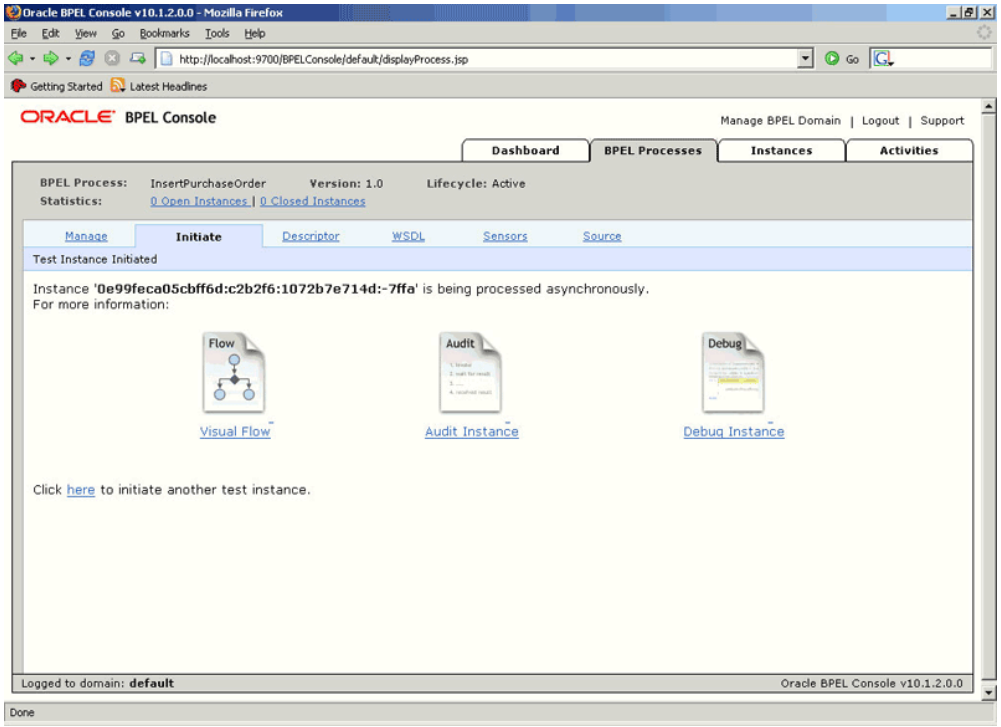


4. Click the BPEL process that you want to initiate. The Initiate page appears. Enter the input values required by the process. You can also specify an XML file for the file adapter to pick.

5. Click **Post XML Message** to initiate the process.

6. The BPEL process is now initiated. You can check the process flow by clicking the **Visual Flow** icon.

BPEL Console Initiate Page



7. The audit trail provides information on the steps that have been executed. The audit trail also records the Reference ID that is returned for the transaction. You can check the audit trail by clicking the **Audit Instance** icon.

   If the BPEL process runs into an error, then a corresponding error code is returned. The following table lists the common error codes, their descriptions, and the fix that you can use for them.

Error Codes for the BPEL Process

Code No.	Code Name	Description	Fix
12400	`APPS_MESSAGE_STANDARD_NOT_FOUND`	Message Standard not set in the header	Set the Message Standard value in the header
12401	`APPS_TRANSACTION_TYPE_NOT_FOUND`	Transaction Type not set in the header	Set the Transaction Type value in the header

Code No.	Code Name	Description	Fix
12402	APPS_TRANSACTION_SUBTYPE_NOT_FOUND	Transaction Subtype not set in the header	Set the Transaction Subtype value in the header
12403	APPS_PARTY_SITE_ID_NOT_FOUND	Party Site Id not set in the header	Set the Party Site Id value in the header
12404	APPS_MESSAGE_TYPE_NOT_FOUND	Message Type not set in the header	Set the Message Type value in the header
12406	APPS_CONTEXT_ERROR	Error in setting Apps Context	Check the username and responsibility values
12407	APPS_AUTHENTICATION_ERROR	Invalid FND username/password	Check the username and password values
12408	APPS_UNKNOWN_EX	Unknown error in Apps Interaction	Check if all the header values are valid
12409	APPS_XMLG_HEADER_NULL	XML Gateway header is null	Pass the required parameters in header

Verifying Records in Oracle Applications

You can verify all successful entries in Oracle Applications. You need to look at the relevant module in Oracle Applications to check for the new records. For example, you can look for a Sales Order entry in the Order Management module.

You can check for unsuccessful entries in corrections. You can use the Reference ID of a transaction to look for a particular record. This is the same Reference ID that you noted in the BPEL audit trail. You can reprocess a record after making the necessary configuration changes. However, you cannot alter the data in the record.

Using Transaction Monitor:

The Transaction Monitor is a tool for monitoring the status of inbound and outbound transactions originating from and going into Oracle Applications that have been

processed by the XML Gateway and delivered or received by the Oracle Transport Agent. The Transaction Monitor shows a complete history and audit trail of these documents.

You can navigate to the Transaction Monitor page using the Workflow Administrator Web responsibility. The Transaction Monitor provides the following:

- Flexible search criteria to support access to a specific document or group of documents
- Search results at the document header level with drill down by document ID
- Resend capability for outbound messages
- Viewing capability of the XML message content

> **Note:** For details on using the Transaction Monitor, see *Oracle XML Gateway User's Guide.* This guide is part of the Oracle Applications documentation library. Oracle Applications documentation can be accessed from the following link:
>
> http://www.oracle.com/technology/documentation/applications.html

Design-Time Task for XML Gateway Outbound Messaging

This section discusses the design-time steps, for an XML Gateway outbound message, that are different from the design-time steps for an inbound message.

Prerequisites to Configure XML Gateway Outbound

You need to set up a trading partner. In the trading partner setup, ensure that **Protocol Type** is set to `BPEL`.

For invoking an outbound partner link, you need to set up the correlation identifier in Oracle Applications. The correlation identifier enables you to label messages meant for a specific agent, in case there are multiple agents listening on the outbound queue. The agent listening for a particular correlation picks up the messages that match the correlation identifier for the agent.

To set up the correlation identifier:

1. Log in to Oracle Applications with the XML Gateway responsibility. The Navigator page appears.
2. Click the **XML Gateway** link.
3. Click the **Define Lookup Values** link under XML Gateway.
4. Enter `COMM_METHOD` for the **Type** field.

5. Enter `BPEL` for the **Code** field. Fill in the other mandatory fields. Oracle XML Gateway puts the correlation of BPEL when enqueueing the message on the ECX_OUTBOUND queue.

The following procedure is required to accomplish the design-time task.

1. Configure XML Gateway for outbound messaging., page 2-46

Configuring XML Gateway for Outbound Messaging

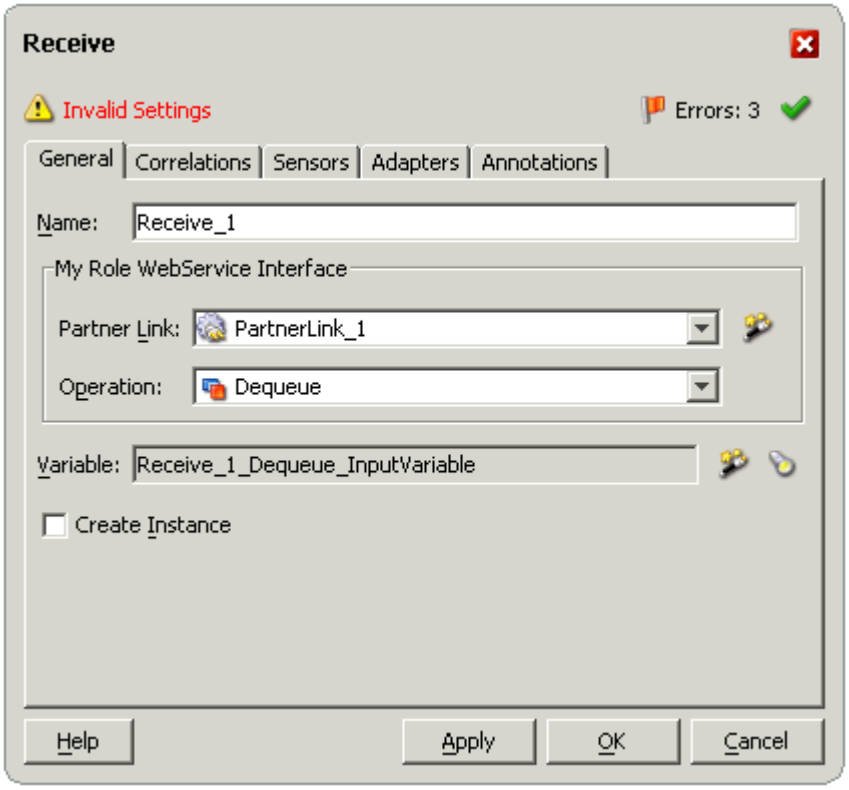
When configuring the OracleAS Adapter for Oracle Applications to use an outbound XML Gateway map, you need to configure the Receive activity for the associated partner link. The Receive activity dequeues the outbound XML messages.

To configure the Receive activity:

1. Drag and drop **Receive** into the process map window.

2. Double-click the **Receive** activity to configure the **Receive** properties. The Receive dialog box appears.

Configuring the Receive Activity

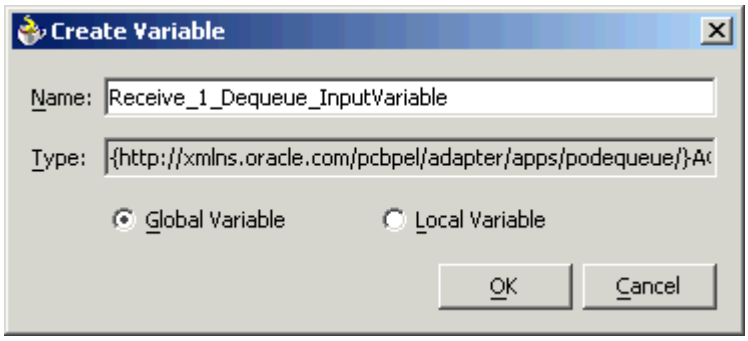


3. Select the **Partner Link** corresponding to the partner link configured with the outbound XML Gateway map.

4. The Dequeue **Operation** is automatically selected if the partner link has been configured with an outbound XML Gateway map.

5. The next task is to specify a **Variable** to receive the message data from the partner link. Click the **Create** icon to the right of the Variable field. The **Create Variable** dialog box appears.

Creating a Variable to Receive Message Data



6. Type a **Name** for the variable. You can also accept the default name. Click **OK.**

7. Click **Apply** in the Receive dialog box, then click **OK.**

> **Note:** You can define a **Header Variable** under the **Adapters** tab of the Receive dialog box. This header variable is populated with context information from the outbound XML message. Values for fields like **MESSAGE_TYPE,MESSAGE_STANDARD** and trading party information like **PARTY_SITE_ID** are returned through this variable.

Run-Time Task for XML Gateway Outbound Messaging

After designing the BPEL process, you can compile, deploy and test it.

1. Test the BPEL process., page 2-47

Testing the BPEL Process

In Oracle Applications, you can check for outbound transactions that have been processed by the XML Gateway and delivered to the Transaction Agent, by using the Transaction Monitor. You can also use the Transaction Monitor to resend an outbound

document, if necessary. Run-Time Tasks for XML Gateway Inbound Messaging, page 2-39 discusses the Transaction Monitor.

> **Note:** For details on using the Transaction Monitor, see *Oracle XML Gateway User's Guide.* This guide is a part of the Oracle Applications documentation library. Oracle Applications documentation can be accessed from the following link:
>
> http://www.oracle.com/technology/documentation/applications.html

If you have used a file adapter to write outbound messages from Oracle Applications to files, you can check the output directory location for the presence of these files after the BPEL process has run.

Troubleshooting and Debugging

If you experience problems with your Oracle XML Gateway integration, you can take the following troubleshooting steps:

- Confirm that you have the correct settings for the following elements of the trading partner setup:
  - Standard Code
  - Transaction Type
  - Transaction Subtype
  - Source Trading Partner Location Code (Party Site ID)
- Confirm that the correct transaction is enabled for the trading partner.
- Check the status of the XML transaction in Transaction Monitor.
- Ensure that the document number is unique within this transaction type.
- For inbound transactions, confirm that ECX Listeners are running.
- For outbound transactions, confirm that Background Engine is running.
- In the trading partner setup, ensure that the Protocol Type is set to `BPEL`.
- Specify the same correlation ID for Oracle Applications as for the Adapter.

  The Adapter Configuration wizard of OracleAS Adapter for Oracle Applications does not specify a correlation ID for XML Gateway transactions for inbound or outbound interfaces. Instead, a default correlation ID of `BPEL` is automatically set in the WSDL file. To make this configuration work, you must configure Oracle

Applications to set the same correlation ID value of `BPEL` for the corresponding XML Gateway transactions.

If you want the Adapter to use a different correlation ID than the default, you need to configure a correlation ID in Oracle Applications, then edit the `Correlation="BPEL"` line contained in the `<jca:operation>` section of the adapter service WSDL. Replace `BPEL` with the string value of the correlation ID you specified in Oracle Applications.

If you still experience problems with your integration, you can enable debugging.

Enabling Debugging

You can enable debugging for XML gateway using the BPEL Process Manager.

To enable debugging:

1. Log into your BPEL Process Manager domain.
2. Select *`yourdomain`*`.collaxa.cube.ws`
3. Select **Debug.**
4. Enable FND Logging to debug XML Gateway Transactions.

Debugging information is output to the log file for your domain. To examine the log file in the BPEL Process Manager, navigate to **Home > BPEL Domains >*yourdomain* > Logs.** The log file is ***yourdomain*.log**.

3

Using Business Events

This chapter covers the following topics:

- Overview of Business Events
- Design-Time Tasks for Outbound Business Events
- Creating a New BPEL Project
- Creating a Partner Link
- Configuring the Receive Activity
- Adding a Partner Link for the File Adapter
- Configuring the Invoke Activity
- Configuring the Assign Activity
- Run-Time Tasks for Outbound Business Events
- Deploying the BPEL Process
- Testing the BPEL Process
- Troubleshooting and Debugging

Overview of Business Events

The *Oracle Workflow Business Event System* (BES) is an application service that leverages the Oracle Advanced Queuing (AQ) infrastructure to communicate business events between systems. The Business Event System consists of the Event Manager and workflow process event activities.

The Event Manager contains a registry of business events, systems, named communication agents within those systems, and subscriptions indicating that an event is significant to a particular system. Events can be raised locally or received from an external system or the local system through AQ. When a local event occurs, the subscribing code is executed in the same transaction as the code that raised the event, unless the subscriptions are deferred.

Subscriptions can include the following types of processing:

- Executing custom code on the event information
- Sending event information to a workflow process
- Sending event information to other queues or systems

Business events are represented within workflow processes by event activities. By including event activities in a workflow process, you can model complex processing or routing logic for business events beyond the options of directly running a predefined function or sending the event to a predefined recipient.

Each business event represents a ready to use integration or extension point. Oracle E-Business Suite currently ships preconfigured with over 900 business events.

The uses of the Business Event System include:

- **System integration messaging hubs** - Oracle Workflow with the Business Event System can serve as a messaging hub for complex system integration scenarios. The Event Manager can be used to "hard–wire" routing between systems based on event and originator. Workflow process event activities can be used to model more advanced routing, content–based routing, transformations, error handling, and so on.
- **Distributed applications messaging** - Applications can supply Generate and Receive event message handlers for their business entities. For example, message handlers can be used to implement Master/Copy replication for distributed applications.
- **Message-based system integration** - You can set up subscriptions, which cause messages to be sent from one system to another when business events occur. In this way, you can use the Event Manager to implement point–to–point messaging integration.
- **Business–event based workflow processes** - You can develop sophisticated workflow processes that include advanced routing or processing based on the content of business events.
- **Non-invasive customization of packaged applications** - Analysts can register interesting business events for their Internet or intranet applications. Users of those applications can register subscriptions to those events to trigger custom code or workflow processes.

Business Events Concepts

Event

A business event is an occurrence in an Internet or intranet application or program that might be significant to other objects in a system or to external agents. For instance, the creation of a purchase order is an example of a business event in a purchasing application.

Event Key

A string that uniquely identifies an instance of an event. Together, the event name, event key, and event data fully communicate what occurred in the event.

Event Message

A standard Workflow structure for communicating business events, defined by the datatype WF_EVENT_T. The event message contains the event data as well as several header properties, including the event name, event key, addressing attributes, and error information.

Event Activity

A business event modeled as an activity so that it can be included in a workflow process.

Event Data

A set of additional details describing an event. The event data can be structured as an XML document. Together, the event name, event key, and event data fully communicate what occurred in the event.

Event Subscription

A registration indicating that a particular event is significant to a system and specifying the processing to perform when the triggering event occurs. Subscription processing can include calling custom code, sending the event message to a workflow process, or sending the event message to an agent.

Deferred Subscription Processing

If you do not want subscriptions for an event to be executed immediately when the event occurs, you can defer the subscriptions. In this way you can return control more quickly to the calling application and let the Event Manager execute any costly subscription processing at a later time.

Agent

An agent is a named point of communication within a system. Communication within and between systems is accomplished by sending a message from one agent to another. A single system can have several different agents representing different communication alternatives. For example, a system may have different agents to support inbound and outbound communication, communication by different protocols, different propagation frequencies, or other alternatives.

Design-Time Tasks for Outbound Business Events

OracleAS Adapter for Oracle Applications is deployed at design-time using Oracle JDeveloper and at run-time using the BPEL Process Manager.

This section discusses the process of configuring OracleAS Adapter for Oracle Applications to create business event outbound subscriptions. It describes the tasks required to configure OracleAS Adapter for Oracle Applications using the Adapter Configuration Wizard in Oracle JDeveloper.

Multiple BPEL Processes Consuming the Same Business Event

Please note that OracleAS Adapter for Oracle Applications can handle multiple BPEL processes consuming the same business event. OracleAS Adapter for Oracle Applications creates only single subscription for a particular business event regardless of the number of BPEL process consuming it. Internally, this subscription forwards business event message to a multi-consumer AQ. Since each BPEL process is an unique consumer for the event, when the message is placed in the queue, all BPEL processes are notified. Therefore, as a user you do not need to create a separate subscription for each BPEL process. All you need to do is to create the service for the event, and OracleAS Adapter for Oracle Applications will take care of message delivery to each BPEL process.

For example, if there are three BPEL processes (BPEL1, BPEL2, and BPEL3) that want to consume the same business event (such as BE1 event). For each BPEL process, you create a service for the BE1 event using OracleAS Adapter for Oracle Applications. OracleAS Adapter for Oracle Applications in turn creates a single subscription for all the three BPEL processes - BPEL1, BPEL2, and BPEL3. This subscription puts BE1 event message in multi-consumer AQ.

At run time, when a BE1 event is raised, since the subscription is applicable to all the three BPEL processes, all these three deployed BPEL processes will be activated and would receive the same BE1 event message.

Prerequisites to Configure Outbound Business Events

- The agentListener must be running on WF_Deferred queue.
- The event should be enabled for BPEL to subscribe to it. The event should not be in the disabled mode.

Following is a list of the procedures required to accomplish the design-time tasks.

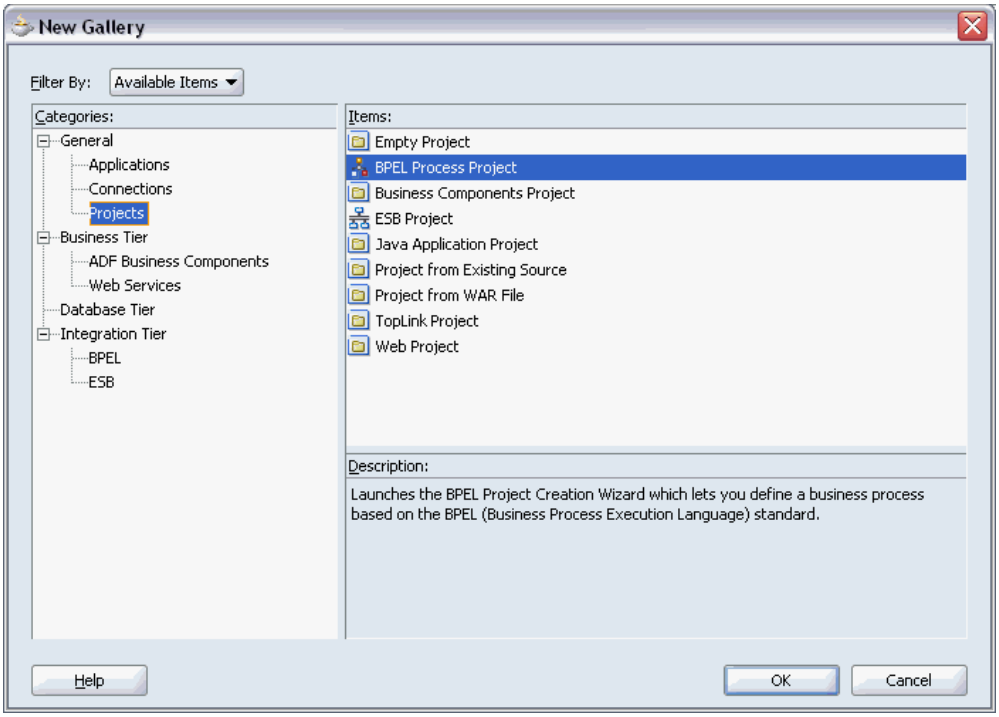
1. Create a new BPEL project, page 3-5.
2. Create a partner link, page 3-7.
3. Configure the Receive activity, page 3-22.
4. Add a partner link for the file adapter, page 3-25.
5. Configure the Invoke activity, page 3-30.
6. Configure the Assign activity, page 3-33.

Creating a New BPEL Project

To create a new BPEL project:

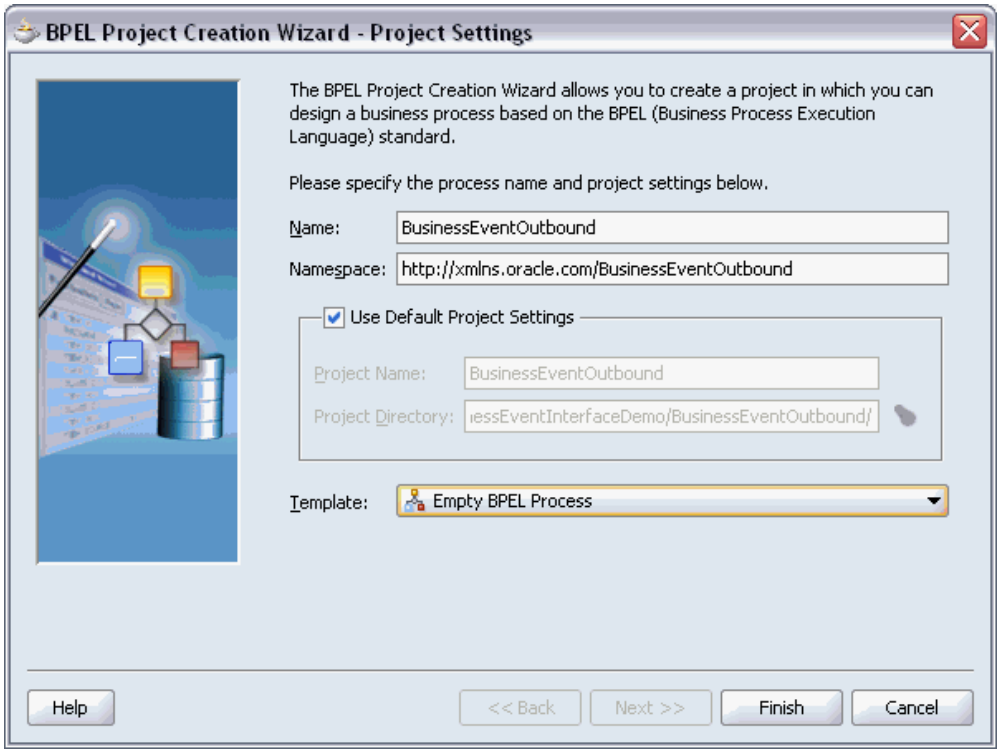
1. Open JDeveloper BPEL Designer.
2. From the **File** menu, select **New.** The New Gallery dialog box appears.

Creating a New BPEL Process Project



3. Select **All Items** from the **Filter By** box. This produces a list of available categories.
4. Expand the **General** node, then select **Projects.**
5. Select **BPEL Process Project** from the **Items** group.
6. Click **OK.** The BPEL Process Project dialog box appears.

Specifying Settings for the New BPEL Process Project



7. In the **Name** field, enter a descriptive name; for example, `BusinessEventOutbound`.

8. From the Template list, select **Empty BPEL Process,** then select **Use Default Project Settings.**

9. Click **Finish.**

   A new BPEL process is created with the required source files including bpel.xml, using the name you specified (for example, `BusinessEventOutbound.bpel`).

Creating a Partner Link

Configuring an outbound business event requires creating a partner link to allow the outbound event to be published to the Oracle BPEL Process Manager.

This task adds 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.

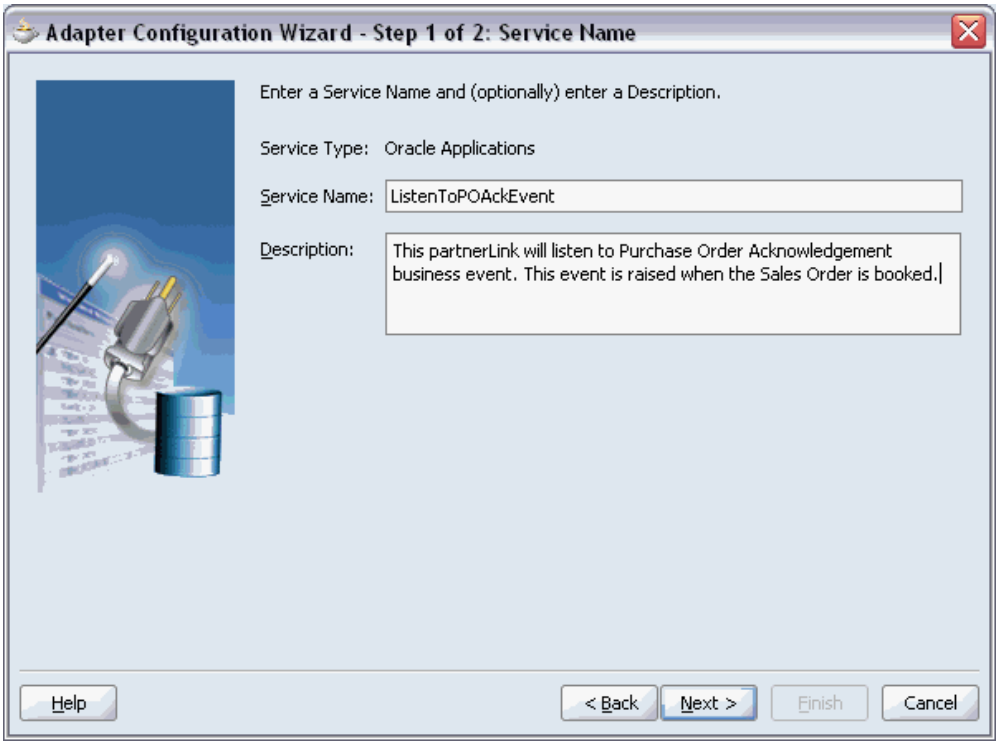
To create a partner link:

1. In JDeveloper BPEL Designer, drag and drop the **Oracle Applications** adapter

service from the Component Palette into the Partner Link border area of the process diagram. The Adapter Configuration Wizard appears.

2. Click **Next.** The Service Name dialog box appears.

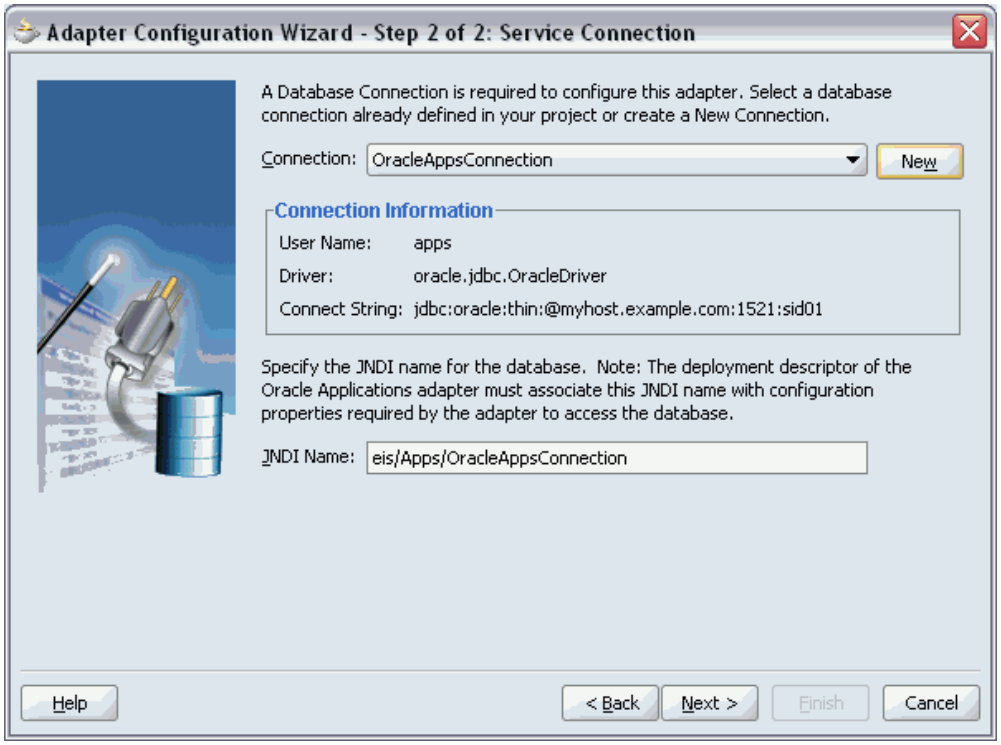
Specifying the Service Name



3. Enter a service name; for example, `ListenToPOAckEvent`. You can also add an optional description of the service.

4. Click **Next.** The Service Connection dialog box appears.

Specifying a Database Service Connection



5. Do one of the following:

   - *Use an existing database connection* - Select a database connection from the **Connection** list, and skip to Step 6 of the current procedure, page 3-13.

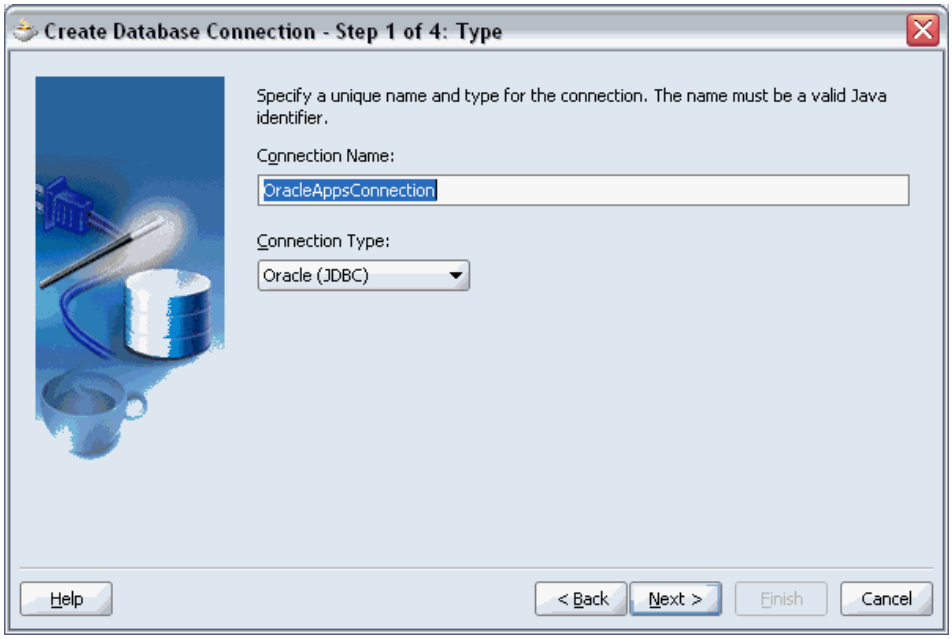
   - *Define a new database connection* - Click **New.** The Create Database Connection Wizard appears. Continue with the following procedure to create a database connection, page 3-9.

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

To create a database connection

1. In the Create Database Connection Wizard welcome dialog, click **Next.** The Type dialog box appears.

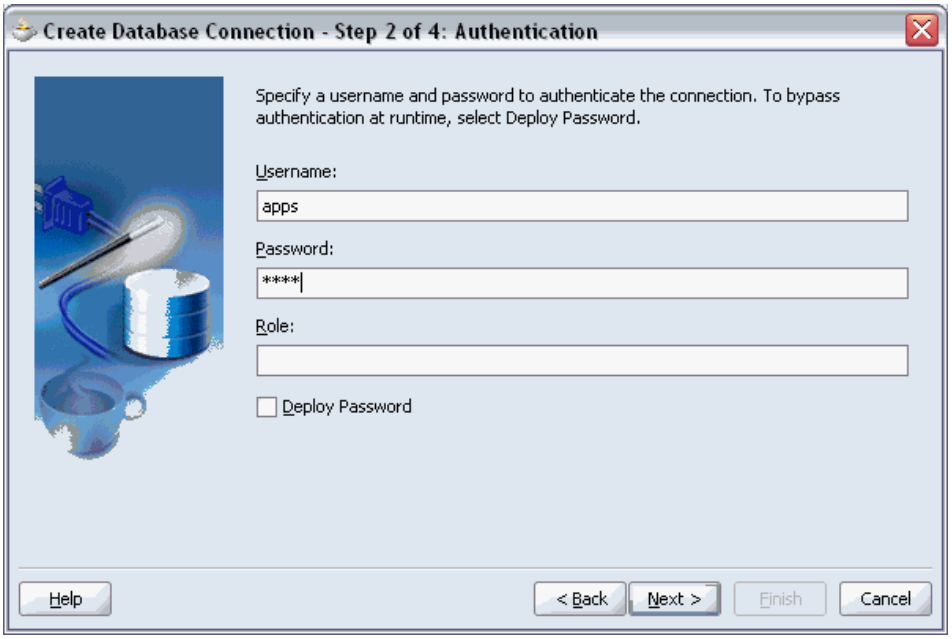
Specifying the Database Connection Type



2. Enter a unique connection name, then select a connection type for the database connection.

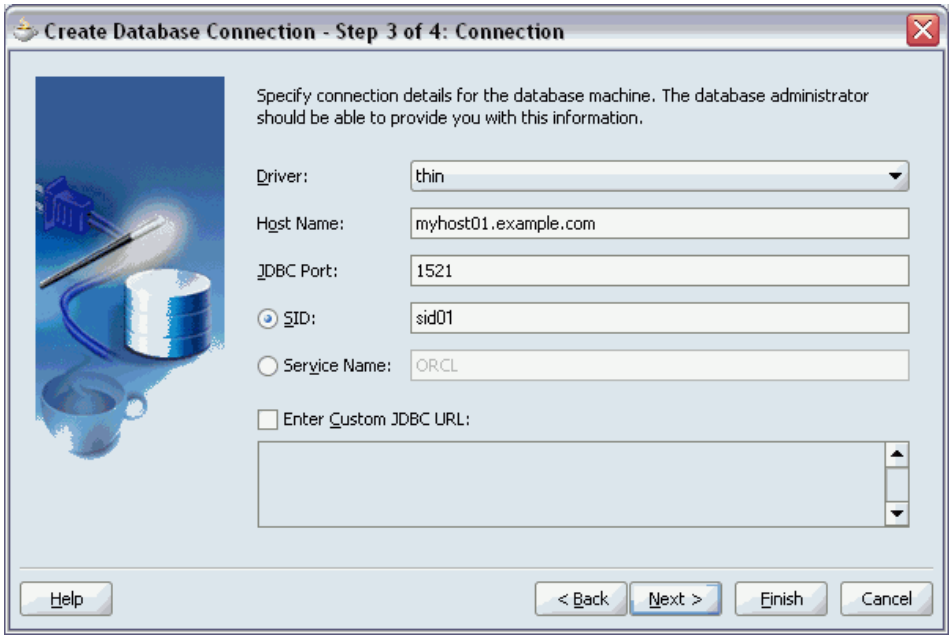
3. Click **Next.** The Authentication dialog box appears.

Authenticating the Database Connection



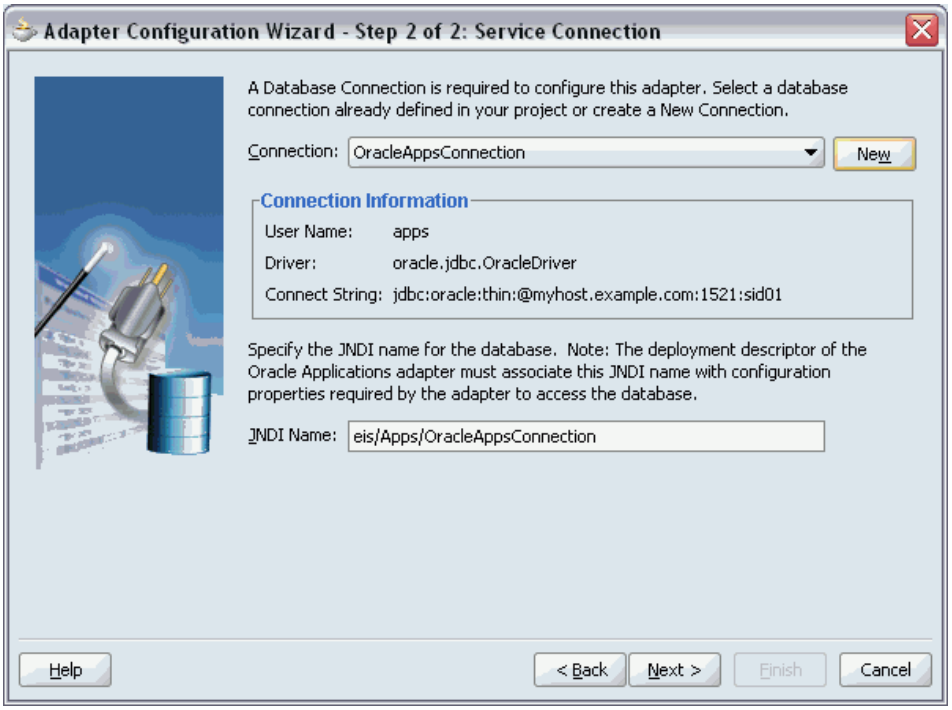
4. Enter an appropriate username and password to authenticate the database connection, then click **Next.** The Connection dialog box appears.

Providing Database Connection Details



5. Specify the following information in the Connection dialog box:
   - From the **Driver** list, select **Thin.**
   - Enter the host name for the database connection; for example, `myhost01.example.com`.
   - Enter the JDBC port number `1521` for the database connection.
   - Select **SID** and specify a unique SID value for the database connection; for example, `sid01`.
6. Click **Next.** The Test dialog box appears.
7. Click **Test Connection** to determine whether the specified information establishes a connection with the database. The status message "`Success!`" indicates a valid connection.
8. Click **Next.** The Service Connection dialog box reappears, providing a summary of the database connection.

Verifying the Database Service Connection



You can now continue with step 6 of the procedure to create a partner link.

6. The JNDI (Java Naming and Directory Interface) name corresponding to the database connection you specified appears automatically in the **JNDI Name** field of the Service Connection dialog box. Alternatively, you can enter a different JNDI name.

   Note: When you specify a JNDI name, the deployment descriptor of the Oracle Applications adapter must associate this JNDI name with configuration properties required by the adapter to access the database.

   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 for production.

7. Click **Next**.

   For Oracle E-Business Suite Release 12:

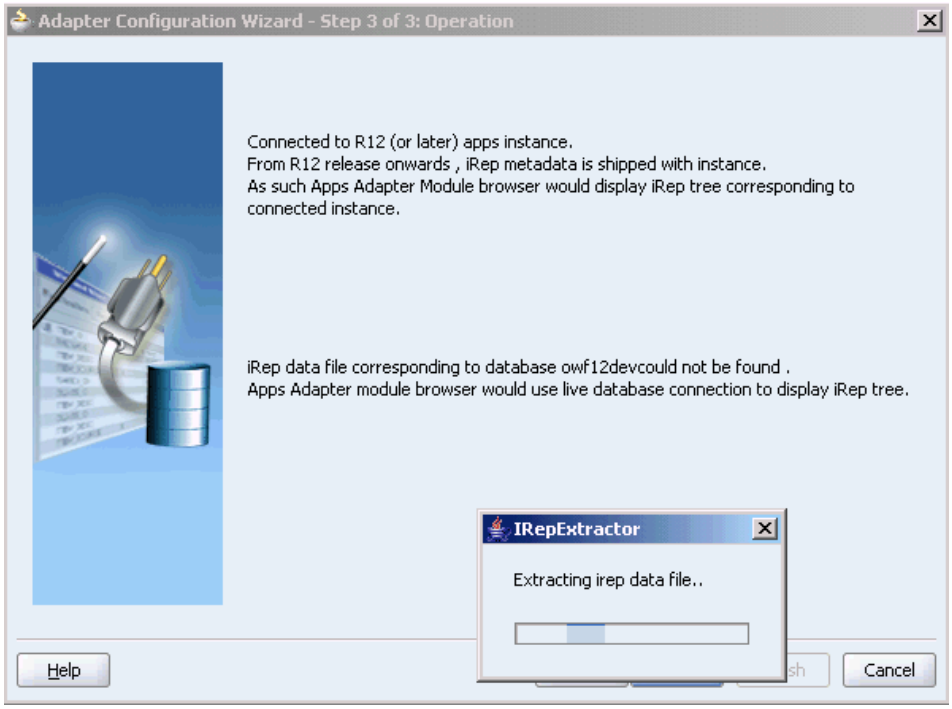
   If you are connecting to Oracle E-Business Suite Release 12, then the **IREP File not present** dialog box appears indicating that OracleAS Adapter could not find the Oracle Integration Repository data file corresponding to the database you are

connecting to Oracle Applications in your workspace. Absence of the data file would make browsing or searching of Integration Repository tree considerably slow. You can choose to extract the data file and create a local copy of the Integration Repository data file. Once it is created successfully, OracleAS Adapter will pick it up automatically next time and retrieve data from your local Integration Repository.

You can select one of the following options:

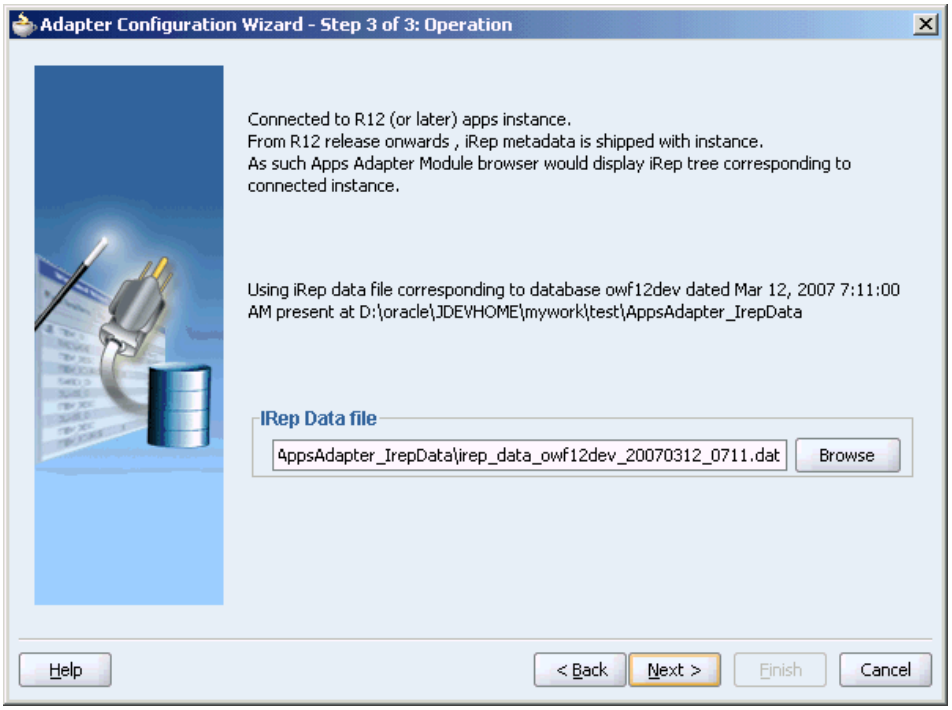
- Click **Yes** to extract the Integration Repository data file.

Extracting Integration Repository Data File



After the system successfully creates a local copy of the Integration Repository data file, next time when you connect to the database, you will find the **IRep Data File** field appears in the Operation dialog box indicating where your local copy exists with the creation date and time as part of the file name.

Using the Local Integration Repository Data File

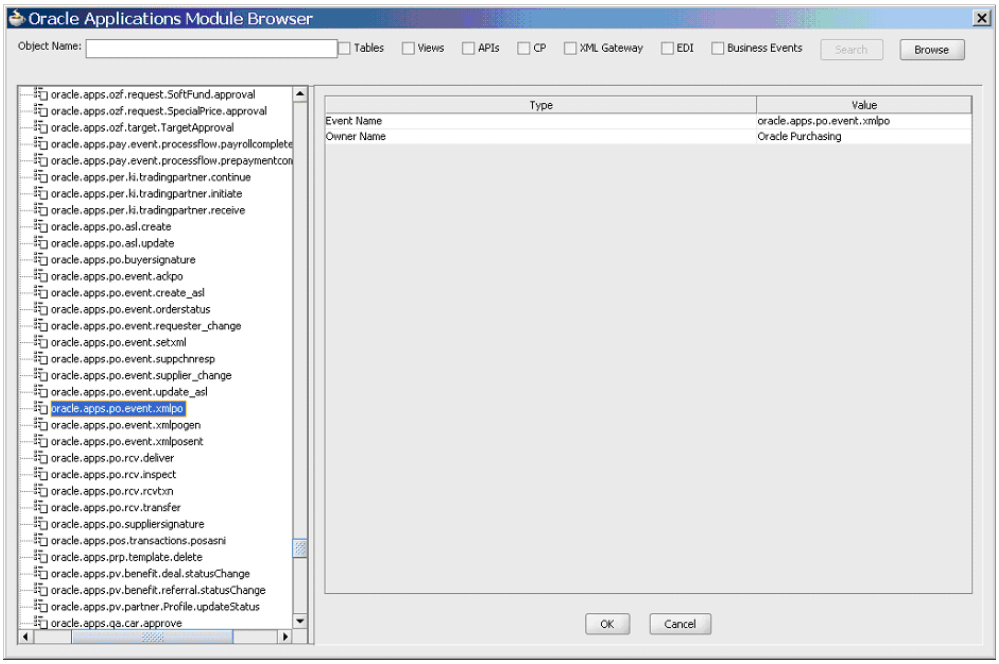


- Click **No** to query the Integration Repository data file from the live database you are connecting to display the Integration Repository tree.

  Note: It is highly recommended that you create a local copy of the Integration Repository data file so that OracleAS Adapter will query the data next time from the local copy in your workspace to enhance the performance.

8. The Oracle Applications Module Browser opens.

Oracle Applications Module Browser



- The Oracle Applications Module Browser lists the various product families that are available in Oracle Applications. For example, the Marketing Suite and the Order Management Suite are product families in Oracle Applications.

- The product families contain the individual products. For example, the Order Management Suite contains the Order Management product.

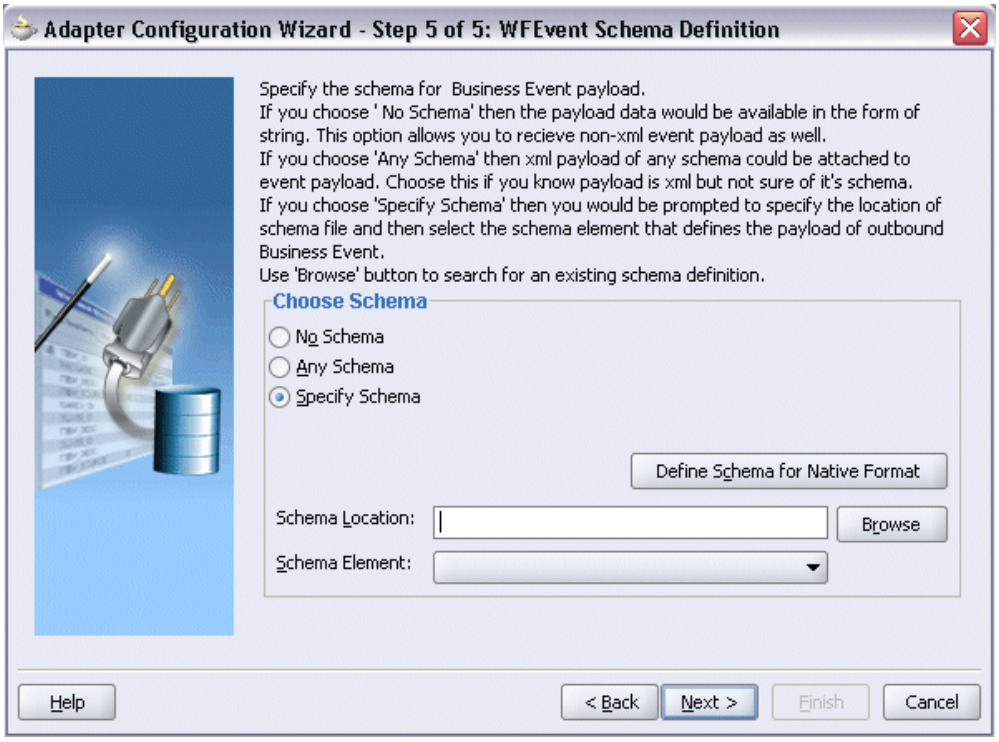
- Each product contains the business entities associated with that product. For example, the Order Management product contains the Sales Order business entity.

  Note: Business Events can be found in the Other Interfaces node, which is at the product family level. For more information, see Using the Oracle Applications Module Browser, page 1-31.

9. Expand the navigation tree to **Product Families > Other Interfaces > Business Events > Outbound.** The direction outbound is from the E-Business Suite perspective, in this case listening to business events from Oracle Applications. Select the appropriate business event, for example, **oracle.apps.po.event.xmlpo**, and click **OK.**

10. Click **Next** in the Operation dialog box. The WF Event Schema Definition dialog box for business event payload appears.

Selecting Business Event Payload Schema



11. You must specify one of the following options to be used for the business event payload:

    No Schema

    If you select the No Schema option, then the payload data would be available in the form of string. This option also allows you to receive non-XML event payload.

    Any Schema

    If you select the Any Schema option, then XML payload of any schema could be attached to event payload. You should select this option if you know the payload is XML, but not sure of its schema.

    > **Note:** When you select either the 'No Schema' or 'Any Schema' option, there is no need to further specify the schema information for your business event service, and you will proceed to the next step.

    Specify Schema

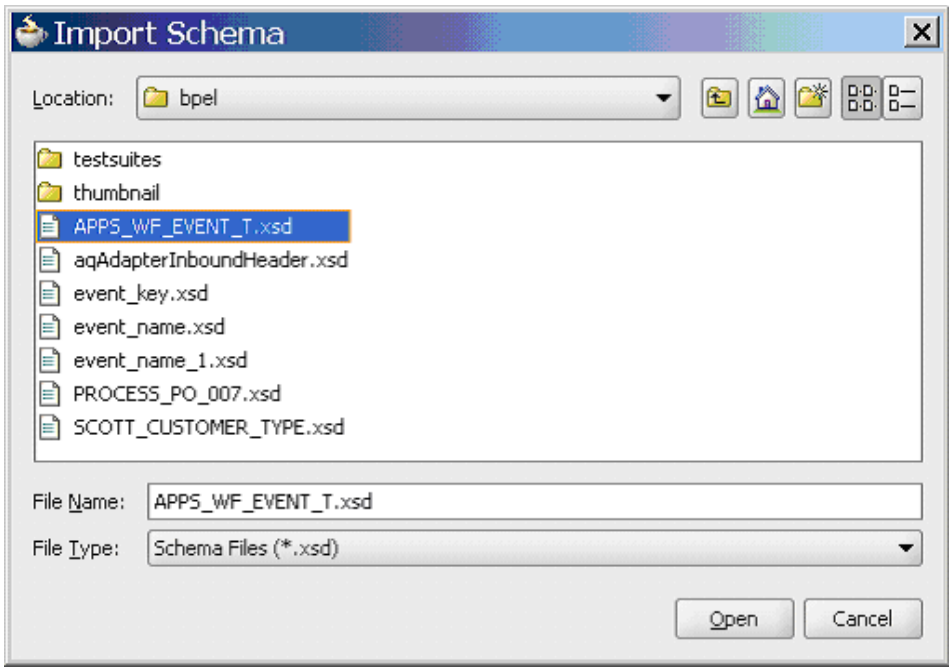
    If you select the Specify Schema option, then the Schema Location and Schema Element fields become visible. You must specify the location of schema file and then select the schema element that defines the payload of outbound business event.

To specify schema location and element

1. Click **Browse** to search for an existing schema definition in the Type Chooser.

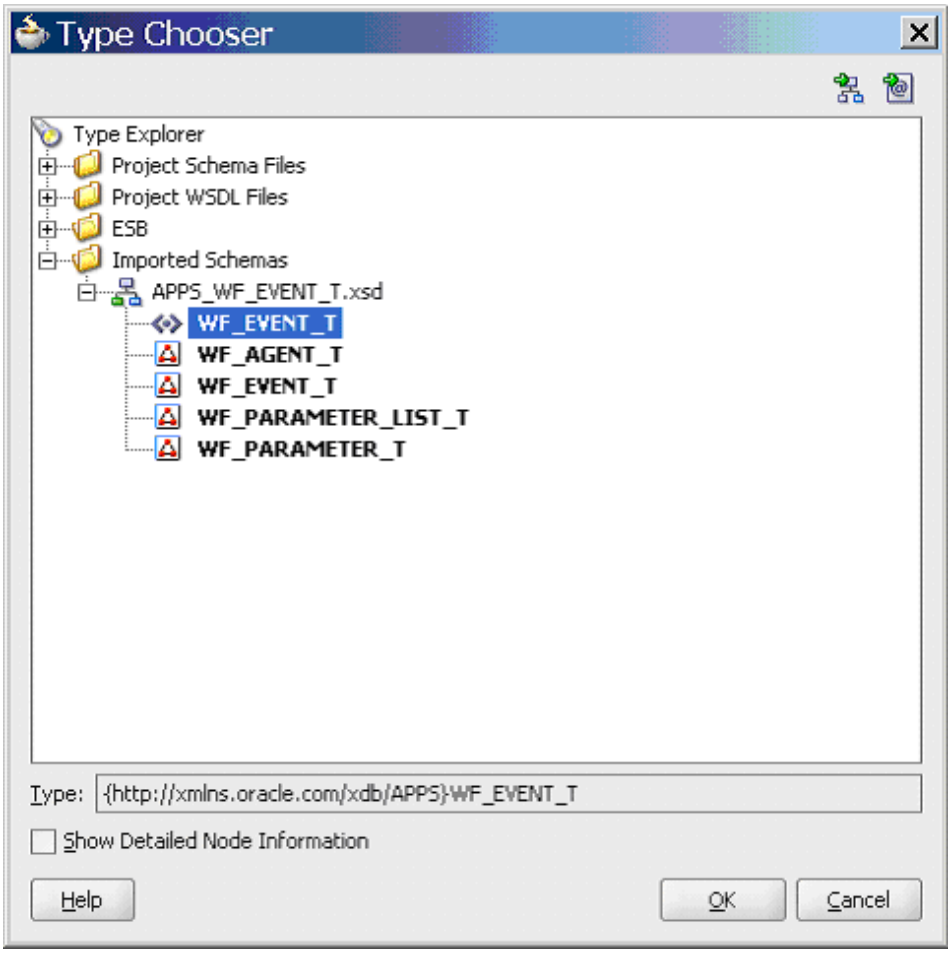
2. Click the **Import Schema File** icon at the upper right of the Type Chooser, then click the **Browse File System** icon in the Import Schema File dialog box.

Selecting a Schema File



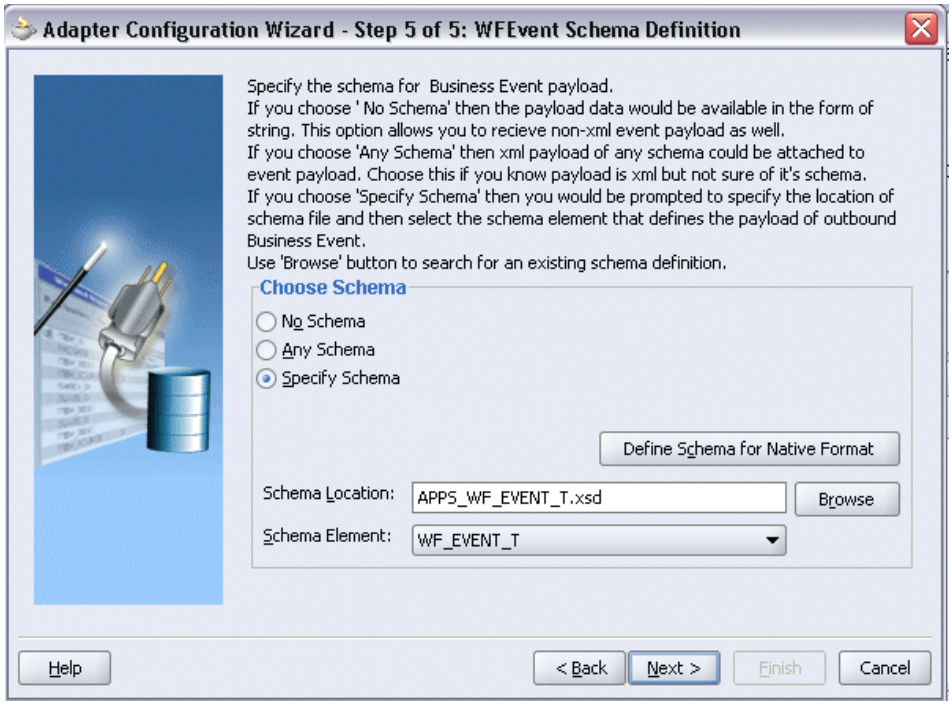
3. In the Import Schema dialog box, navigate to the schema file APPS_WF_EVENT_T.xsd and open it. The Type Chooser reappears with the selected schema in the **Imported Schemas** section.

Choosing the Schema



4. Select the schema element **WF_EVENT_T** for the business event and click **OK.** The WF Event Schema Definition dialog box reappears with your selected schema location and element information populated. Click **Next**.

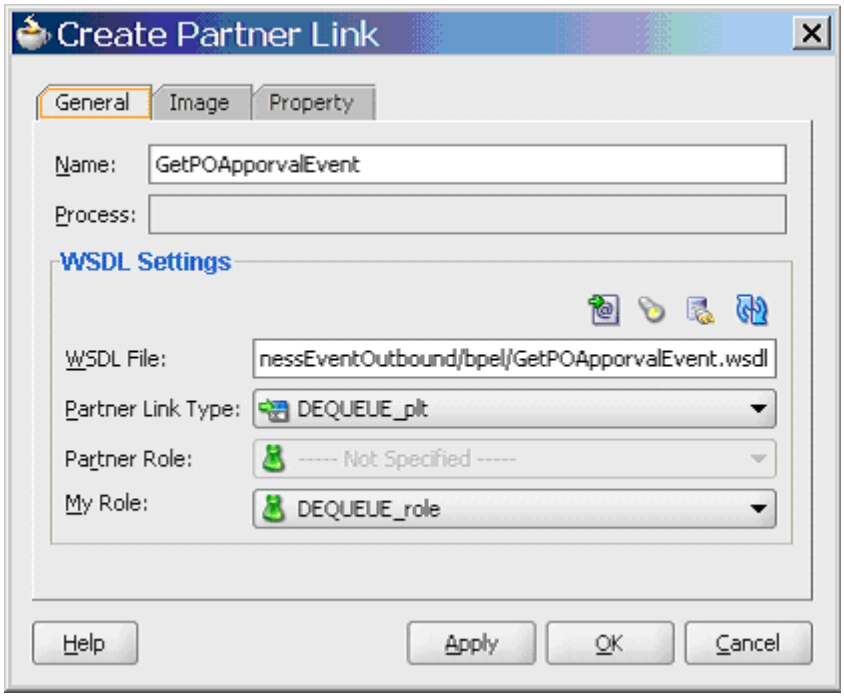
Populating Selected Business Event Payload Schema



12. The Finish dialog box appears indicating that you have finished defining the business event service. The wizard generates the GetPOApprovalEvent WSDL file corresponding to the **oracle.apps.po.event.xmlpo** business event service.

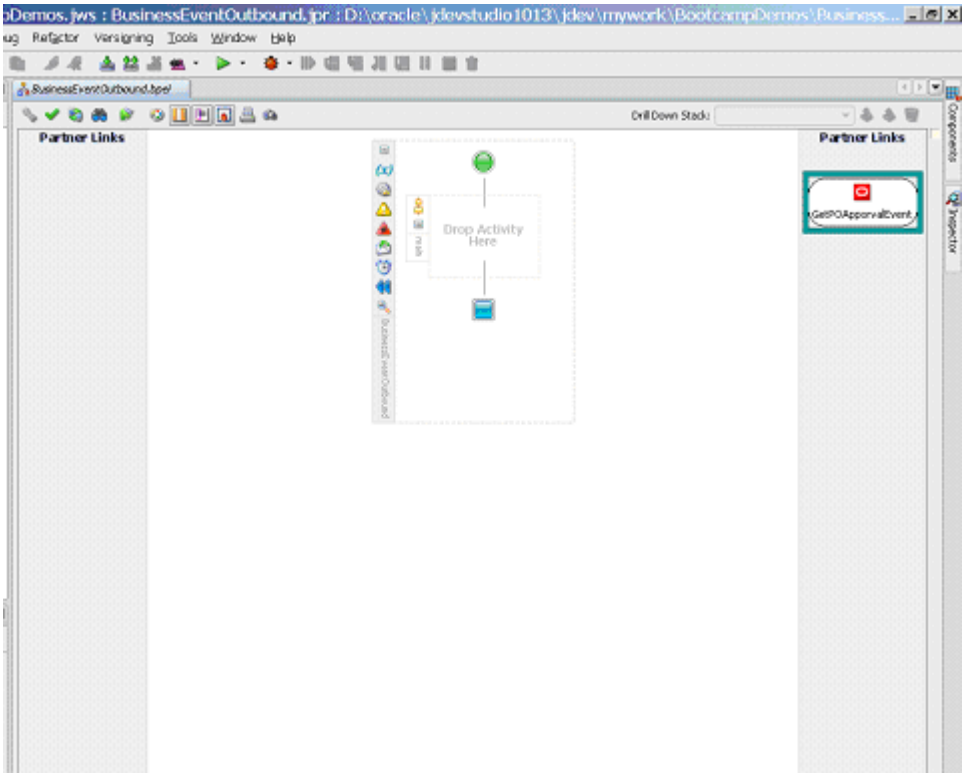
    The main Create Partner Link dialog box appears, specifying the new WSDL file.

Completing the Partner Link Configuration



13. Click **OK** to complete the partner link configuration. The partner link is created with the required WSDL settings, and is represented in the BPEL project by a new icon in the border area of the process diagram.

BPEL Project with an Applications Partner Link



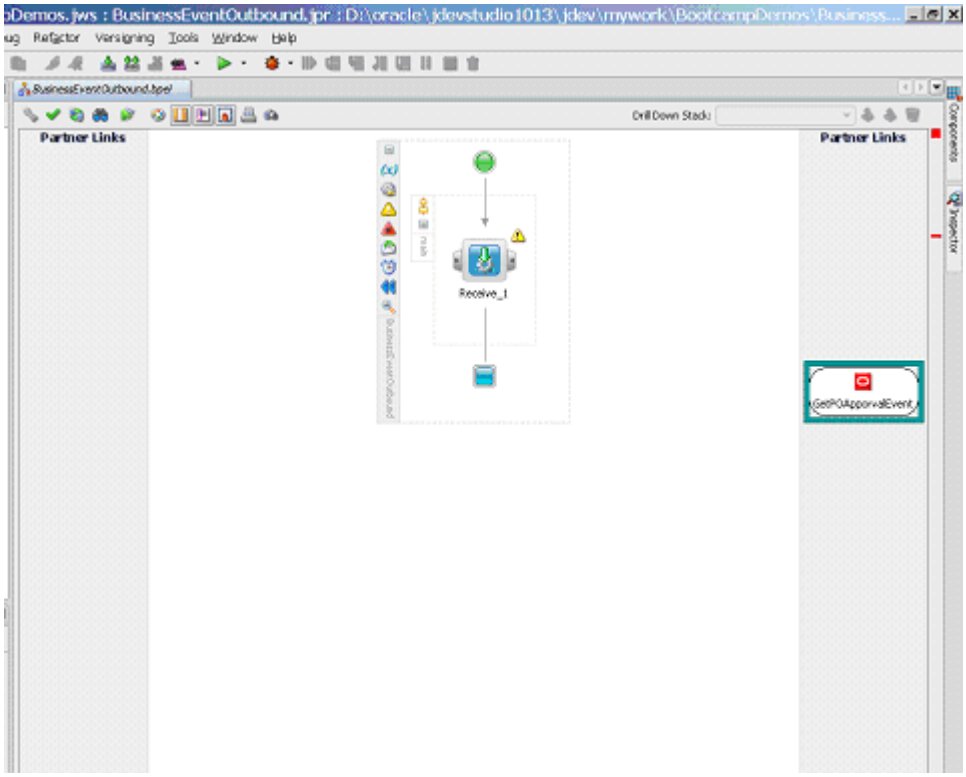
Configuring the Receive Activity

The next task is to configure a Receive activity to receive XML data from the partner link that you configured for the Oracle Application adapter service for business events.

To configure the Receive activity:

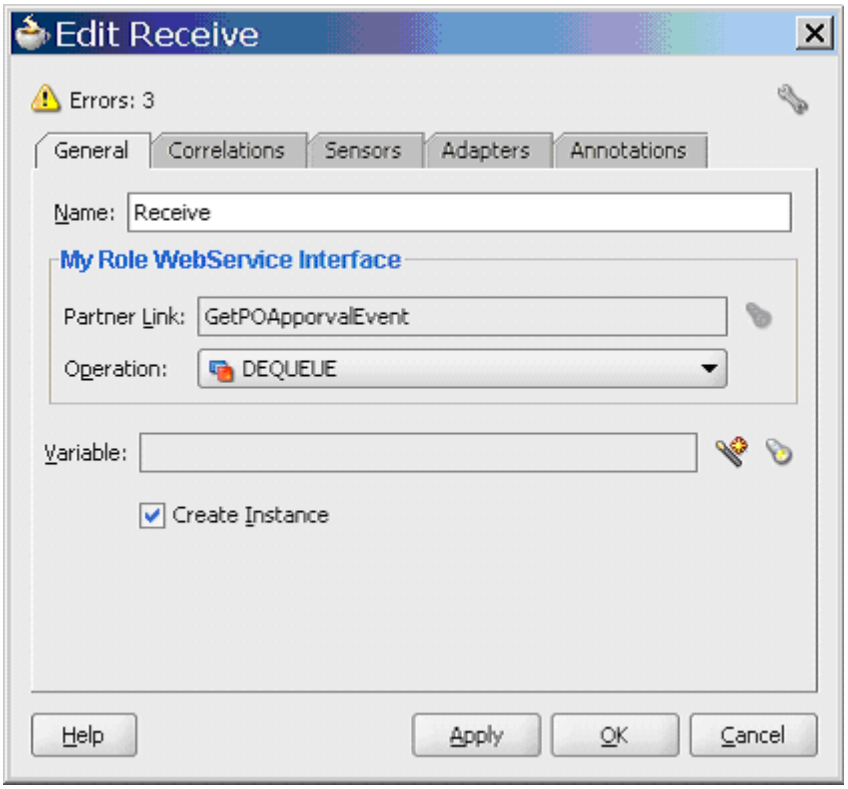
1. In JDeveloper BPEL Designer, drag and drop the **Receive** activity from the **BPEL Activities** section of the Component Palette into the Activity box of the process diagram.

Adding the Receive Activity



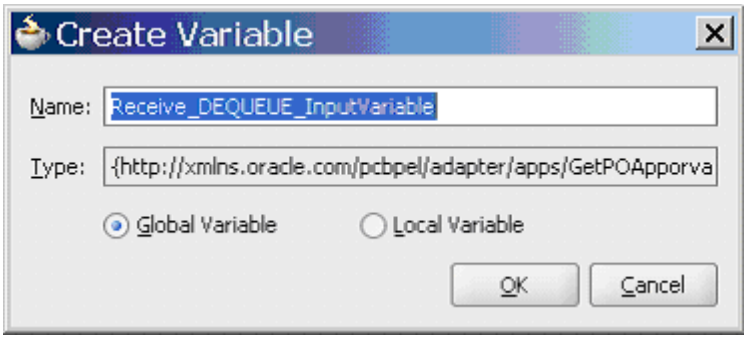
2. Link the Receive activity to the GetPOApprovalEvent partner link. The Receive activity will take event data from the partner link. The Edit Receive dialog box appears.

Editing the Receive Activity



3. Enter a name for the receive activity, then click the **Create** icon next to the **Variable** field to create a new variable. The Create Variable dialog box appears.

Creating a Variable



4. Select **Global Variable,** then enter a name for the variable. You can also accept the default name. Click **OK** to return to the Edit Receive dialog box.

5. Select **Create Instance,** then click **Apply** and **OK** to finish configuring the Receive activity.

Adding a Partner Link for the File Adapter

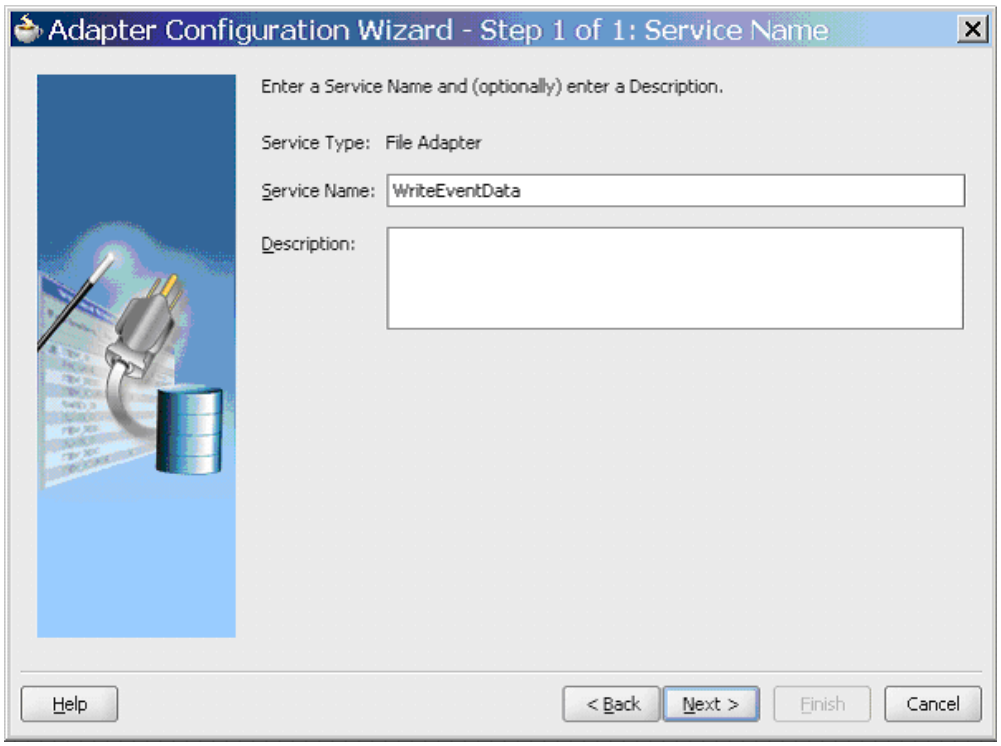
If you are configuring an outbound business event, you need to add another partner link for the file adapter. This allows the outbound business event to be returned to an XML file.

To add a partner link for the file adapter:

1. In JDeveloper BPEL Designer, drag and drop the **File Adapter** service from the **Adapter Service** section of the Component Palette into the Partner Link area of the process diagram. The Adapter Configuration wizard appears.

2. Click **Next.** The Service Name dialog box appears.

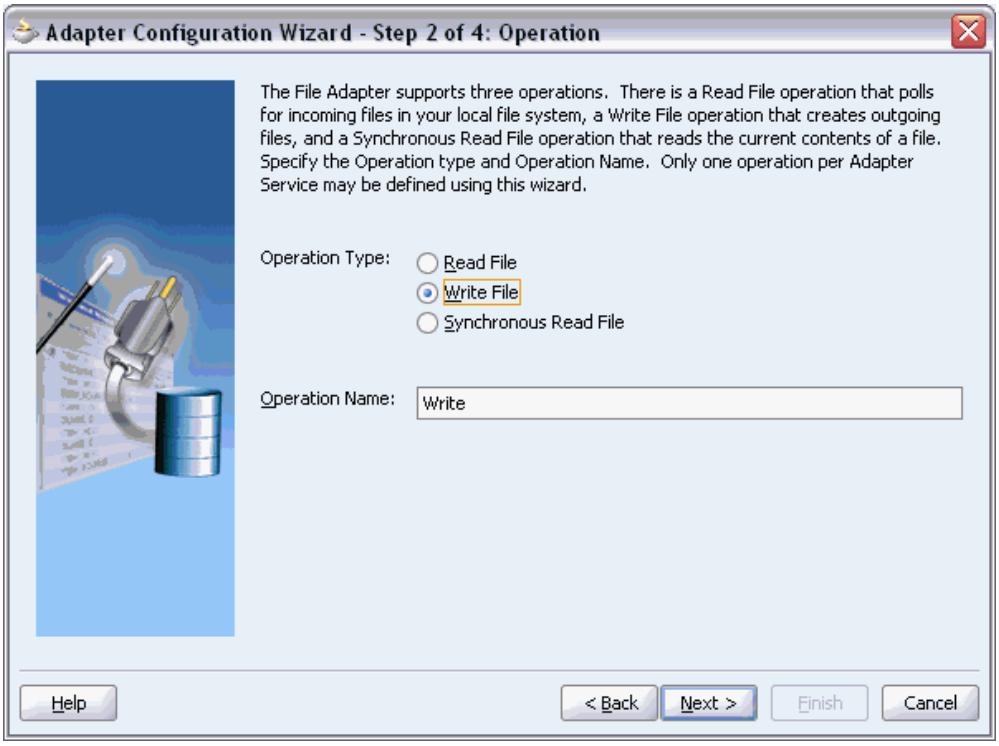
Specifying the Service Name



3. Enter a name for the file adapter service; for example, `WriteEventData`. You can also add an optional description of the service.

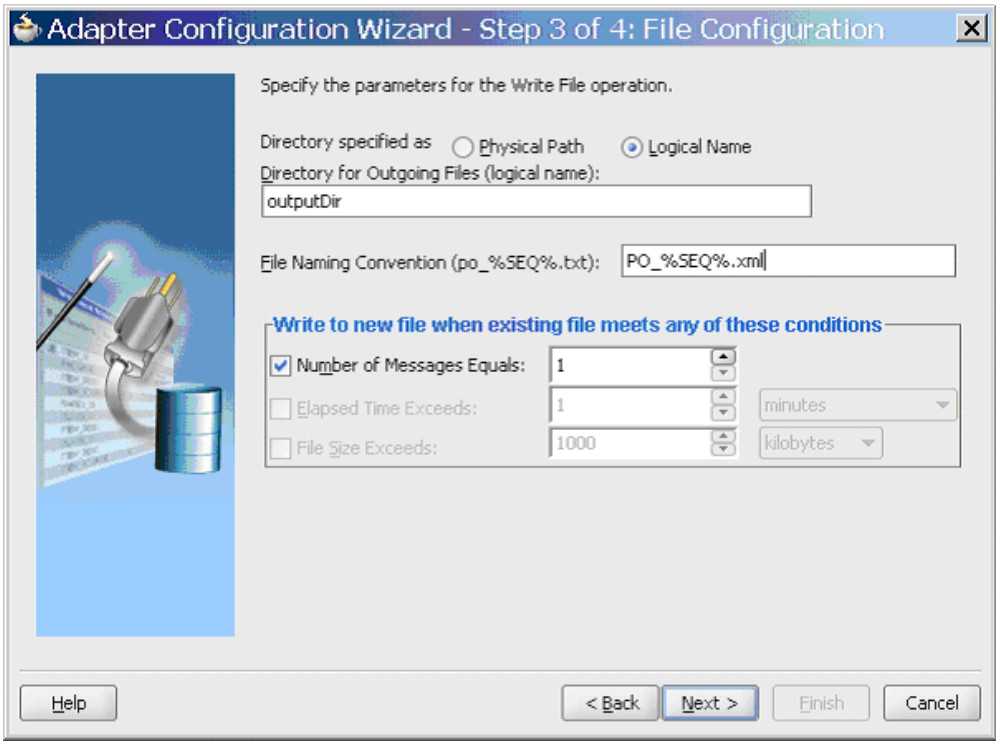
4. Click **Next** and the Operation dialog box appears.

Specifying the Operation



5. Specify the operation type; in this example, **Write File.** This automatically populates the **Operation Name** field. Click **Next** to access the File Configuration dialog box.

Configuring the Output File



6. For **Directory specified as,** select **Logical Name.** Enter `outputDir` as the **Directory for Outgoing Files,** and specify a naming convention for the output file; for example, `PO_%SEQ%.xml`.

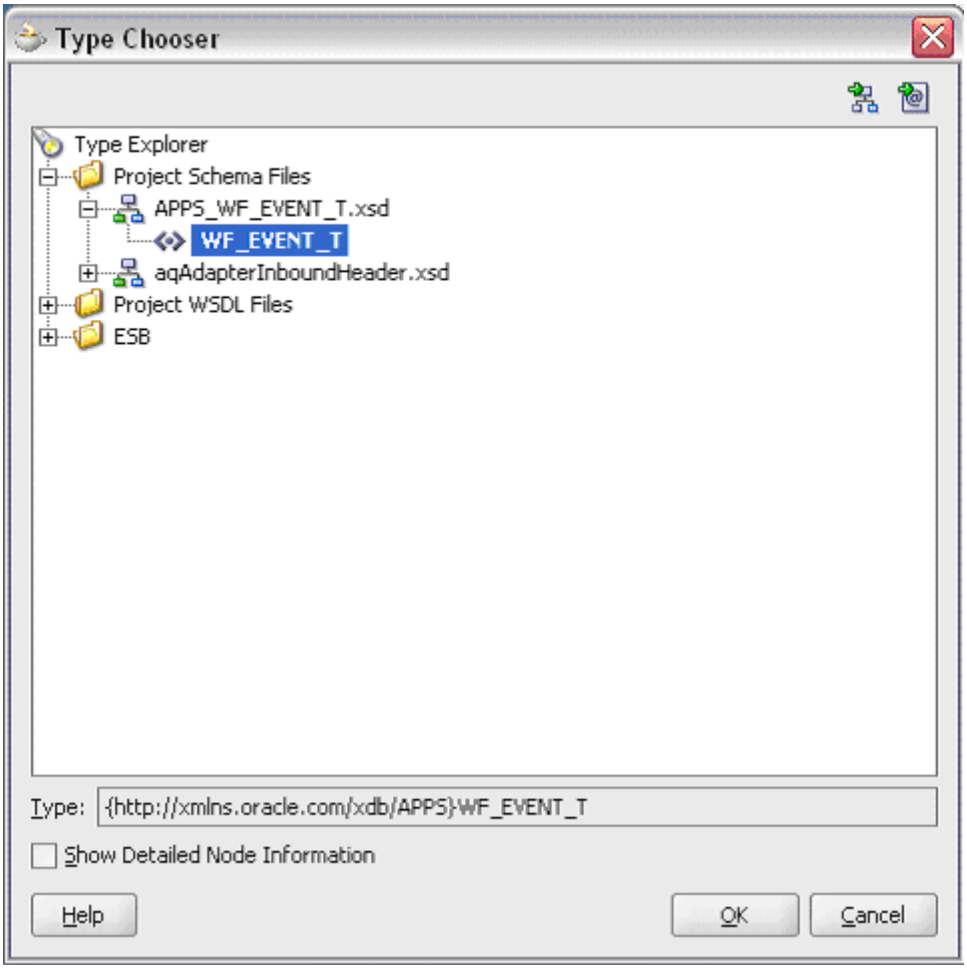
   Tip: When you type a percent sign (`%`), you can choose from a list of date variables or a sequence number variable (`SEQ`) as part of the filename.

   Confirm the default write condition: **Number of Messages Equals 1**.

7. Click **Next,** and the Messages dialog box appears. For the output file to be written, you must provide a schema.

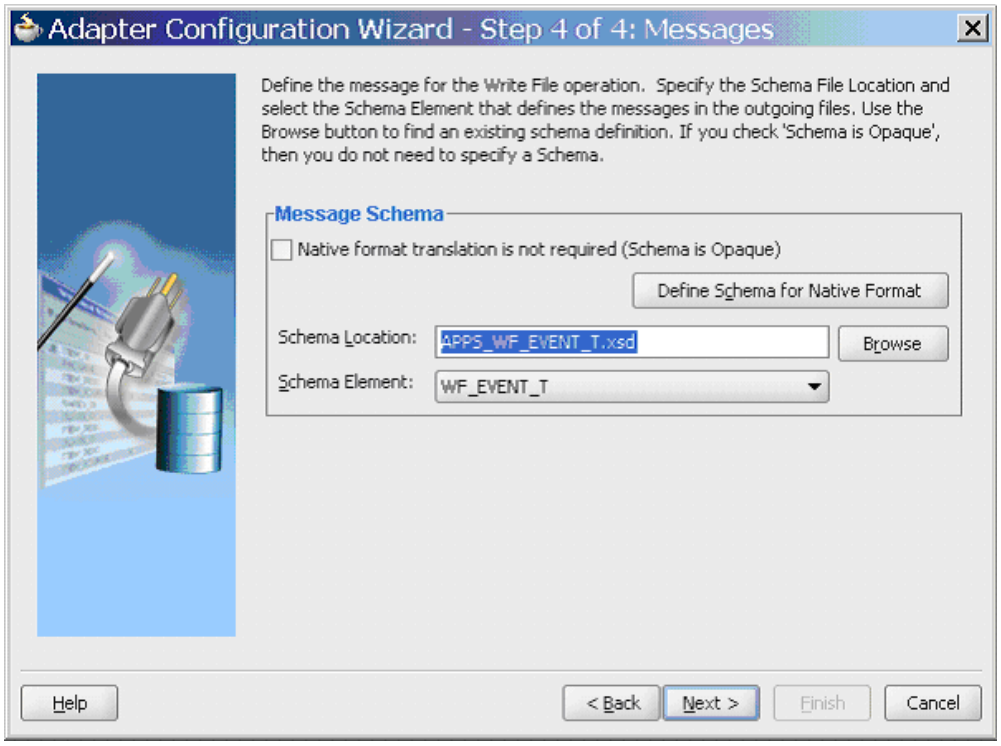
8. Click **Browse** to access the Type Chooser.

Choosing the Message Schema



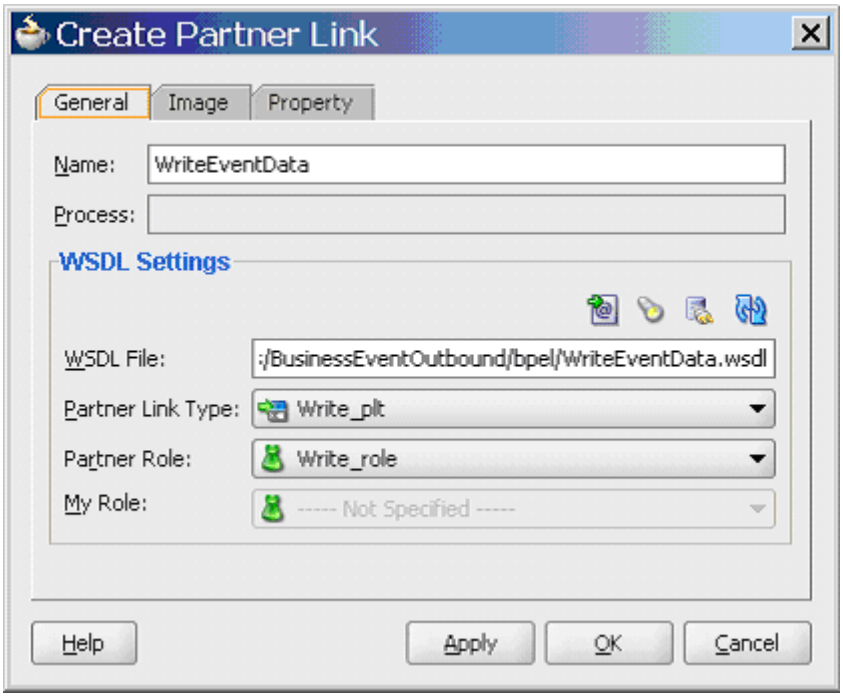
9. In the Type Chooser, navigate to and select the **WF_EVENT_T** schema from the Project Schema Files section, then click **OK** to return to the Messages dialog box.

Specifying the Message Schema



10. Click **Next**, then **Finish**. The wizard generates the WSDL file corresponding to the partner link. The main Create Partner Link dialog box appears, specifying the new WSDL file.

Completing the Partner Link Configuration



11. Click **OK** to complete the configuration and create the partner link with the required WSDL settings for the File Adapter Service.

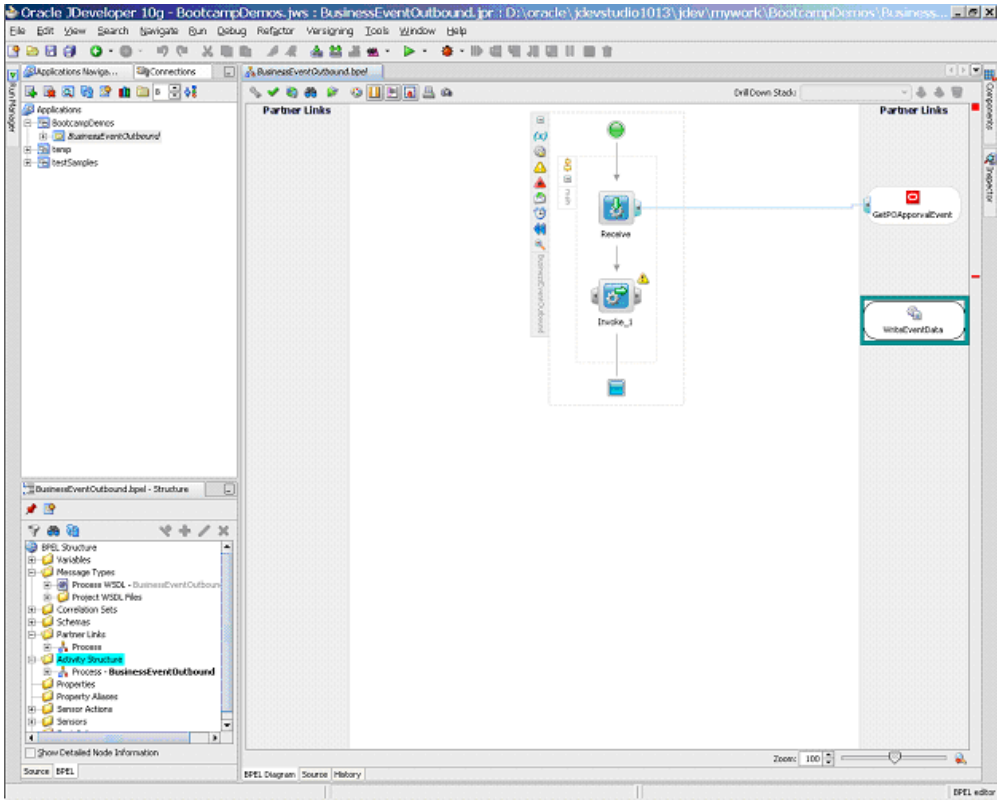
Configuring the Invoke Activity

After adding and configuring the partner link, the next task is to configure the BPEL process itself. You can start by configuring the Invoke process activity to write the business event information to the file.

To configure the Invoke activity:

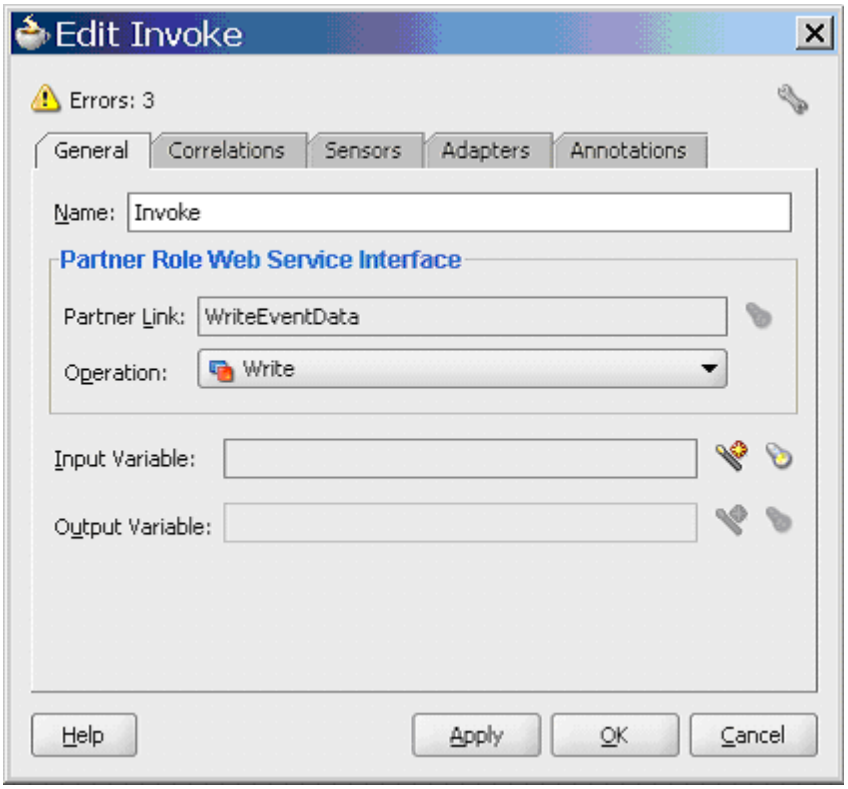
1. In JDeveloper BPEL Designer, drag and drop the **Invoke** activity from the Component Palette into the Activity box of the process diagram, below the **Receive** activity.

Adding the Invoke Activity



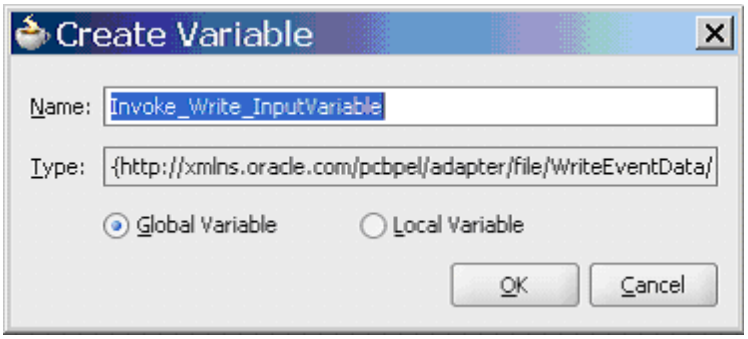
2. Link the Invoke activity to the WriteEventData file adapter service. The Invoke activity will send event data to the partner link. The Edit Invoke dialog box appears.

Editing the Invoke Activity



3. Enter a name for the Invoke activity, then click the **Create** icon next to the **Input Variable** field to create a new variable. The Create Variable dialog box appears.

Creating a Variable



4. Select **Global Variable,** then enter a name for the variable. You can also accept the default name. Click **OK,** then **OK** in the Edit Invoke dialog box to finish configuring the Invoke activity.

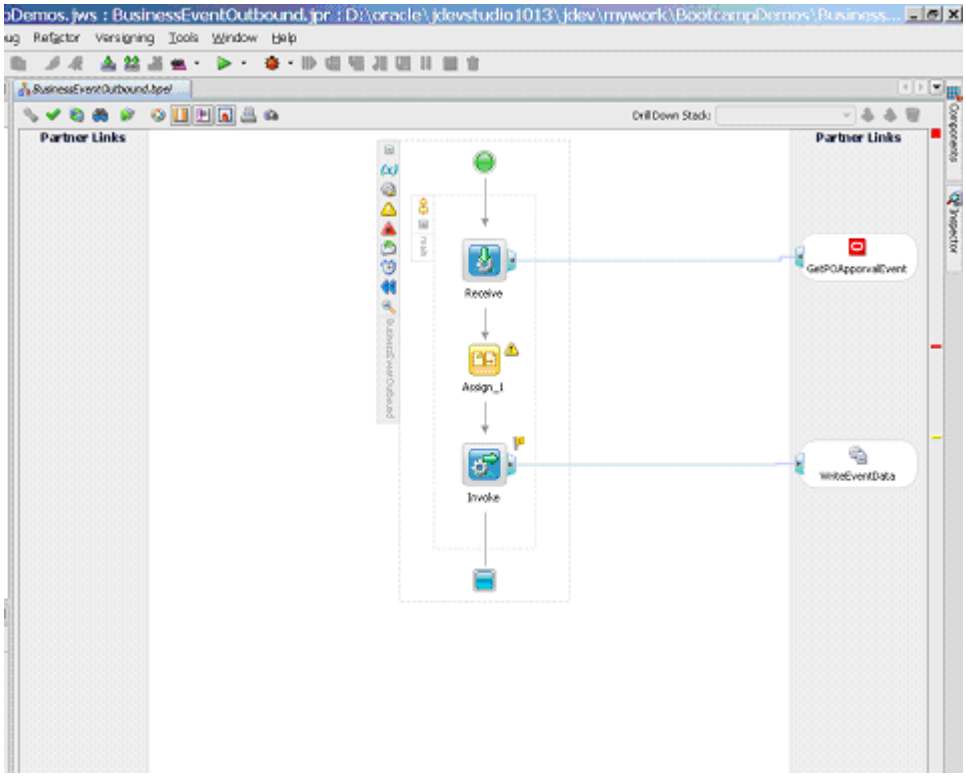
Configuring the Assign Activity

The next task is to add an Assign activity to the process map. This activity is configured to assign the values to the variables to invoke activity.

To configure the Assign activity:

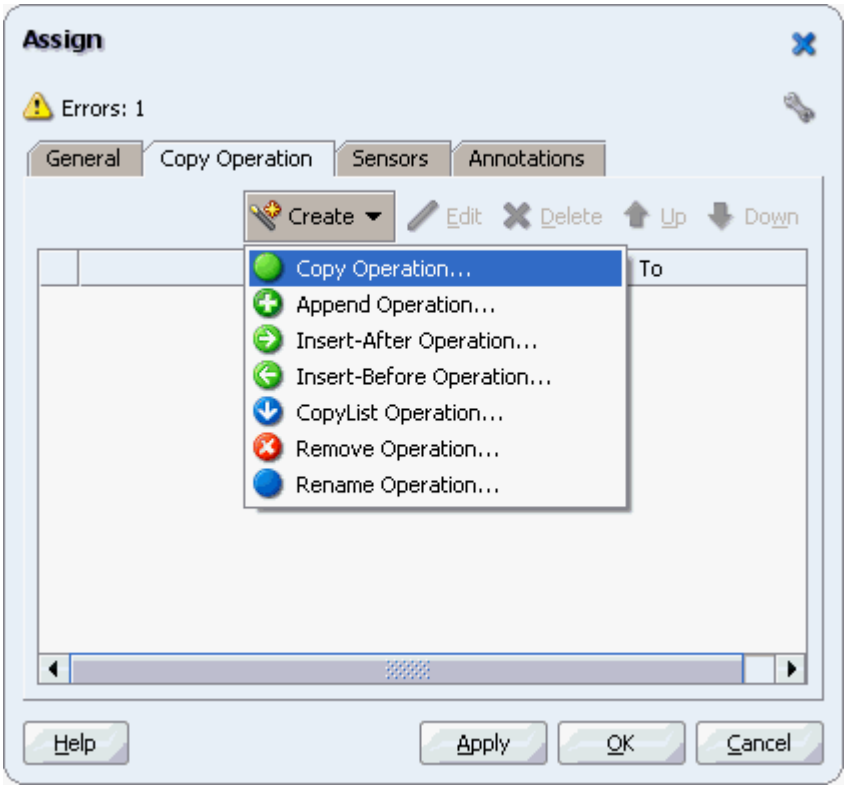
1. In JDeveloper BPEL Designer, drag and drop the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the **Receive** activity and the **Invoke** activity.

Adding the Assign Activity



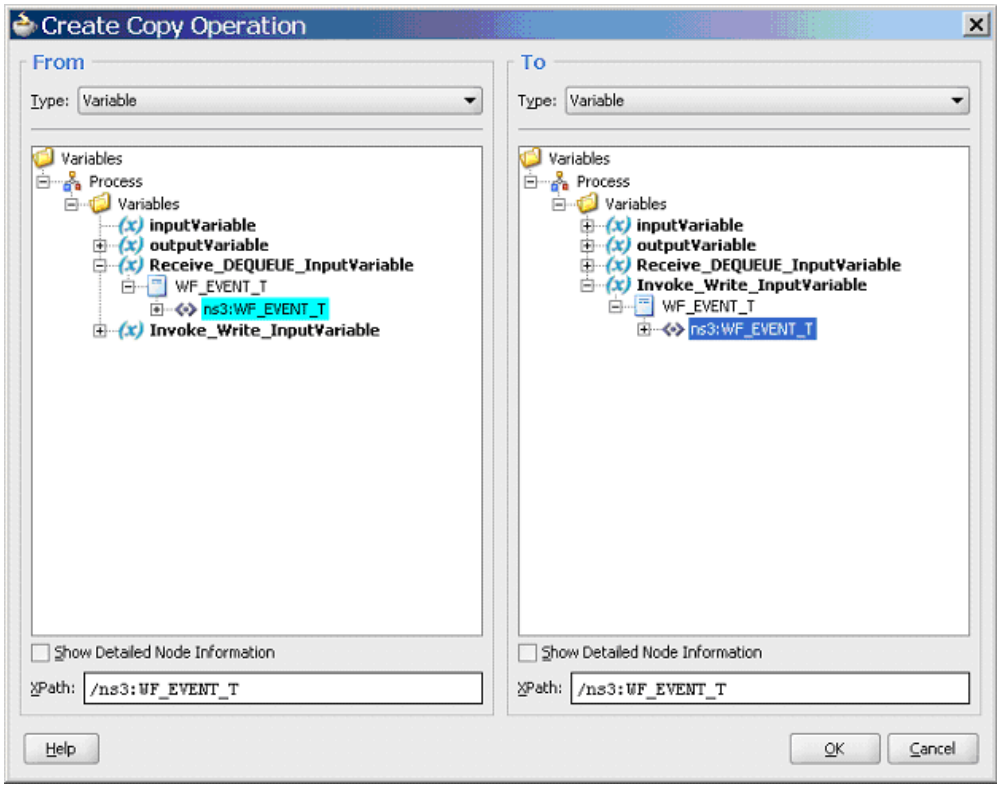
2. Double-click the **Assign** activity to access the Edit Assign dialog box.

Specifying a Copy Operation Action



3. On the Copy Operation tab, click **Create**, then select **Copy Operation** from the menu. The Create Copy Operation window appears.

Defining the Copy Operation

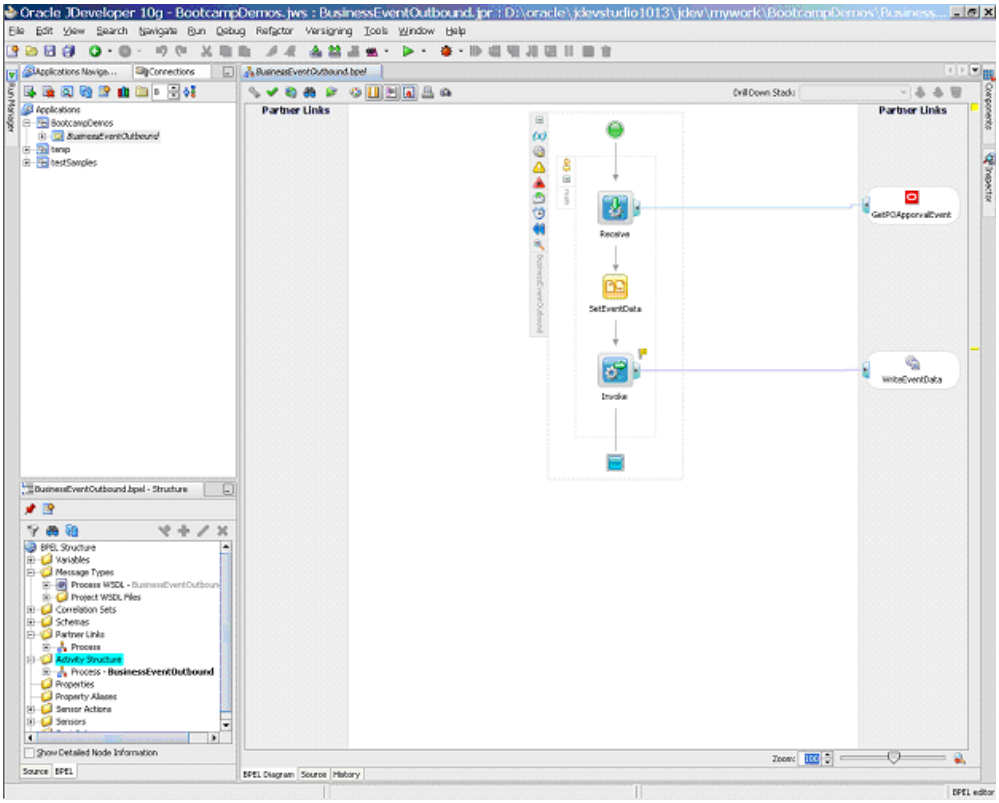


4. In the From navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Receive_DEQUEUE_InputVariable > WF_EVENT_T** and select **ns3:WF_EVENT_T**. The XPath field should contain your selected entry.

5. In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke_Write_InputVariable > WF_EVENT_T** and select **ns3:WF_EVENT_T**. The XPath field should contain your selected entry.

6. Click **OK** and then click **OK** in the Edit Assign dialog box to complete the configuration of the Assign activity.

Completed Outbound Business Event BPEL Process Project



Run-Time Tasks for Outbound Business Events

After designing the BPEL process, you can compile, deploy and test it.

1. Deploy the BPEL process, page 3-36.
2. Test the BPEL process, page 3-38.

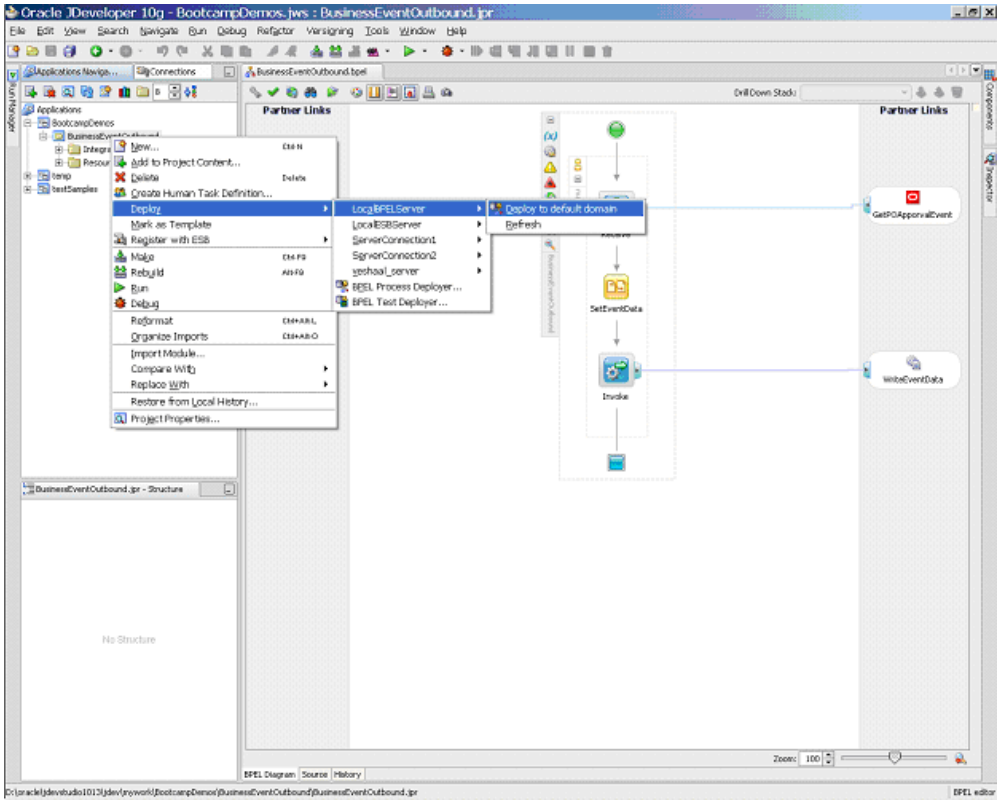
Deploying the BPEL Process

You need to deploy the BPEL process before you can run it. The BPEL process is first compiled and then deployed to the BPEL server.

To deploy the BPEL process:

1. In the Applications Navigator of JDeveloper BPEL Designer, select the **BusinessEventOutbound** project.

Deploying the BPEL Process



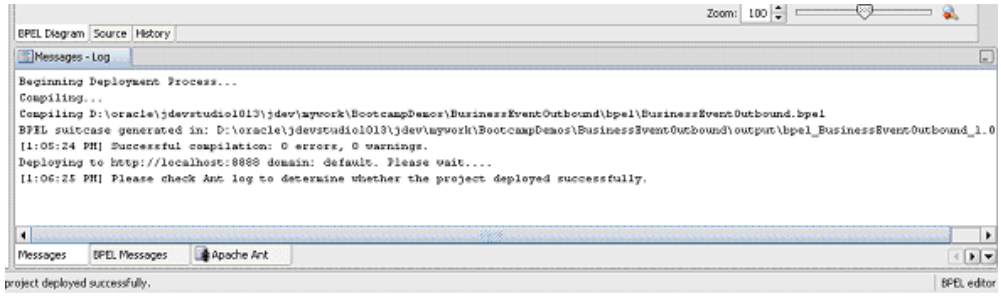
2. Right-click the project and select **Deploy > LocalBPELServer > Deploy to Default Domain** from the menu.

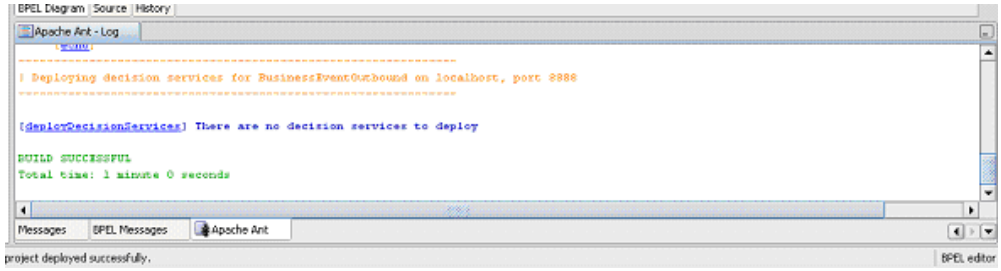
   The Password Prompt dialog box appears.

3. Enter the password for the default domain in the **Domain Password** field and click **OK.**

   The BPEL project is compiled and successfully deployed.

Compilation and Deployment Message Logs





Testing the BPEL Process

Once the BPEL process is deployed, it can be seen in the BPEL console. You can manage and monitor the process from the BPEL console. You can also test the process and the integration interface by manually initiating the process.

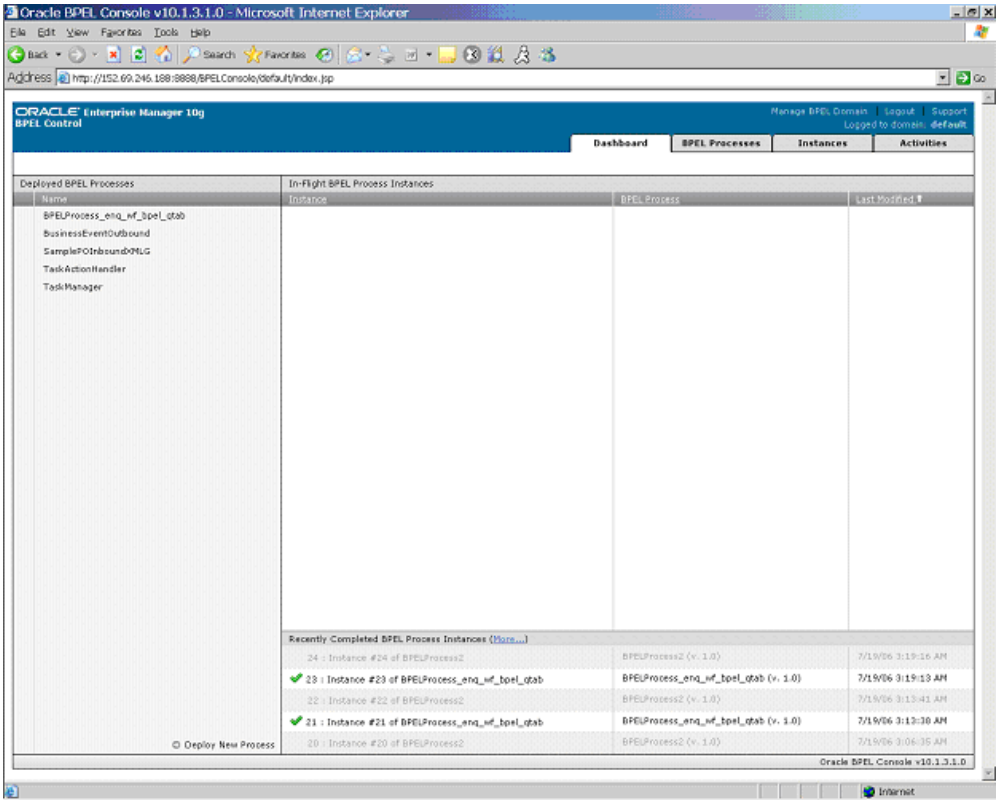
To test the BPEL process:

1. Log into Oracle BPEL Process Manager, then select **BPEL Console.** The BPEL console login screen appears.

2. Select **Default** in the **Domain** box. Enter the password for the default domain in the **Password** field and click **Login** to access the console.

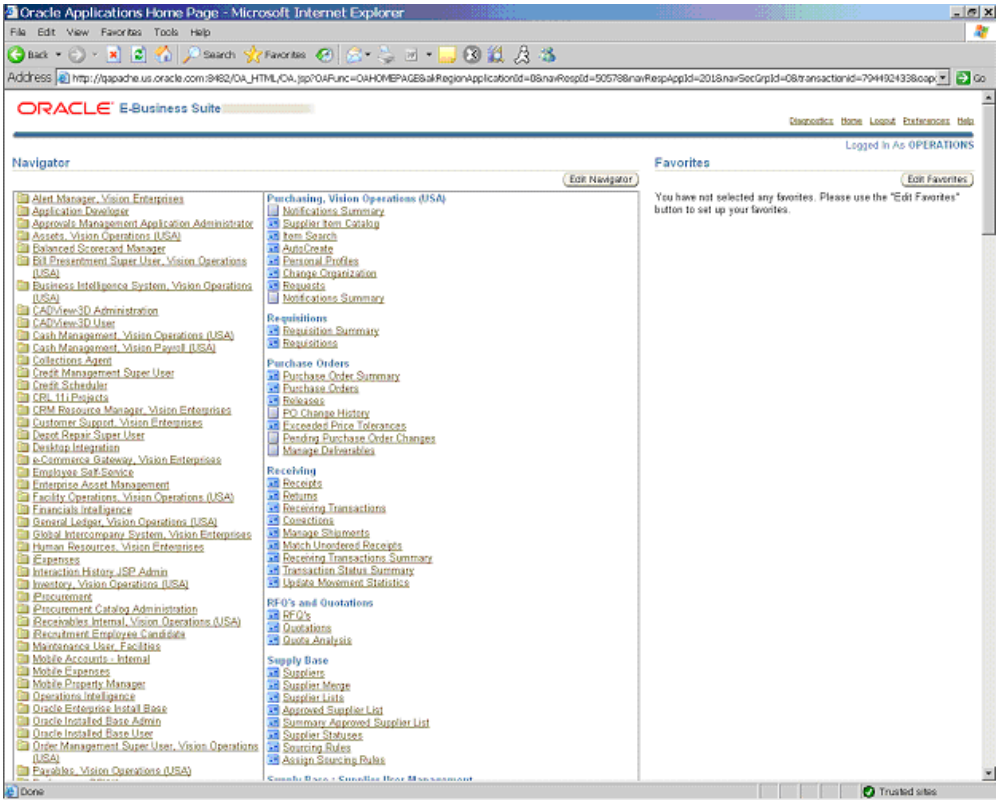
3. In the BPEL console, confirm that **BusinessEventOutbound** has been deployed.

Deployed BPEL Processes



4. Open up Oracle Applications and log in. The Oracle Applications home page appears, with the list of responsibilities.

Accessing the Oracle Applications Home Page

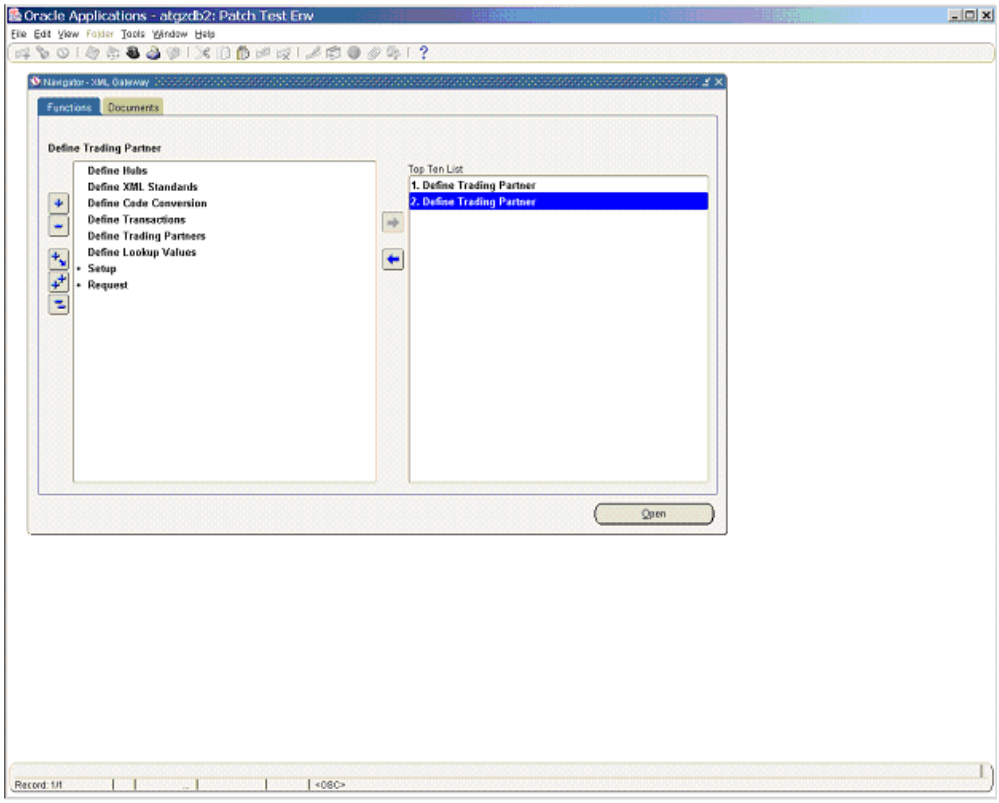


5. Select the responsibility **Purchasing, Vision Operations (USA),** then click **Purchase Orders** within that responsibility.

   The Oracle Applications Forms open up with the Purchase Order forms.

6. Close the Forms and select **File > Switch Responsibility,** then select the **XML Gateway** responsibility.

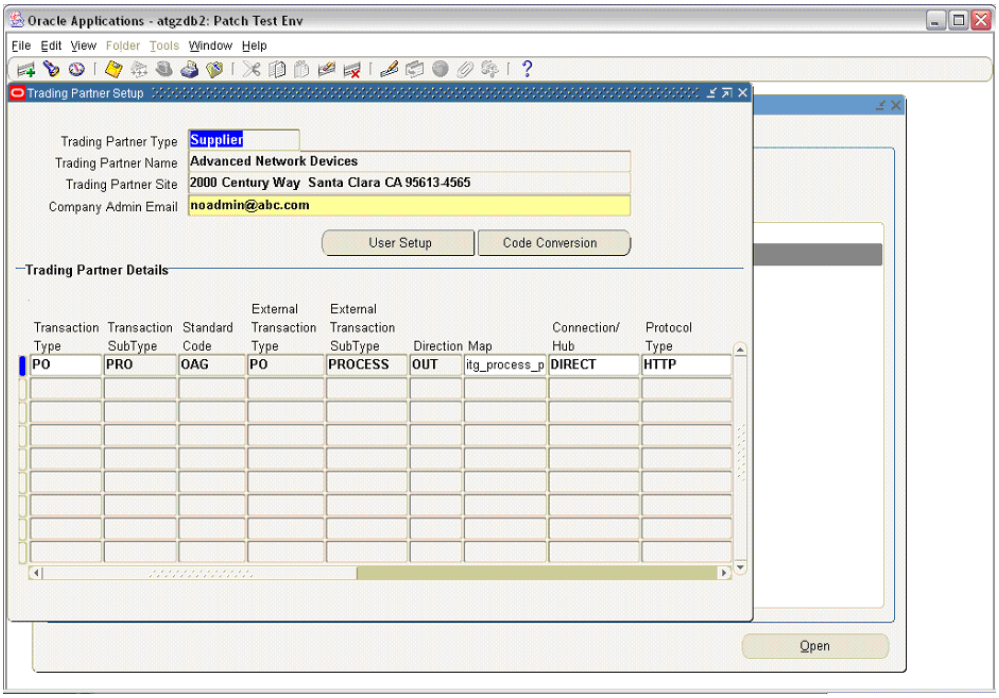
Selecting XML Gateway Functions



7. Select **Define Trading Partner** to access the Trading Partner forms.

   Press **<F11>** to access the Trading Partner Setup form.

Setting up Trading Partners

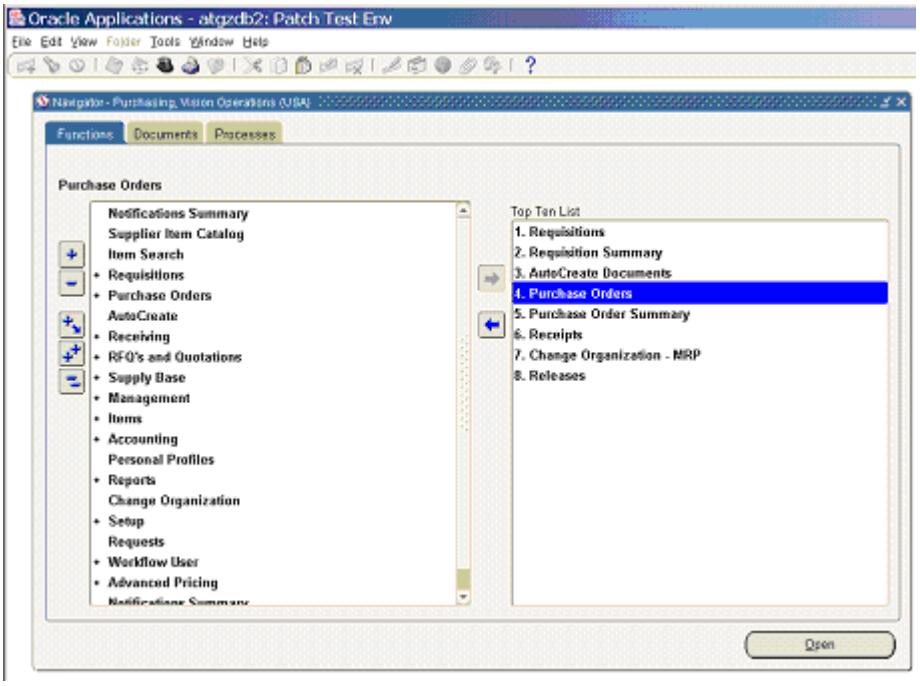


8. Enter the header values on the Trading Partner Setup form as follows:
   - **Trading Partner Type** - `Supplier`
   - **Trading Partner Name** - for example, `Advanced Network Devices`
   - **Trading Partner Site** - *trading_partner_site_address*
   - **Company Admin Email** - *valid_email_id*
9. Enter a data row with the following values:
   - **Transaction SubType** - `PRO`
   - **Standard Code** - `OAG`
   - **External Transaction Type** - `PO`
   - **External Transaction SubType** - `PROCESS`
   - **Direction** - `OUT`
   - **Map** - `itg_process_po_007_out`

- **Connection / Hub** - `DIRECT`
- **Protocol Type** - `HTTP`
- **Username** - *username*
- **Password** - *password*
- **Protocol Address** - *valid_protocol_address*
- **Source Trading Partner Location Code** - *valid_location_code*

10. Save the trading partner details and switch responsibility back to **Purchasing, Vision Operations (USA).**

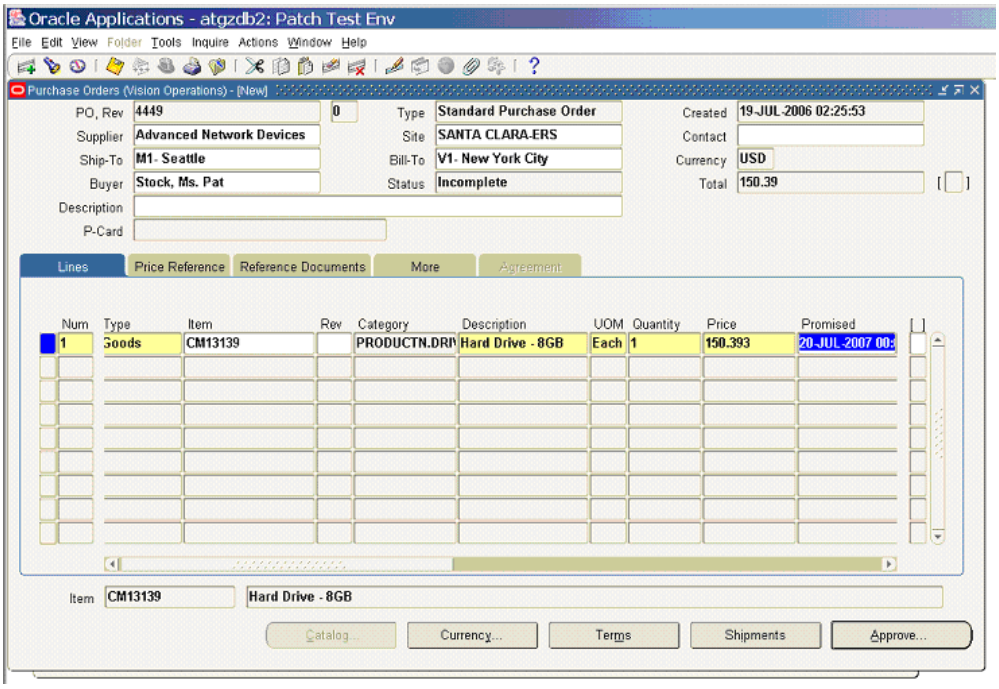
Selecting Purchasing Functions



11. Select **Purchase Orders** to access the Purchase Order forms.

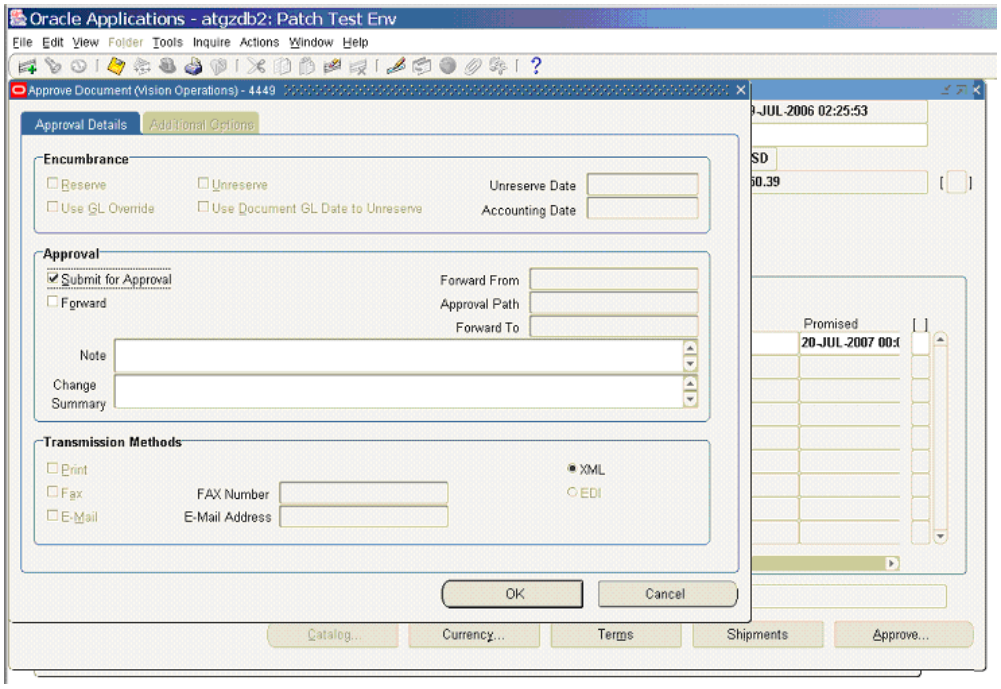
    Press **<F11>** to access the Purchase Order Setup form.

Setting up a Purchase Order



12. Create a purchase order with header values reflecting the trading partner you previously defined:
    - **Supplier** - `Advanced Network Devices`
    - **Site** - `SANTA CLARA-ERS`
13. On the Lines tab, enter a data row with the following values:
    - **Item** - `CM13139`
    - **Quantity** - `1`
    - **Promised** - *some_future_date*
14. Save the purchase order. The Approve Document form appears.

Approving the Purchase Order



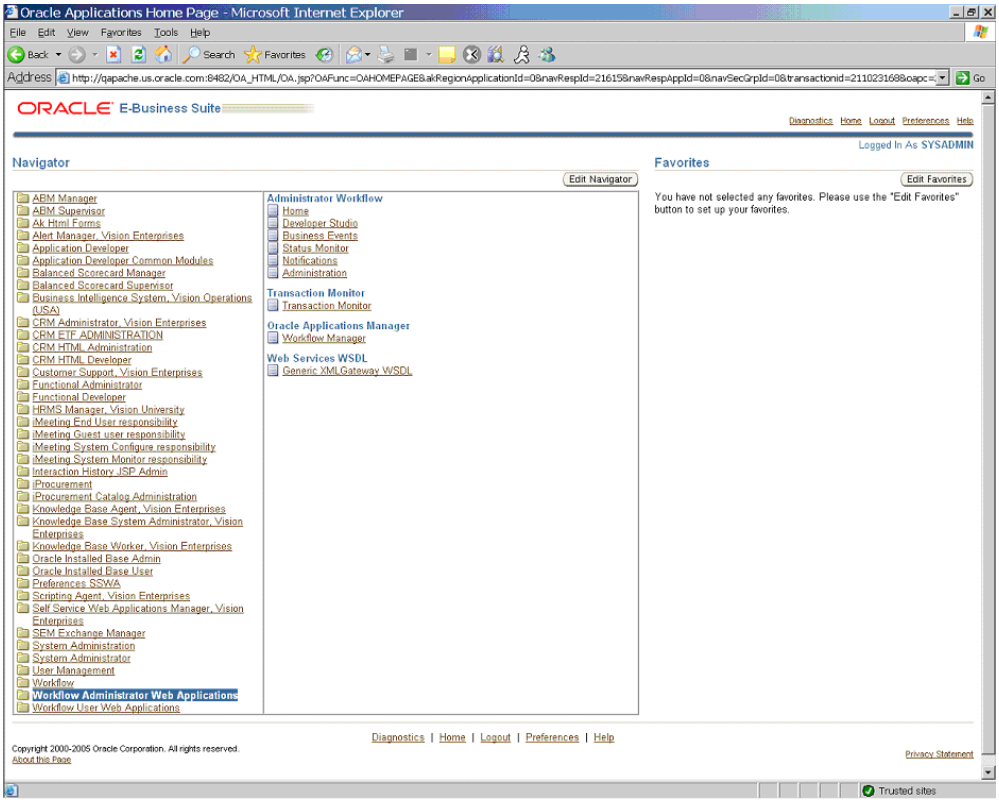
15. **Note:** Because the trading partner is set up and valid, the transmission method is automatically set to **XML**.

Click **OK** to approve the purchase order. The Purchase Order Setup form reappears. The status of the purchase order is now `Approved`. For future reference, note the value of the **PO, Rev** field. Once the purchase order is approved, the business event **oracle.apps.po.event.xmlpo** is raised.

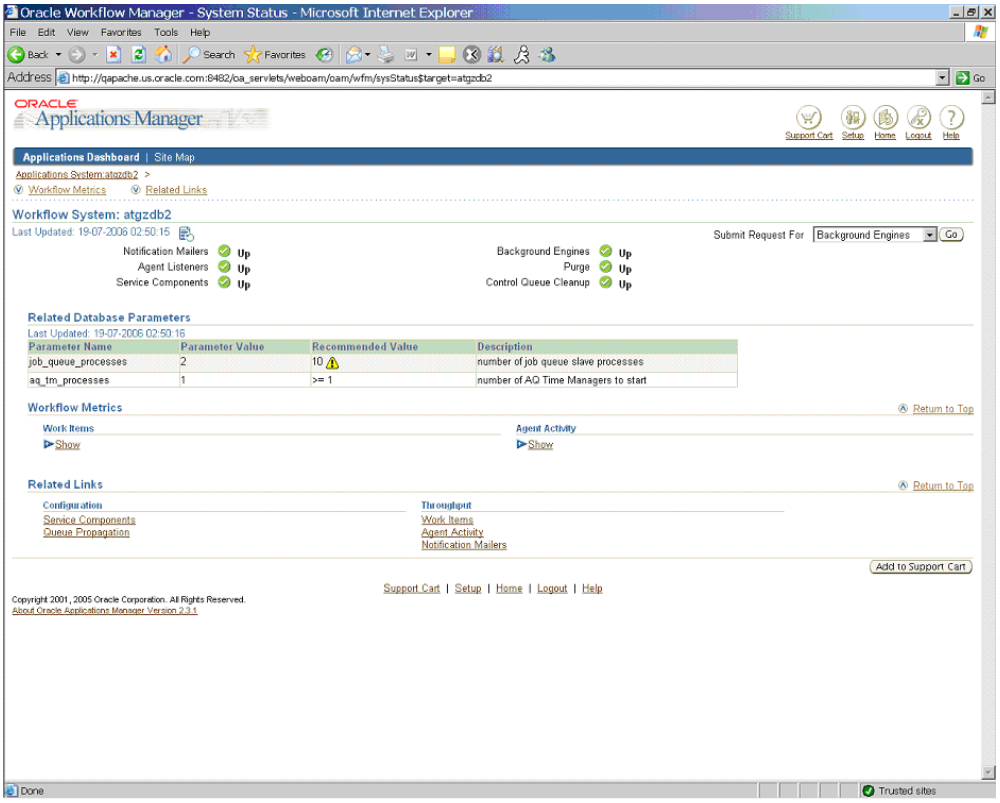
Next, you should ensure that the **WF_Deferred** agent listener is running on the target database.

16. Log into Oracle Applications as System Administrator.

Accessing the Oracle Applications Home Page

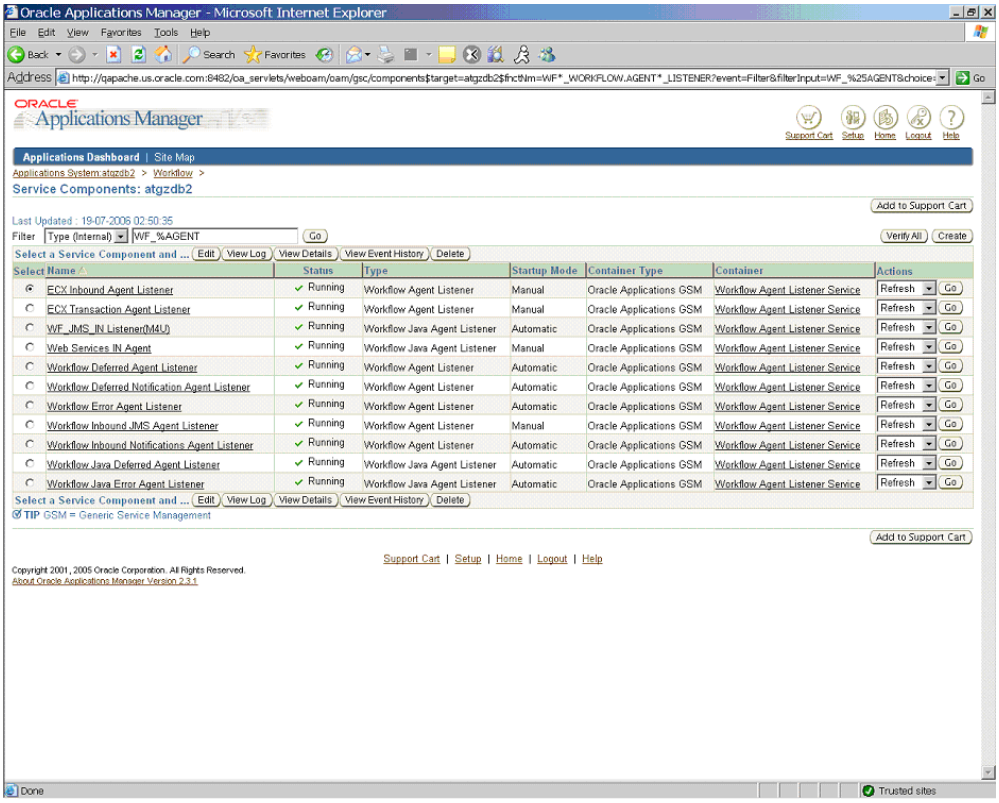


17. On the Oracle Applications home page, select the **Workflow Administrator Web Applications** responsibility, then **Oracle Applications Manager > Workflow Manager.**



18. On the Applications Manager page, click the **Agent Listeners** icon. The Service Components page appears, containing a list of the installed agent listeners.

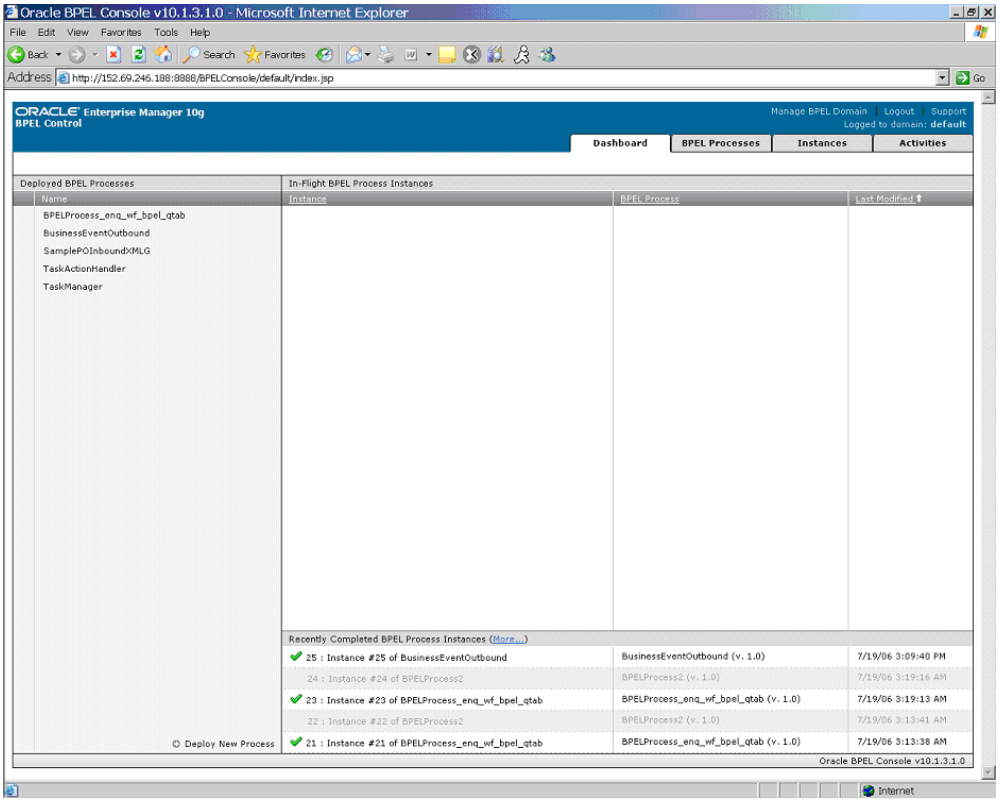
Reviewing Agent Listener status



19. Confirm that the **Workflow Deferred Agent Listener** is in `Running` status.

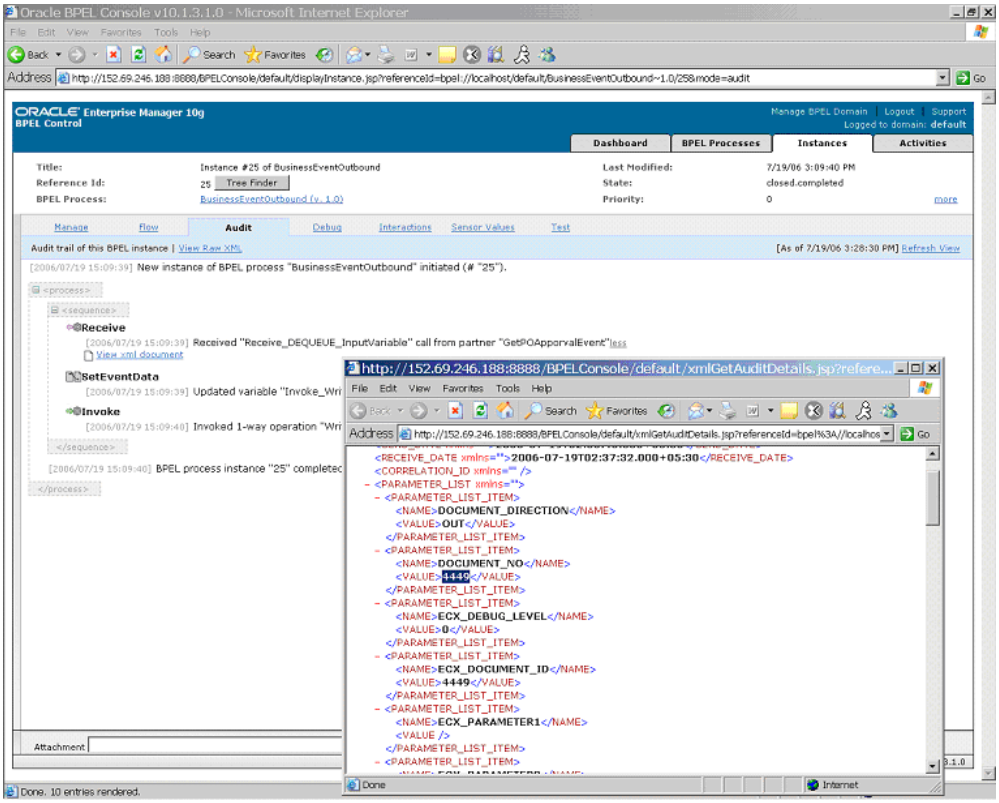
20. Log into Oracle BPEL Process Manager, and return to the BPEL Console.

Confirming Completed BPEL Processes



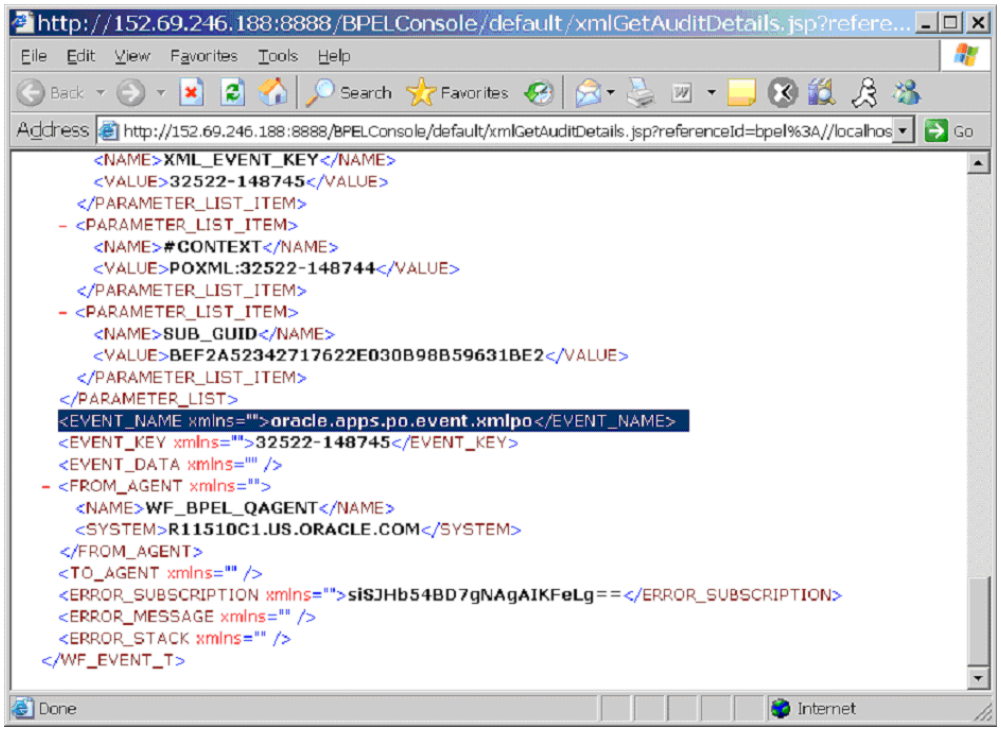
21. Confirm that the **BusinessEventOutbound** process has completed, then select the instance, which opens up in the **Instance** tab. Select the **Audit** secondary tab.

Auditing the BusinessEventOutbound Instance Receive Activity



22. Open the xml file for the **Receive** activity, and note the name of the event raised - `oracle.apps.po.event.xmlpo`.

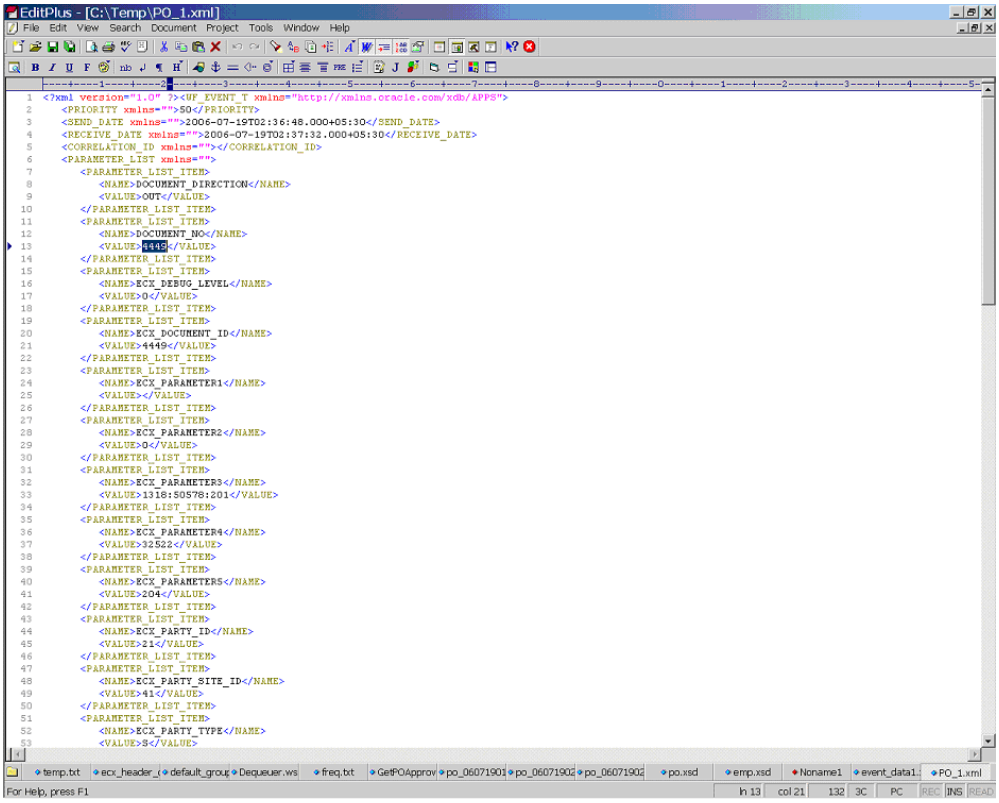
Examining the Receive Event Name



23. Examine the **Assign** and **Invoke** activities as well for the event raised and document number.

24. Go to the directory you specified for the write operation; for example, **outputDir** (typically `c:\temp`). Open the output file (for example, `PO_1.xml`), and confirm that the order number is same as that of the approved purchase order.

Confirming the Output Order Number



Troubleshooting and Debugging

If you experience problems with your Business Event System integration, you can take the following troubleshooting steps:

- Confirm that WF_Listener is up and running.
- Ensure that business events are raised only after the BPEL process is deployed.
- If the BPEL process is created on one instance of Oracle Applications and deployed on another instance, ensure the following:
  - WF_BPEL_Q, WF_BPEL_QTab, and WF_BPEL_QAgent should be present on the target database.
  - A custom subscription for the raised business event should be present on the target database.

If you still experience problems with your integration, you can enable debugging.

Enabling Debugging

You can enable debugging for business events using the BPEL Process Manager.

To enable debugging:

1. Log into your BPEL Process Manager domain.
2. Select `yourdomain.collaxa.cube.ws`
3. Select **Debug.**

Debugging information is output to the log file for your domain. To examine the log file in the BPEL Process Manager, navigate to **Home > BPEL Domains >*yourdomain* > Logs.** The log file is ***yourdomain*.log**.

4

Using Concurrent Programs

This chapter covers the following topics:

- Overview of Concurrent Programs
- Design-Time Tasks for Concurrent Programs
- Creating a New BPEL Project
- Adding a Partner Link
- Configuring the Invoke Activity
- Configuring the Transform Activity
- Run-Time Tasks for Concurrent Programs
- Deploying the BPEL Process
- Testing the BPEL Process
- Verifying Records in Oracle Applications

Overview of Concurrent Programs

OracleAS Adapter for Oracle Applications uses *concurrent programs* to move data from interface tables to base tables.

A concurrent program is an instance of an execution file. Concurrent programs use a concurrent program executable to locate the correct execution file. Several concurrent programs may use the same execution file to perform their specific tasks, each having different parameter defaults.

Design-Time Tasks for Concurrent Programs

This section describes how to configure the OracleAS Adapter for Oracle Applications to use concurrent programs. It describes the tasks required to configure OracleAS Adapter for Oracle Applications using the Adapter Configuration Wizard in Oracle JDeveloper.

Prerequisites to Configuring Concurrent Programs

OracleAS adapter for Oracle Applications is deployed using the BPEL Process Manager (PM) in Oracle JDeveloper. The BPEL PM creates the WSDL interfaces for the concurrent programs.

You must populate certain variables in the BPEL PM in order to provide context information for Oracle Applications. The context information required for a concurrent program includes the *username* and *responsibility* of an Oracle Applications user who has sufficient privileges (based on the applications context of Organization ID, Username and Responsibility) to run the program. The default value passed for the username is `SYSADMIN`. The default value passed for responsibility is `SYSTEM ADMINISTRATOR`.

You can change the default values specified in the generated WSDL for the username and responsibility. This is a static way of changing the context information. These values will apply to only the invocation of the deployed business process where you change the values. However, if you must provide different context information for different invocations of the business process, then you can dynamically populate the header variable with values for username and responsibility. The context information can be specified by configuring an Assign activity before the Invoke activity in the BPEL PM.

Following is a list of the procedures required to accomplish the design-time tasks.

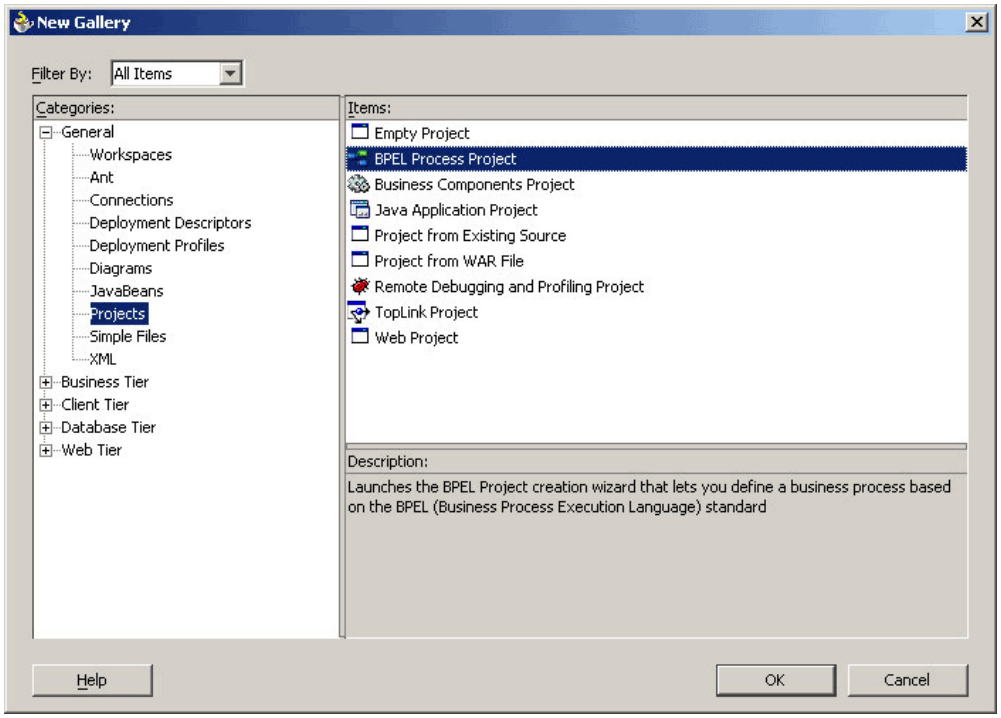
1. Create a new BPEL project., page 4-2
2. Add a partner link., page 4-5
3. Configure the Invoke activity., page 4-15
4. Configure the Transform activity., page 4-23

Creating a New BPEL Project

To create a new BPEL project:

1. Open JDeveloper BPEL Designer.
2. From the **File** menu, select **New**. The New Gallery dialog box appears.
3. Select **All Items** 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.

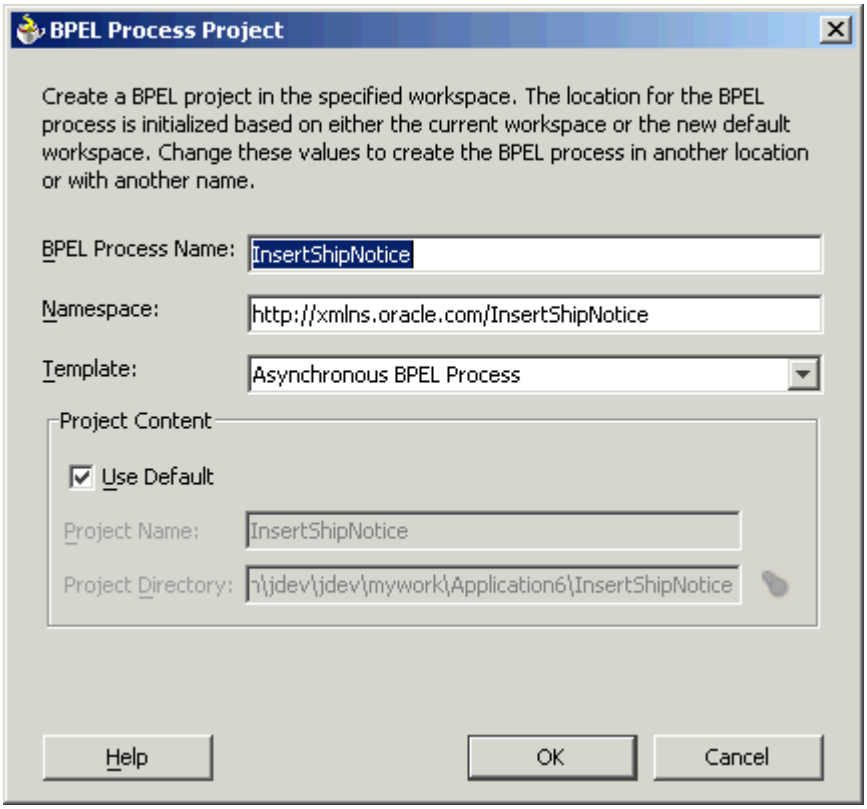
Creating a New BPEL Process Project



6. Click **OK**. The BPEL Process Project dialog box appears.

7. In the **BPEL Process Name** field, enter a descriptive name. For example, `InsertShipNotice`.

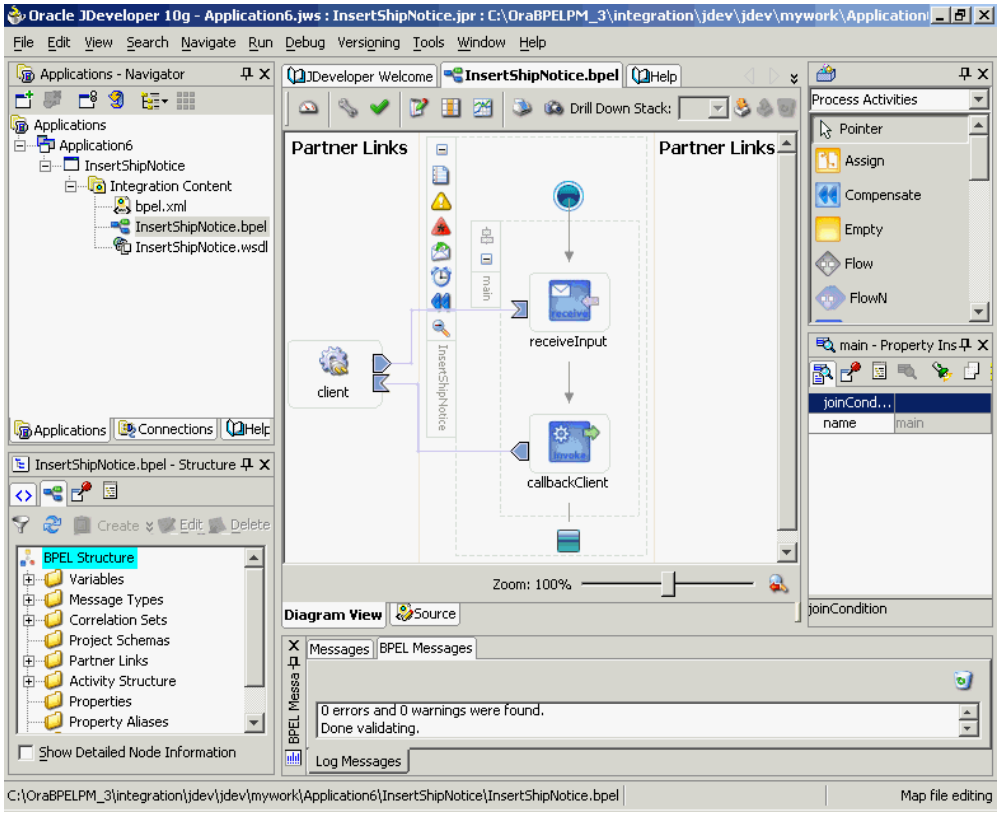
Specifying a Name for the New BPEL Process Project



8. From the **Template** box, select **Asynchronous BPEL Process**. Keep the default selection for **Use Default** under Project Content.

9. Click **OK**. A new BPEL process, with the required source files including `bpel.xml`, `InsertShipNotice.bpel`, and `InsertShipNotice.wsdl` is created.

New BPEL Process Project



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.

To add a partner link:

1. Drag and drop **PartnerLink**, from the Component Palette, into the border area of the process diagram. The Create Partner Link dialog box appears.

2. Click the **Define Adapter Service** icon in WSDL Settings. The Adapter Configuration Wizard appears.

3. Click **Next.** The Adapter Type dialog box appears.

4. Select **Oracle Applications** to specify the adapter you want to configure.

Selecting OracleAS Adapter for Oracle Applications

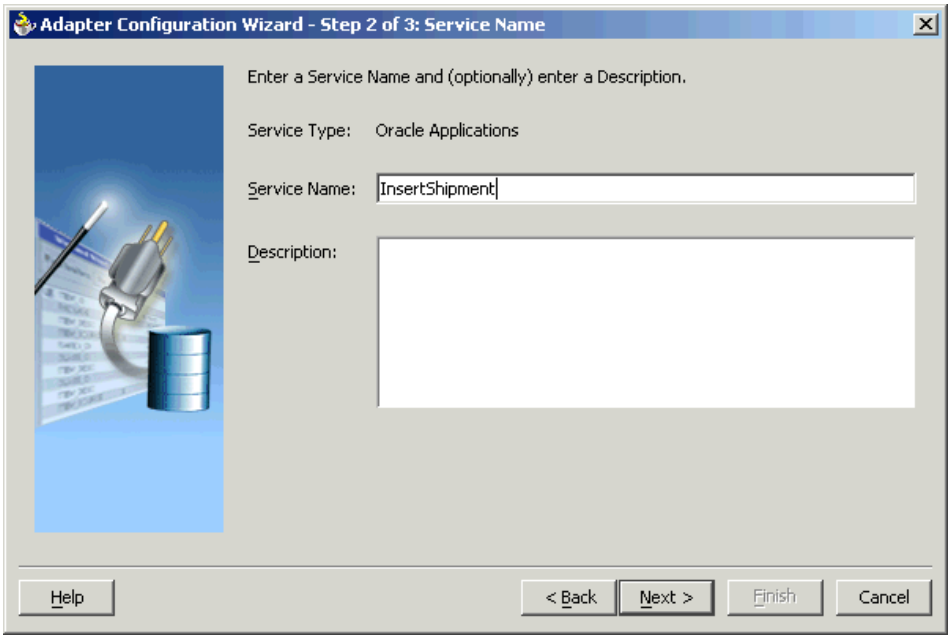
5. Click **Next,** and the Service Name dialog box appears.

6. Enter the following information:

   1. In the **Service Name** field, enter a service name.

   2. In the **Description** field, enter a description for the service. This is an optional field. Click **Next**.

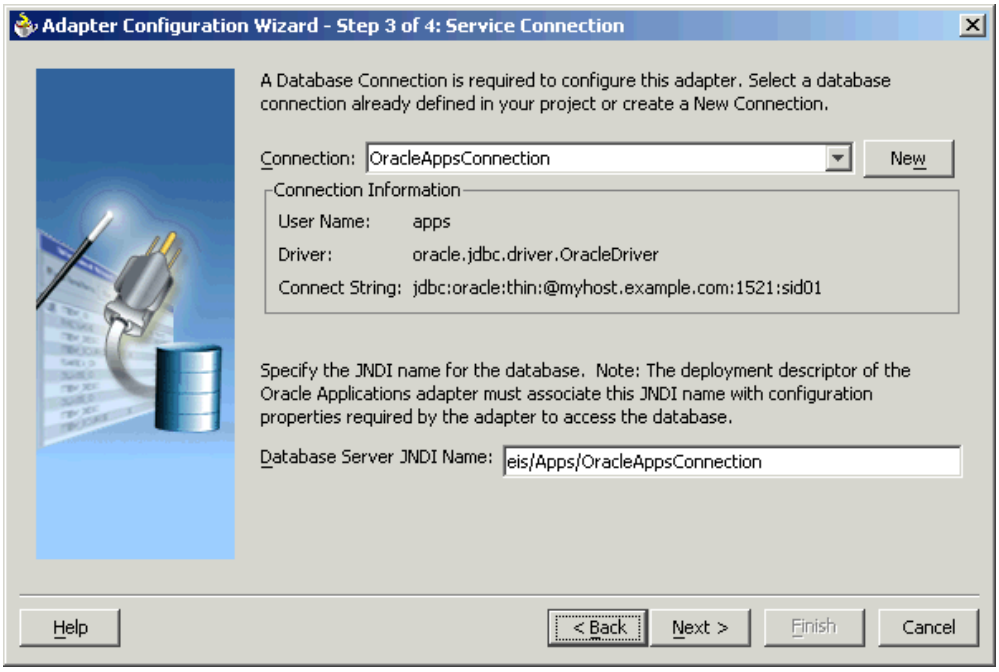
Specifying the Service Name



The Service Connection dialog box appears.

7. Click **Next**. The Service Connection dialog box appears.

Creating a New Database Connection



8. Click **New** to define a database connection. The Create Database Connection Wizard appears. Alternatively, you can select an existing database connection from the Connection list.

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

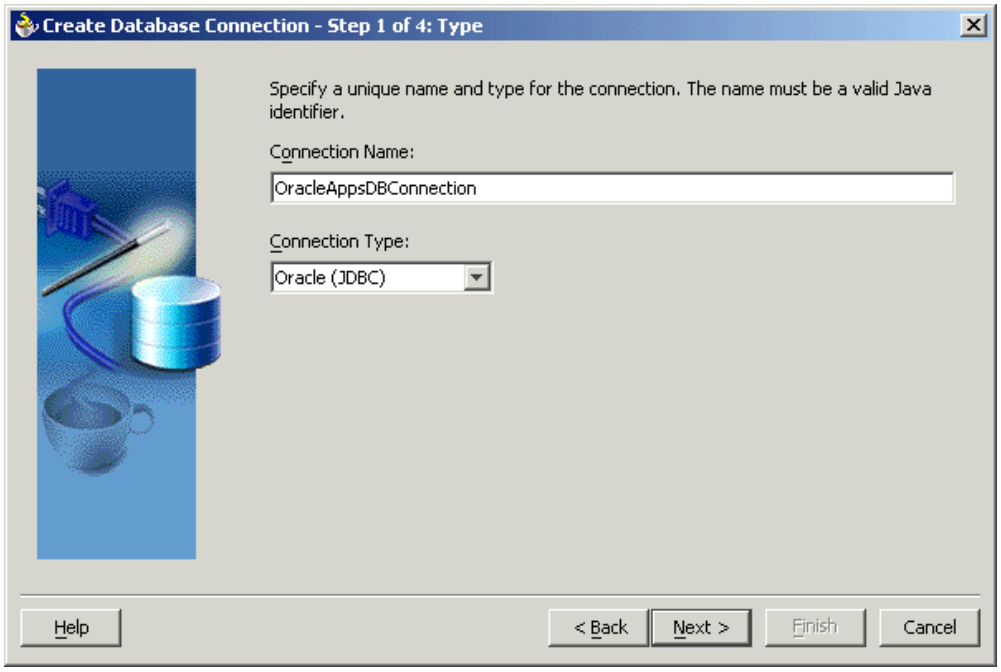
9. Click **Next**. The Type dialog box appears.

10. Enter the following information in the Type dialog box:

    1. In the **Connection Name** field, specify a unique name for the database connection.

    2. From the **Connection Type** box, select the type of connection for your database connection.

Specifying the Connection Name and Type of Connection



11. Click **Next**. The Authentication dialog box appears.

12. Enter information in the following fields:

    1. In the **UserName** field, specify a unique name for the database connection.

    2. In the **Password** field, specify a password for the database connection.

13. Click **Next**. The Connection dialog box appears.

14. Enter information in the following fields:

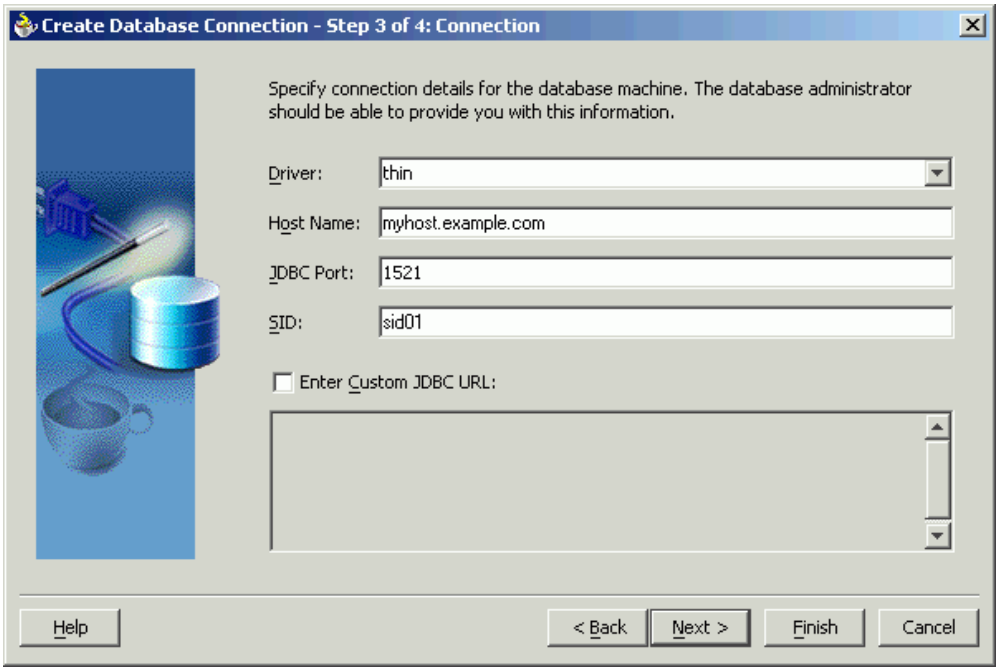
    1. From the **Driver** list, select **Thin**.

    2. In the **Host Name** field, specify the host name for the database connection.

    3. In the **JDBC Port** field, specify the port number for the database connection.

    4. In the **SID** field, specify a unique SID value for the database connection.

Specifying the New Database Connection Information



15. Click **Next**. The Test dialog box appears.

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

17. Click **Next**. The Service Connection dialog box appears, providing a summary of the database connection.

18. The JNDI (Java Naming and Directory Interface) name corresponding to the database connection appears automatically in the **Database Server JNDI Name** field. Alternatively, you can specify a JNDI name.

    Note: When you specify a JNDI name, the deployment descriptor of the Oracle Applications adapter must associate this JNDI name with configuration properties required by the adapter to access the database.

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.

Note: For more information about JNDI concepts, refer to *Oracle*

Application Server Adapter Concepts.

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

    Once you have completed creating a new connection for the service, you can add a concurrent program by browsing through the list of concurrent programs available in Oracle Applications.

20. Click **Next** in the Service Connection dialog box.

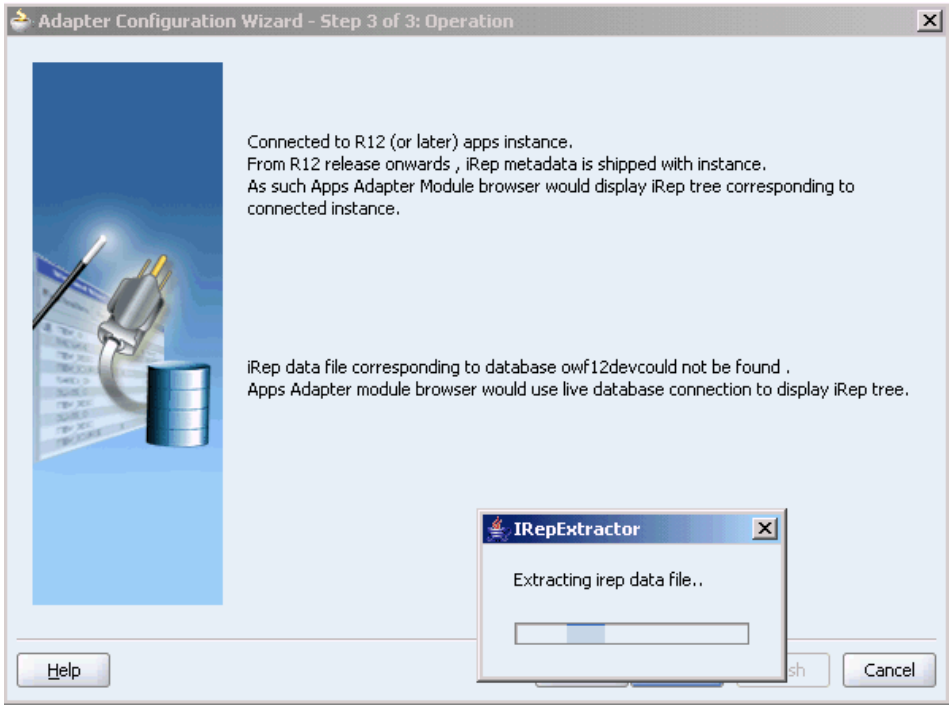
    For Oracle E-Business Suite Release 12:

    If you are connecting to Oracle E-Business Suite Release 12, then the **IREP File not present** dialog box appears indicating that OracleAS Adapter could not find the Oracle Integration Repository data file corresponding to the database you are connecting to Oracle Applications in your workspace. Absence of the data file would make browsing or searching of Integration Repository tree considerably slow. You can choose to extract the data file and create a local copy of the Integration Repository data file. Once it is created successfully, OracleAS Adapter will pick it up automatically next time and retrieve data from your local Integration Repository.

    You can select one of the following options:

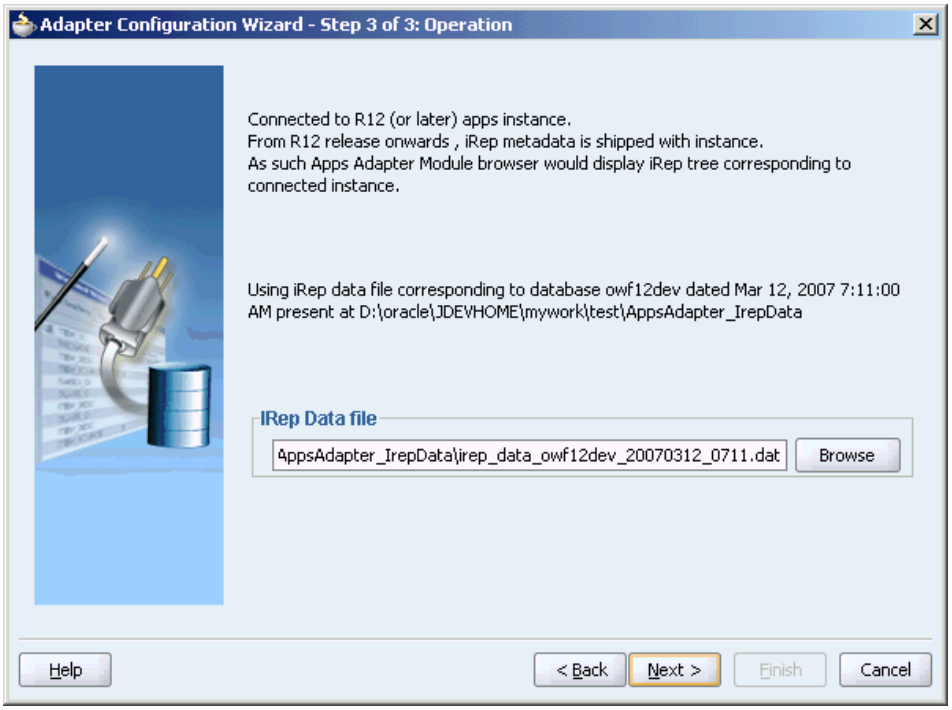
    - Click **Yes** to extract the Integration Repository data file.

Extracting Integration Repository Data File



After the system successfully creates a local copy of the Integration Repository data file, next time when you connect to the database, you will find the **IRep Data File** field appears in the Operation dialog box indicating where your local copy exists with the creation date and time as part of the file name.

Using the Local Integration Repository Data File



- Click **No** to query the Integration Repository data file from the live database you are connecting to display the Integration Repository tree.

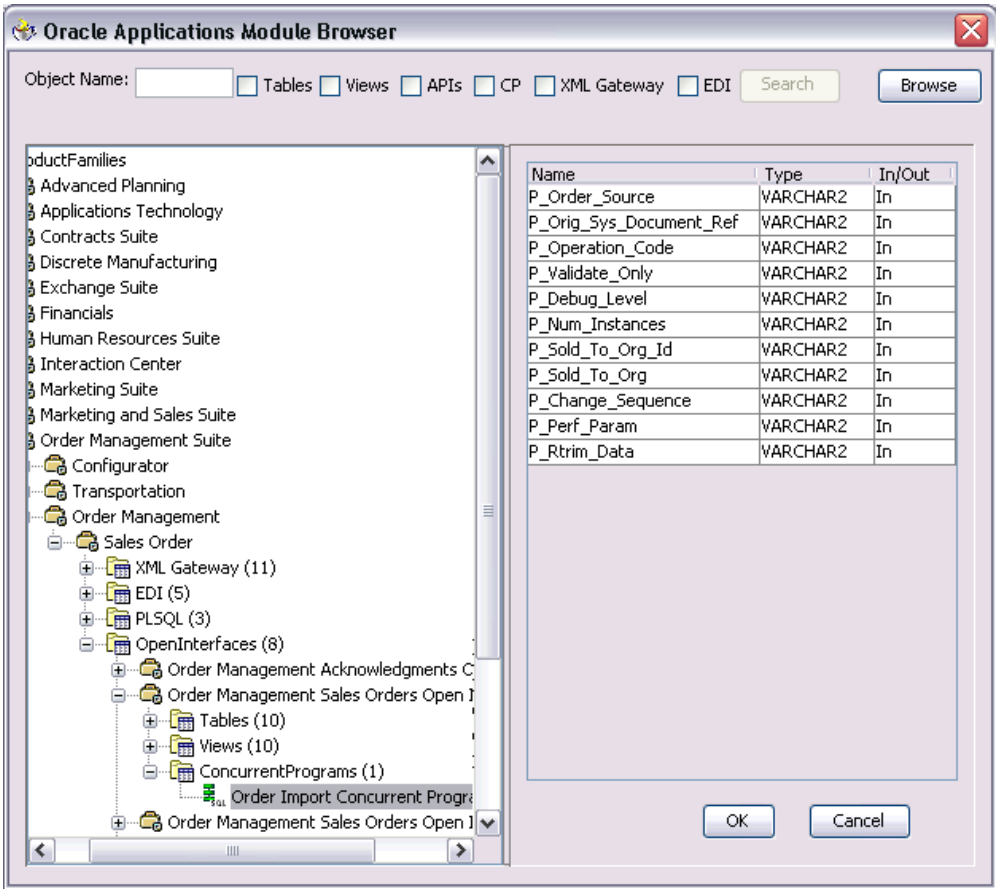
  Note: It is highly recommended that you create a local copy of the Integration Repository data file so that OracleAS Adapter will query the data next time from the local copy in your workspace to enhance the performance.

For Oracle E-Business Suite pre-Release 11.5.10:

If you are connecting to a pre-11.5.10 Oracle Applications instance, you must select the interface type in the Adapter Configuration Wizard. Select **Tables/Views/APIs/Concurrent Programs** to proceed.

21. Click **Get Object** to open the Oracle Applications Module Browser.

Selecting a Concurrent Program from the Module Browser



22. Oracle Applications Module Browser includes the various product families that are available in Oracle Applications. For example, Applications Technology or Order Management Suite are product families in Oracle Applications. The product families contain the individual products. For example, Order Management Suite contains the Order Management product. The individual products contain the business entities associated with the product. For example, the Order Management product contains the Sales Order business entity.

    Business entities contain the various application modules that are exposed for integration. These modules are grouped according to the interface they provide. concurrent programs can be found under the OpenIntefaces category.

23. Select a concurrent program, and then click **OK**. You can select only one concurrent program for each adapter service.

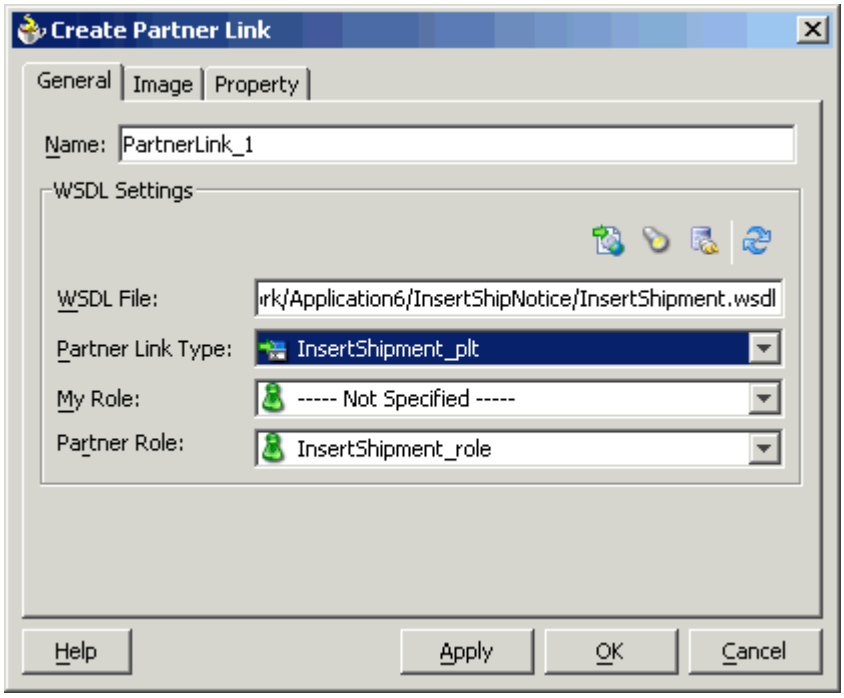
    Note: You can also search for a concurrent program by entering the name of the program in the **Object Name** field. Select the **CP** check

box, and then click **Search**.

24. The concurrent program is added to Operation Objects. Click **Next** in the Operation dialog box.

25. Click **Finish**. The wizard generates the WSDL file corresponding to the selected interface. This WSDL file is now available for the partner link.

Completing the Partner Link Configuration



26. Click **OK**. The partner link is created with the required WSDL settings.

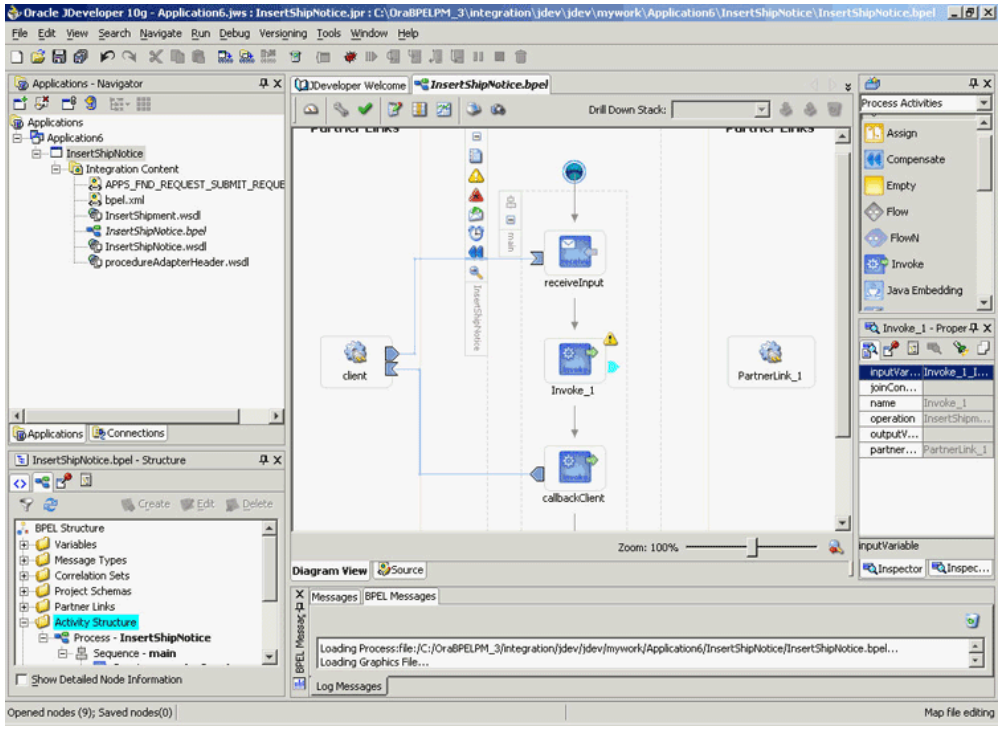
Configuring the Invoke Activity

After adding and configuring the partner link, the next task is to configure the BPEL process. This includes configuring the Invoke activity and the Transform activity.

To configure the Invoke activity:

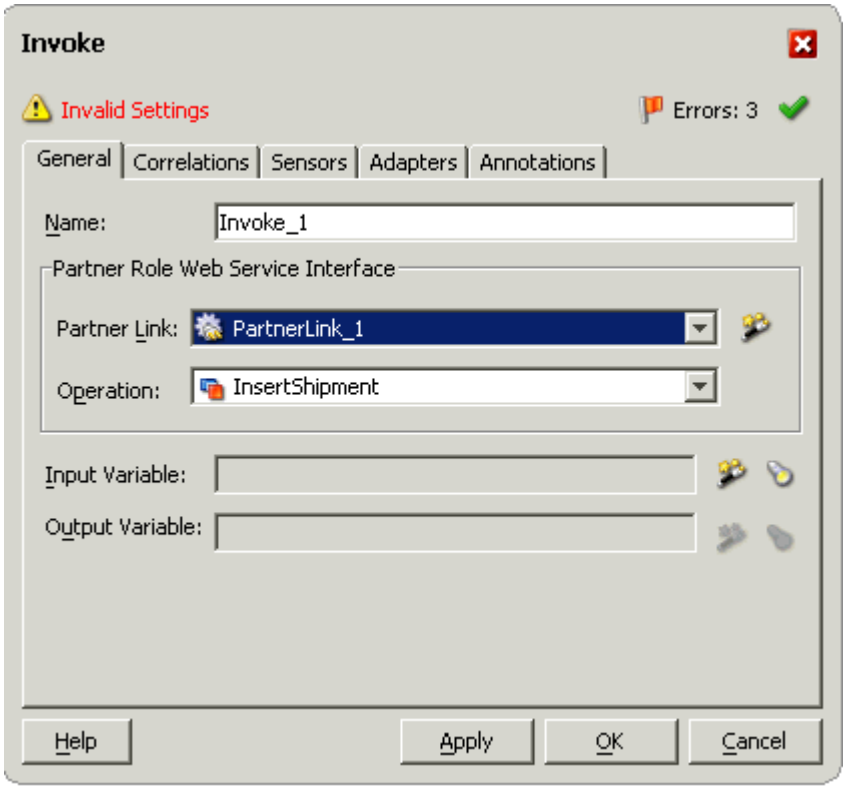
1. Drag and drop the **Invoke** activity into the process map window.

Adding the Invoke Activity



2. Double-click **Invoke** in the process map to open the Invoke dialog box. The **General** tab is selected by default.

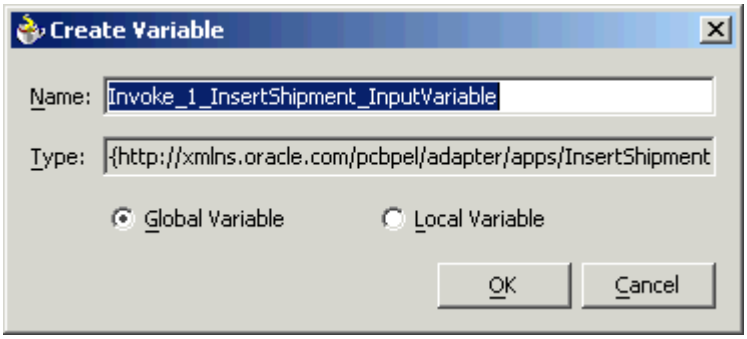
Configuring the Invoke Activity



3. In the **Partner Link** box, select the partner link to invoke. This is the partner link that you configured in the previous section. The **Operation** is automatically selected, depending on the concurrent program that you chose when configuring the partner link.

4. Click the **Create** icon next to the **Input Variable** field. Enter a descriptive name for the variable in the Create Variable dialog box that appears. You can also accept the default name. Click **OK**.

Creating the Input Variable



5. Click the **Create** icon next to the **Output Variable** field. Enter a descriptive name for the variable in the Create Variable dialog box that appears. You can also accept the default name. Click **OK**.

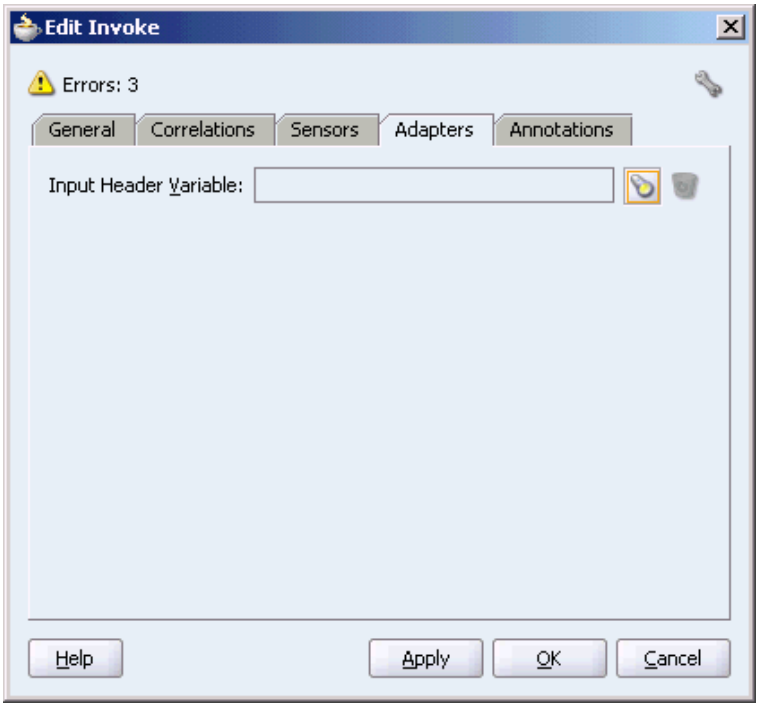
6. In the Invoke dialog box, click **Apply**, and then click **OK**.

   Note: You can define an **Input Header Variable** under the **Adapters** tab of the Invoke dialog box. This variable can be used to provide context information for Oracle Applications in supporting multiple languages and multiple organization setups.

   To Create the Header Variable:

   1. Click on Adapters tab in the Edit Invoke dialog box and click **Browse Variable...** icon for the Input Header Variable field.

Specifying Input Header Variable

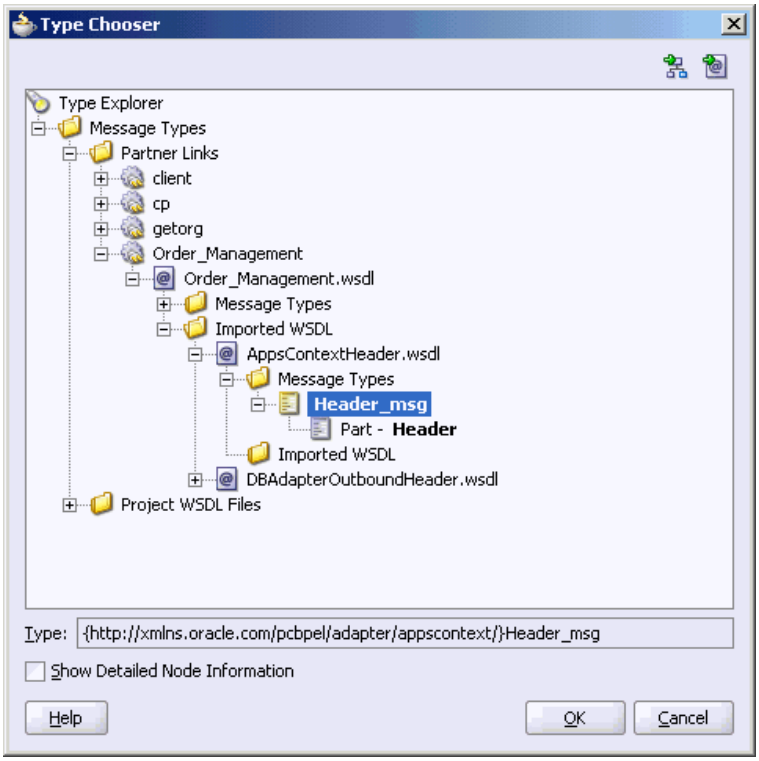


2. In the Variable Chooser dialog box, right-click on the Variables node and select Create Variable option from the menu.

3. Select **Message Type** and click **Browse Message Type...** icon to open Type Chooser dialog box.

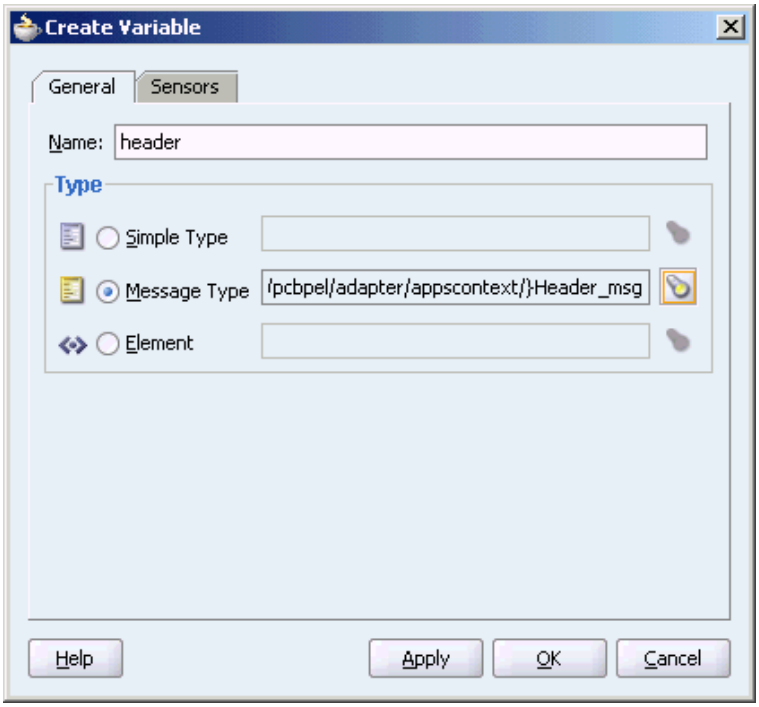
4. Expand the Partner Link node and locate the Header_msg node `{http://xmlns.oracle.com/pcbpel/adapter/appscontext/}Header_msg` for your partner link. The Header_msg node should be under the path, Partner Links node >*Your Partner Link WSDL* > Imported WSDL > AppsContextHeader.wsdl > Message Types > Header_msg.

Declaring Header Variable



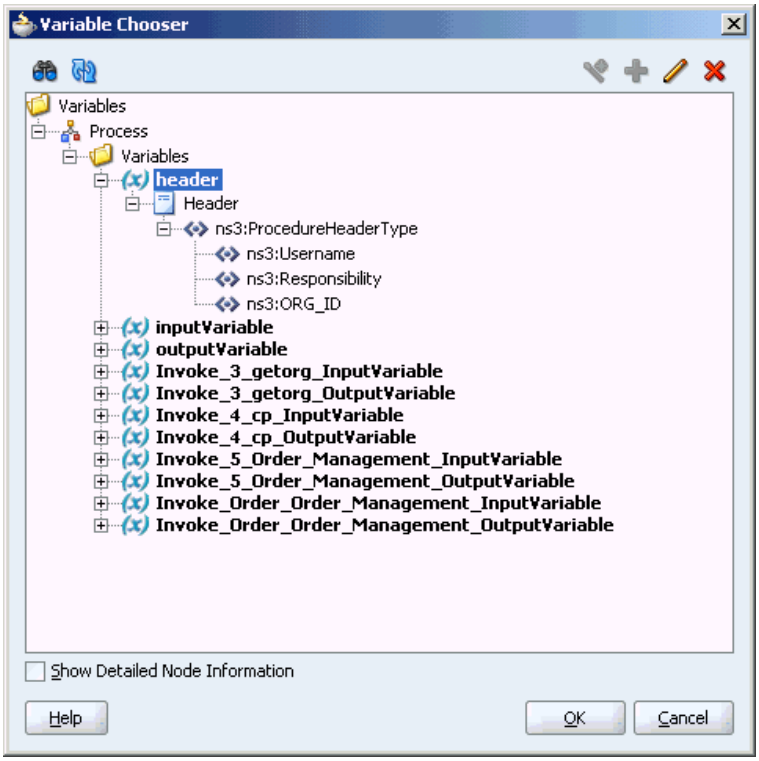
5. Click **OK** to return to the Create Variable dialog box with your selected message type populated.

Populating Selected Header Variable



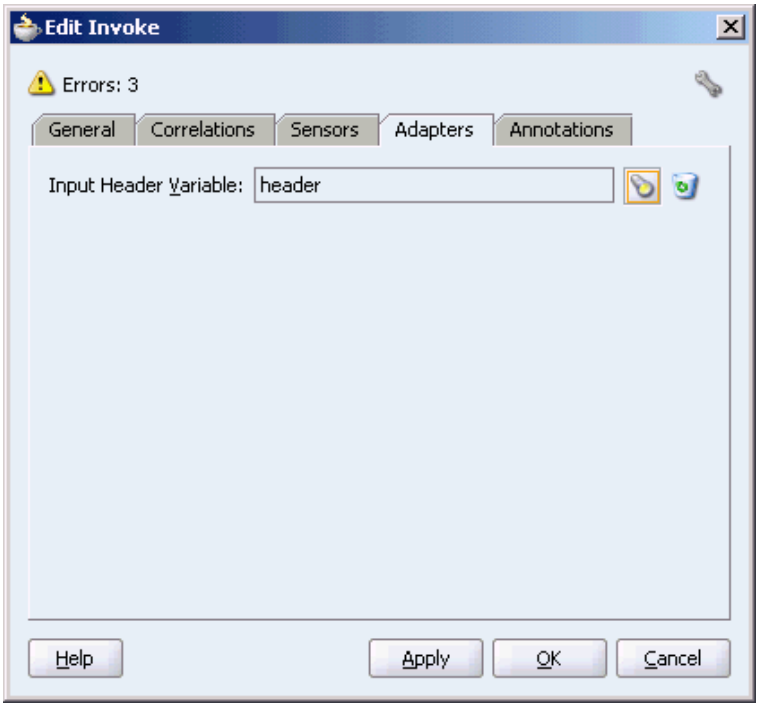
6. To view your header variable with Username, Responsibility, and ORG_ID, click **Browse Message Type...** icon to open Variable Chooser dialog box. Locate the header variable to view the variable hierarchical structure with these three parameters needed for applications context.

Viewing Header Variable Structure



7. Click **OK** to return to the Edit Invoke dialog with the selected header variable populated for the Input Header Variable field. Click **Apply** to complete the header creation.

Populating Input Header Variable



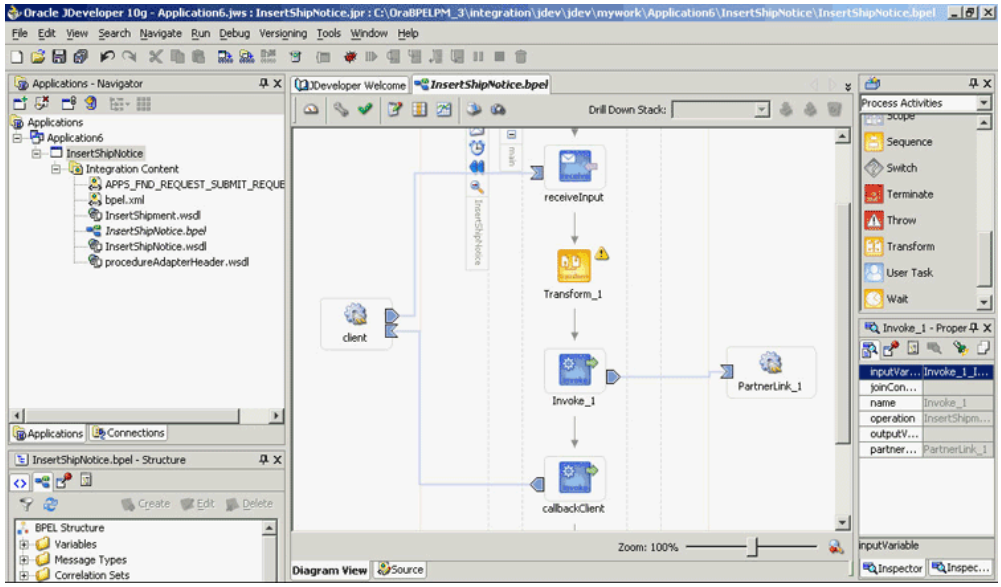
Configuring the Transform Activity

The Transform activity can be used to configure the parameters for the input and output variable. The Transform activity can also be used if variable values need to be transformed before updating them in Oracle Applications.

To configure the Transform activity:

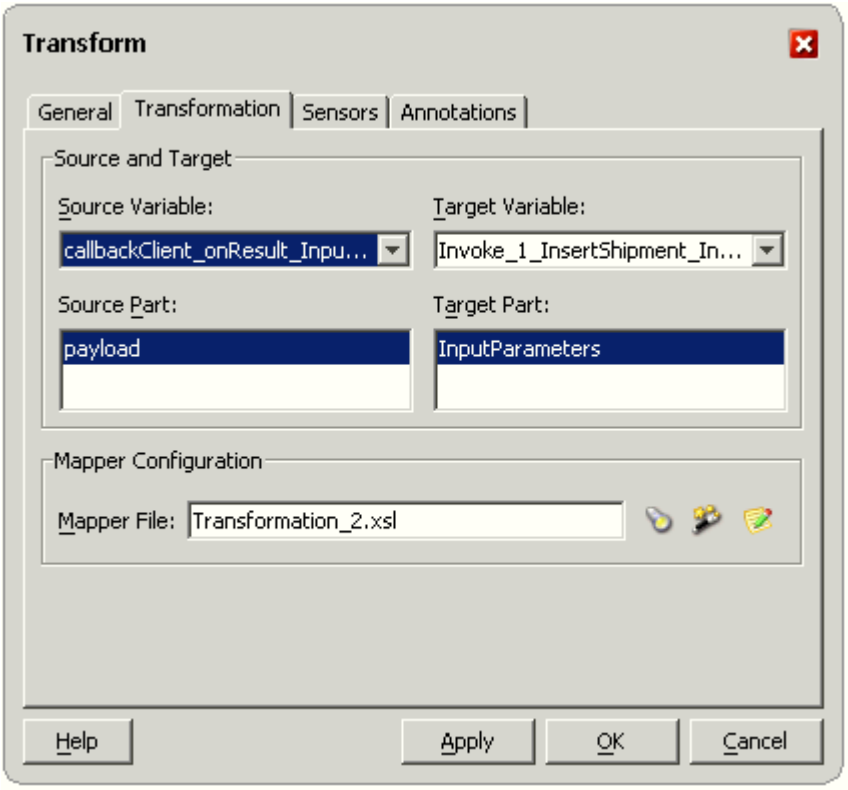
1. Drag and drop the **Transform** activity into the process map window. The **Transform** activity should be placed in between **Receive** and **Invoke**.

Adding the Transform Activity



2. Double-click **Transform** in the process map to open the Transform dialog box. The **Transformation** tab is selected by default.

Configuring the Transform Activity



3. Select the **Source Variable** and **Target Variable** from the respective boxes. Elements are mapped from the **Source Variable** to the **Target Variable**.

4. Select the **Source Part** of the variable from which to map elements. For example, the source part may be a payload schema consisting of a purchase order request.

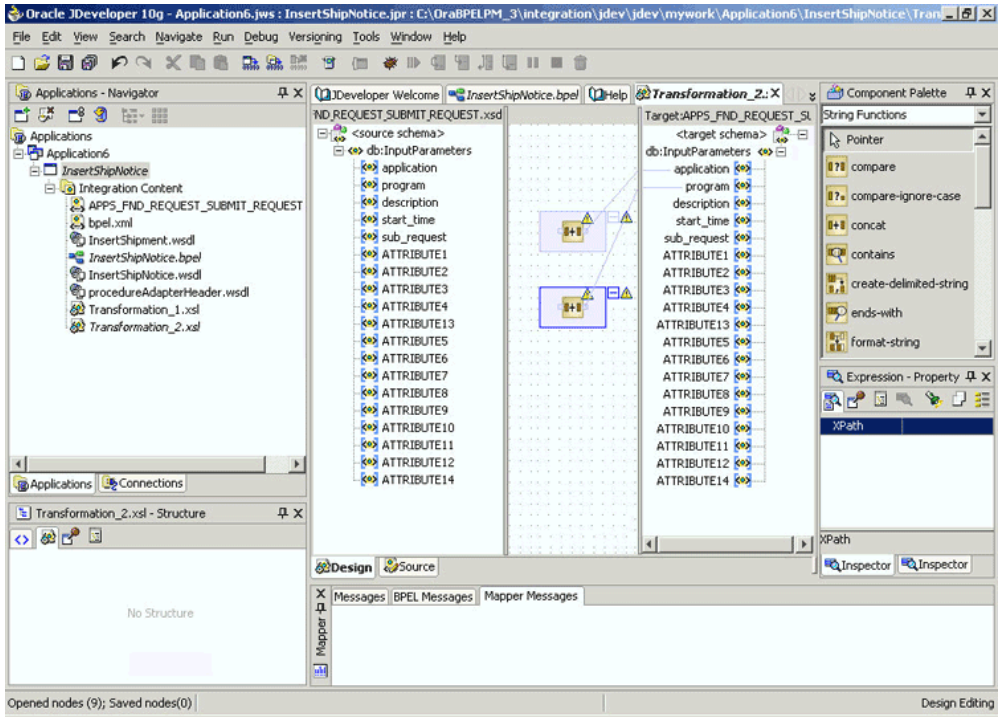
5. Select the **Target Part** of the variable to which to map elements. For example, the target part may be a payload schema consisting of a purchase order acknowledgment.

6. **Mapper File** specifies the file in which you create the mappings using the XSLT Mapper Transformation tool. Click the **Create** icon next to the **Mapper File** field to create a new transformation mapping file.

   The transformation mapping file appears.

7. The **Design** view for the transformation mapping file appears by default.

Editing the Transformation Mapping File

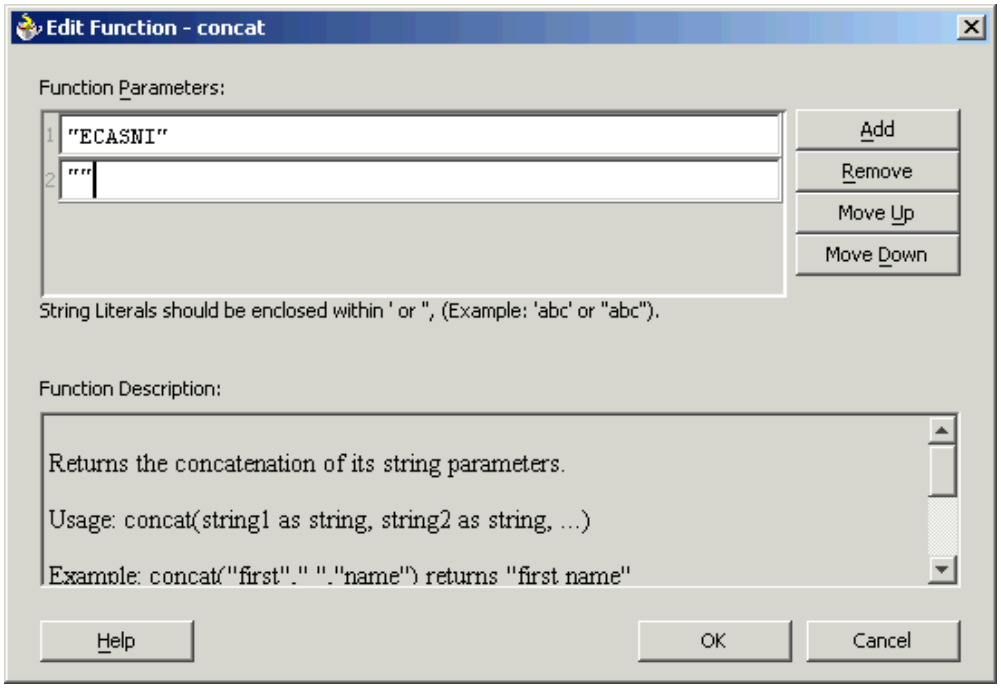


8. You can define the parameter values in the **Design** view. Drag a string function to the **Design** area. Connect the function to the appropriate parameter for which you want to define a value.

   Note: You can use an input parameter value from the source variable, transform it using a string function, and then use it as the input parameter value for the target variable.

9. Double-click the icon for the function. The Edit Function dialog box appears.

Supplying the Function Parameters



10. Repeat steps 8 and 9 for all the parameters that you must supply.

    Note: You can use either the Transform activity or the Assign activity to configure the variables. If the number of mappings to be configured is less, say three or four, then you can use the Assign activity. The Transform activity is more suitable for situations where the number of mappings is large or where variable transformation is required.

Run-Time Tasks for Concurrent Programs

After designing the BPEL process, the next step is to deploy, run and monitor it.

1. Deploy the BPEL process., page 4-28
2. Test the BPEL process., page 4-29
3. Verify records in Oracle Applications., page 4-31

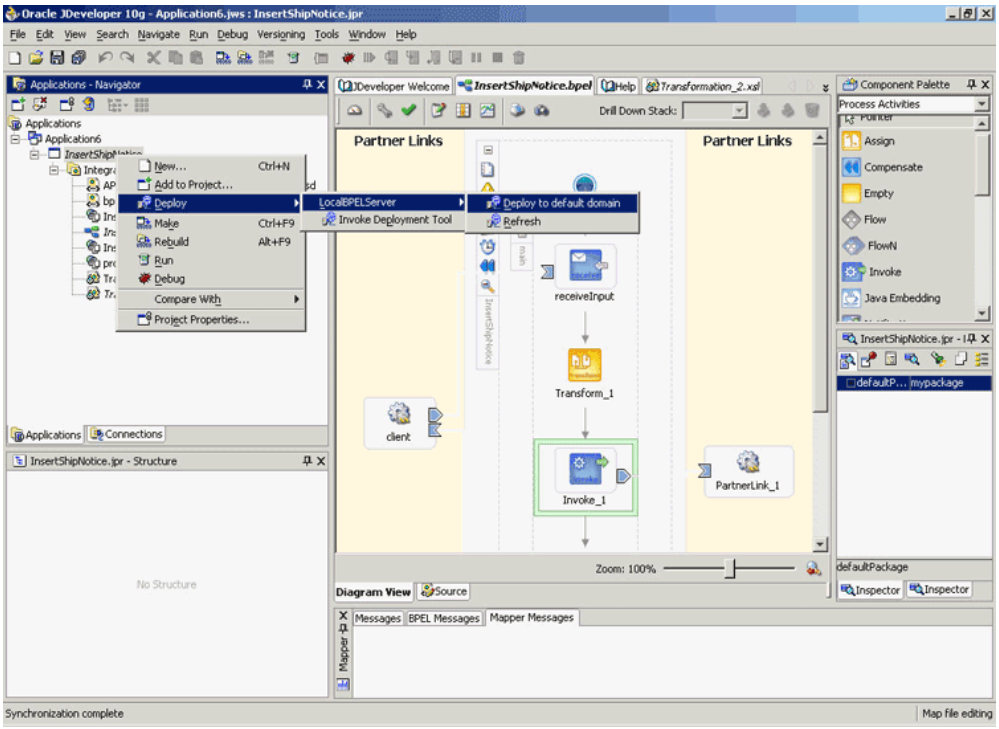
Deploying the BPEL Process

You must deploy the BPEL process before you can run it. The BPEL process is first compiled, and then deployed to the BPEL server.

To deploy the BPEL process:

1. Select the BPEL project in the Applications window.
2. Right-click the project name, then select **Deploy** from the menu that appears.
3. If you are deploying the process on the local BPEL server, select **Local BPEL Server,** followed by **Deploy to Default Domain.**

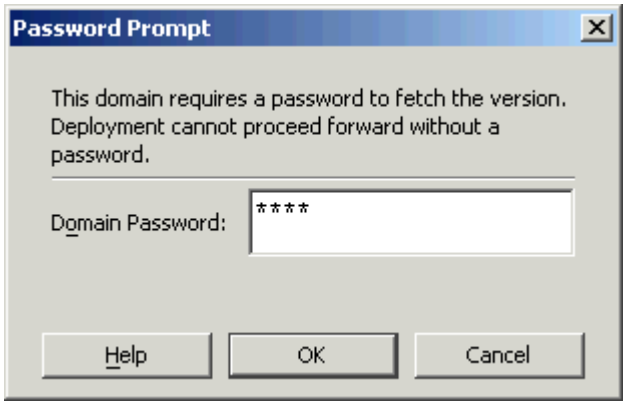
Deploying the BPEL Process



Note: If you want to deploy to a different BPEL server, select **Invoke Deployment Tool.**

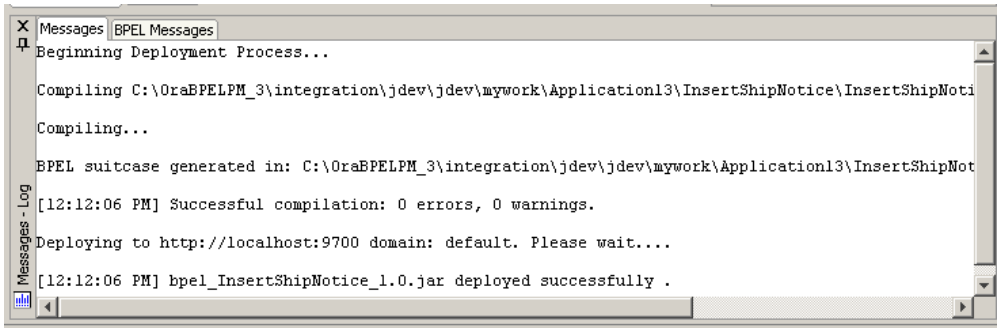
4. The Password Prompt dialog box appears. Enter the password for the default domain in the **Domain Password** field, and then click **OK.**

Specifying the Domain Password



5. The BPEL process is compiled and deployed. You can check the progress in the Messages window.

Messages Window



Testing the BPEL Process

Once the BPEL process is deployed, it can be seen in the BPEL console. You can manage and monitor the process from the BPEL console. You can also test the process and the integration interface by manually initiating the BPEL process.

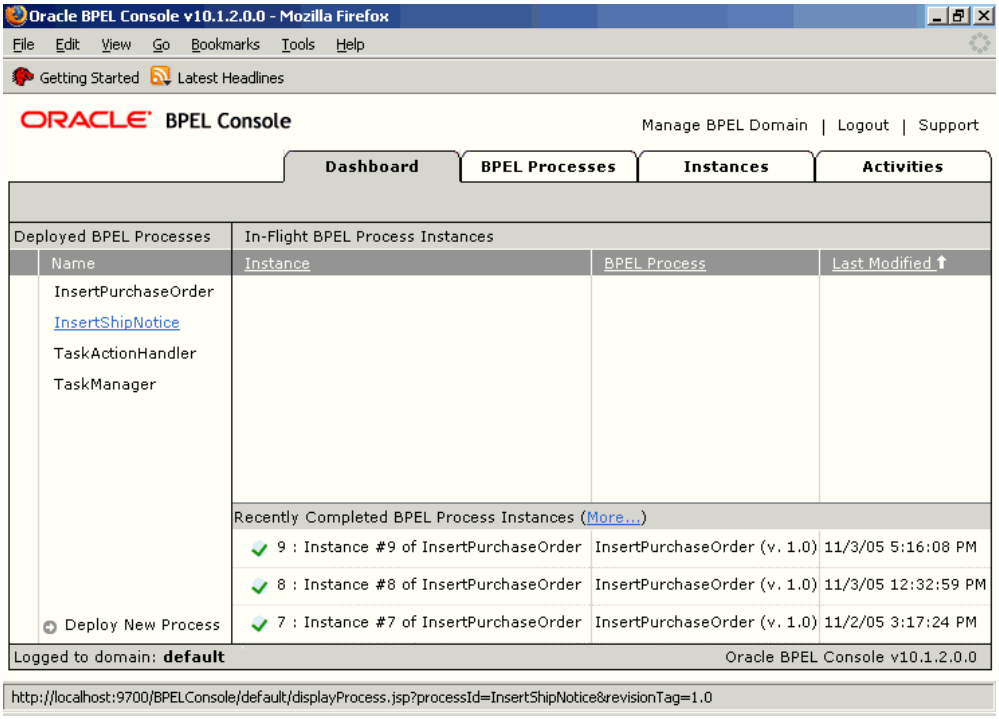
To test the BPEL process:

1. To open the BPEL console, click **Start,** then choose **Programs.** In the Programs menu, select **Oracle -** *ORACLE_HOME*,**Oracle BPEL Process Manager 10.1.3,** then click **BPEL Console.**

2. The BPEL console login page appears. Select **Default** in the **Domain** box. Enter the password for the default domain in the **Password** field, then click **Login**.

3. Oracle BPEL console appears. The list of deployed processes is shown under Deployed BPEL Processes.

Deployed BPEL Processes

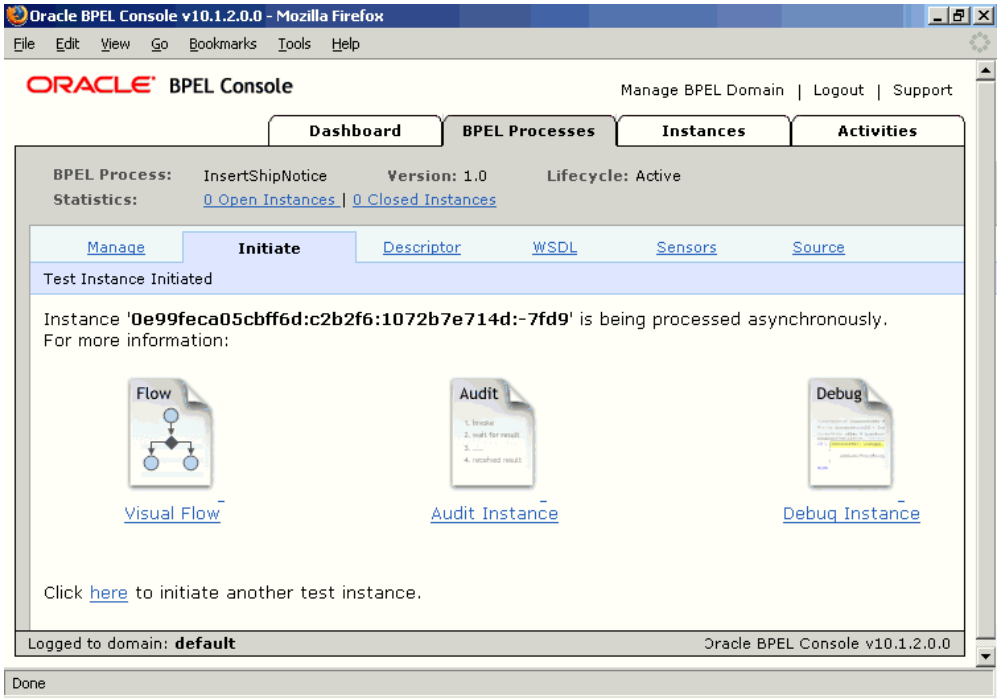


4. Click the BPEL process that you want to initiate. The Initiate page appears. Enter the input string required by the process.

5. Click **Post XML Message** to initiate the process.

6. The BPEL process is now initiated. You can check the process flow by clicking the **Visual Flow** icon.

BPEL Console Initiate Page



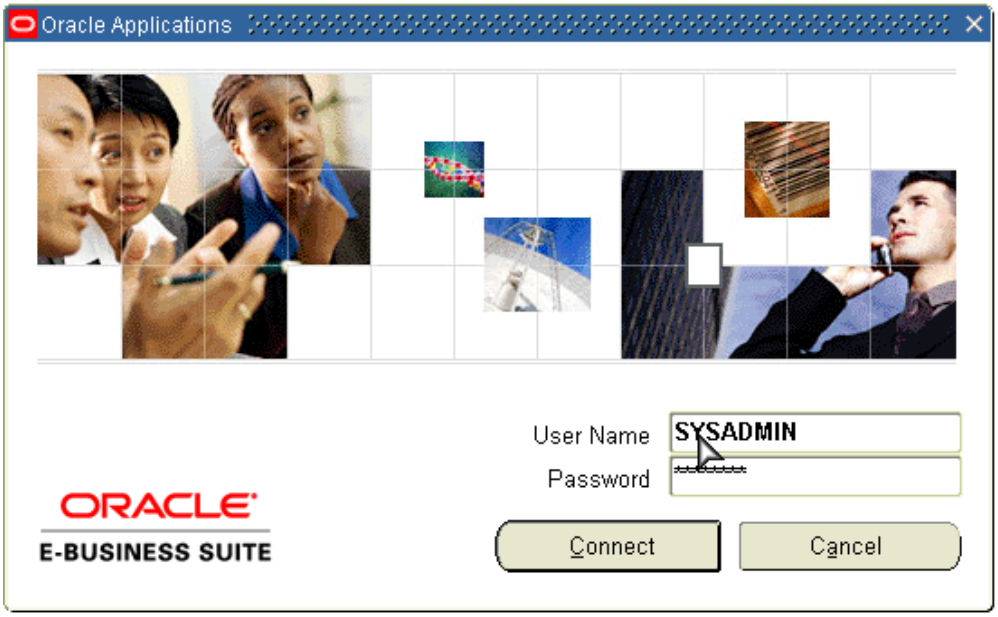
7. The audit trail provides information about the steps that have been executed. The audit trail also records the Request ID that is returned for the transaction. You can check the audit trail by clicking the **Audit Instance** icon.

Verifying Records in Oracle Applications

To verify records in Oracle Applications:

1. Log in to Oracle Applications as the System Administrator.

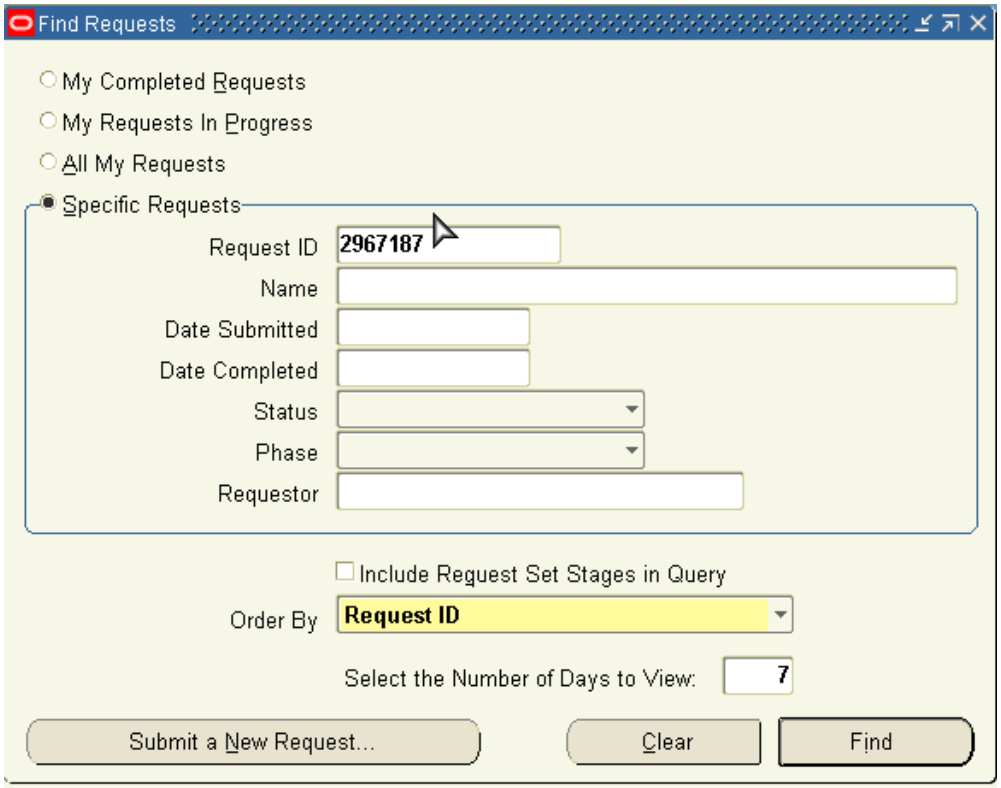
Oracle Applications Login Dialog Box



2. Select **Requests** from the **View** menu.

3. Search for the Request by entering the Request Id that you got from the audit trail, then click **Find.**

Find Requests Dialog Box



4. The Request details are displayed. You can check for details such as the Phase and Status of the request.

5. If the **Status** of the request is `Complete`, you can also query the appropriate table in Oracle Applications to search for the relevant records that have been inserted.

Querying Oracle Applications for a Record

```
SQL> select     distinct(shipment_num)  from    rcv_headers_interface   where  s
hipment_num     =       'S1722427';

SHIPMENT_NUM
------------------------------
S1722427

SQL>
```

5

Using Tables and Views

This chapter covers the following topics:

- Overview of Tables and Views
- Design-Time Tasks for Interface Tables
- Creating a New BPEL Project
- Adding a Partner Link
- Configuring the Invoke Activity
- Configuring the Assign Activity
- Run-Time Tasks for Interface Tables
- Deploying the BPEL Process
- Testing the BPEL Process
- Design-Time Tasks for Views
- Creating a New BPEL Project
- Adding a Partner Link
- Configuring the Invoke Activity
- Configuring the Assign Activity
- Run-Time Tasks for Views
- Deploying the BPEL Process
- Testing the BPEL Process
- Troubleshooting and Debugging

Overview of Tables and Views

OracleAS Adapter for Oracle Applications uses interface tables to insert and update data in Oracle Applications, and uses views to retrieve data from Oracle Applications.

This chapter describes the following interfaces:

- Interface Tables
- Views

Interface Tables

OracleAS Adapter for Oracle Applications use interface tables to insert data in Oracle Applications. For example, by using interface tables, you can insert a purchase order into Oracle Applications to generate the sales order automatically. Data is never loaded directly into Oracle Applications base tables. Instead, data is first loaded into interface tables, and then Oracle-supplied concurrent programs move data from interface tables to base tables. This ensures that all business logic and processing is handled using Oracle components.

OracleAS Adapter for Oracle Applications use open interface tables to integrate with Oracle Applications through direct database access. The OracleAS Adapter for Oracle Applications inserts data into the open interface tables. These interface tables can be used only for insert operations and support only an inbound connection into Oracle Applications.

Interface tables are intermediate tables into which the data is inserted first. Once the data gets inserted into the interface tables, the data is validated, and then transferred to the base tables. Base tables are real application tables that reside in the application database. The data that resides in the interface tables is transferred to the base tables using concurrent programs. A concurrent program is an instance of an execution file. Concurrent programs are scheduled in Oracle Applications to move data from interface tables to base tables. These programs perform the application-level checks and run validation before inserting data into base tables.

Views

OraclesAS Adapter for Oracle Applications uses views to retrieve data from Oracle Applications. For example, by using views, you can retrieve information about your customers from the required tables in Oracle Applications.

OracleAS Adapter for Oracle Applications uses views to retrieve data from Oracle Applications. Views allow only simple definition. By using views, you can get synchronous data access to Oracle Applications. In OracleAS Adapter for Oracle Applications, views are created on base tables as well as interface tables. These views can be used *only* for select operations.

In the Oracle Applications 11.5.10 release, you cannot work on multiple views. A work around to address this would be to create a view that spans multiple views.

Design-Time Tasks for Interface Tables

This section describes how to configure the OracleAS Adapter for Oracle Applications to use interface tables. It describes the steps to configure OracleAS Adapter for Oracle Applications using the Adapter Configuration Wizard in Oracle JDeveloper.

Prerequisites to Configure Interface Tables

- Define primary keys on all the interface tables being used.
- Define parent-child relationships among all the interface tables used.

Following is a list of the procedures required to accomplish the design-time tasks.

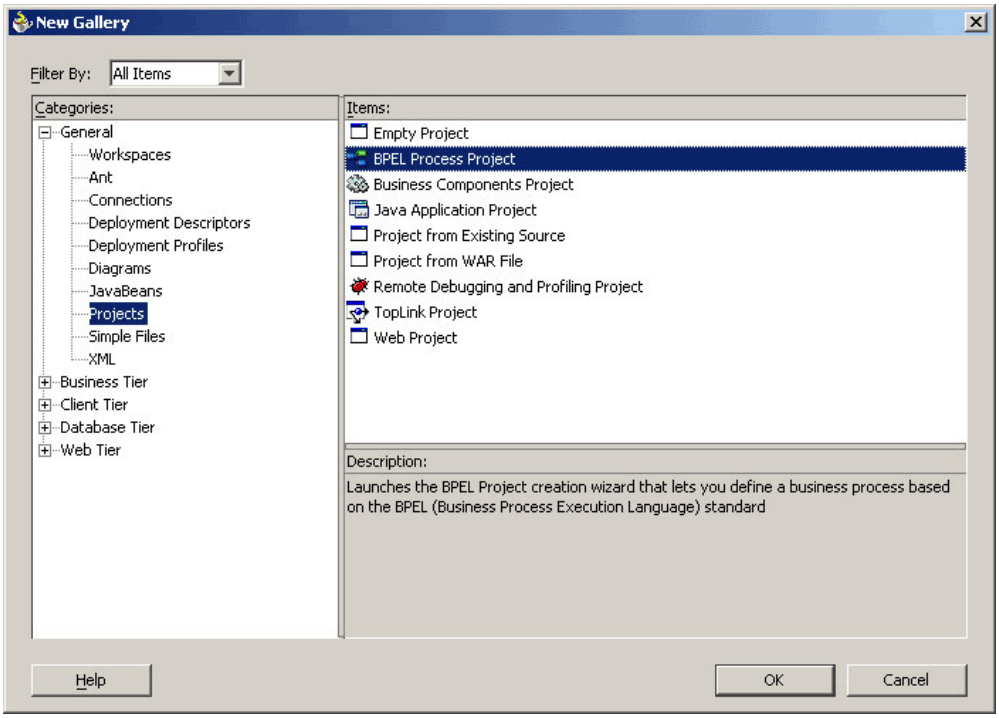
1. Create a new BPEL project., page 5-3
2. Add a partner link., page 5-6
3. Configure the Invoke activity., page 5-19
4. Configure the Assign activity., page 5-22

Creating a New BPEL Project

To create a new BPEL project:

1. Open JDeveloper BPEL Designer.
2. From the **File** menu, select **New**. The New Gallery dialog box appears.
3. Select **All Items** from the **Filter By** list. This displays a list of available categories.
4. Expand the **General** node, and then select **Projects**.
5. Select **BPEL Process Project** from the **Items** list.

Creating a New BPEL Process Project

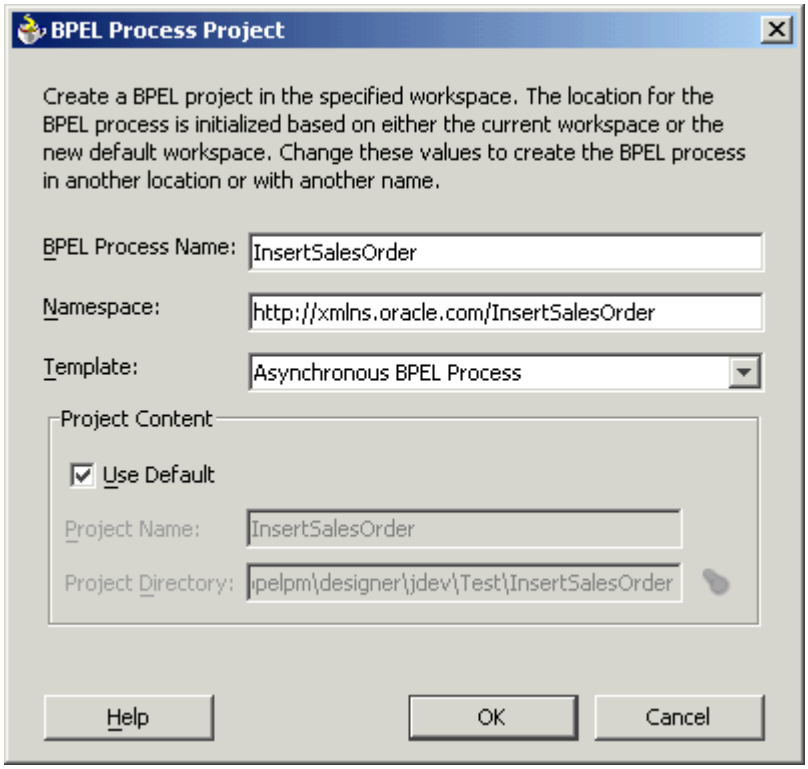


6. Click **OK**. The BPEL Process Project dialog box appears.

7. In the **BPEL Process Name** field, enter a descriptive name. For example, `InsertSalesOrder`.

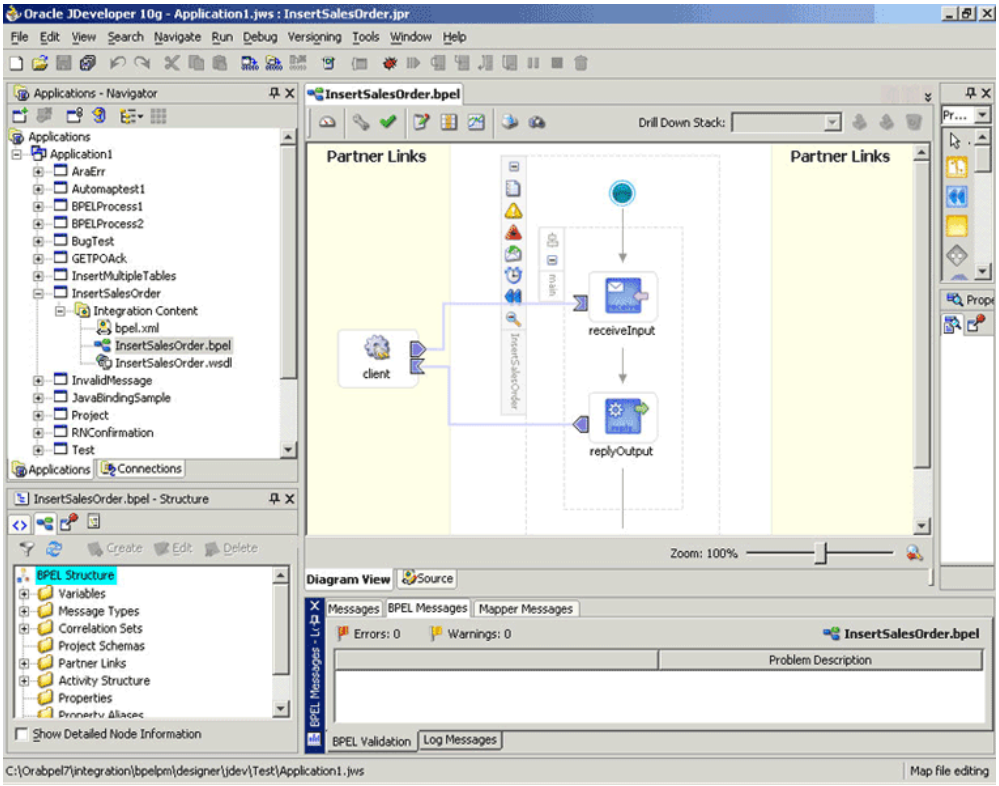
8. From the **Template** list, select **Asynchronous BPEL Process**. Keep the default selection for Use Default in the Project Content section.

Specifying a Name for the New BPEL Process Project



9. Click **OK**. A new asynchronous BPEL process, with the required source files including `bpel.xml`, `InsertSalesOrder.xml`, and `InsertSalesOrder.wsdl`, is created.

New BPEL Process Project



Adding a Partner Link

This section describes how to add a partner link to your BPEL process. A partner link defines the link name, type, and the role of the BPEL process that interacts with the partner service.

To add a partner link:

1. Drag and drop **PartnerLink** into the border area of the process diagram. The Create Partner Link dialog box appears.

2. Click **Define Adapter Service** in the WSDL Settings section. The Adapter Configuration Wizard appears.

3. Click **Next**. The Adapter Type dialog box appears.

4. Select **Oracle Applications** to specify the adapter you want to configure.

Selecting OracleAS Adapter for Oracle Applications

5. Click **Next**. The Service Name dialog box appears.

6. Enter the following information:

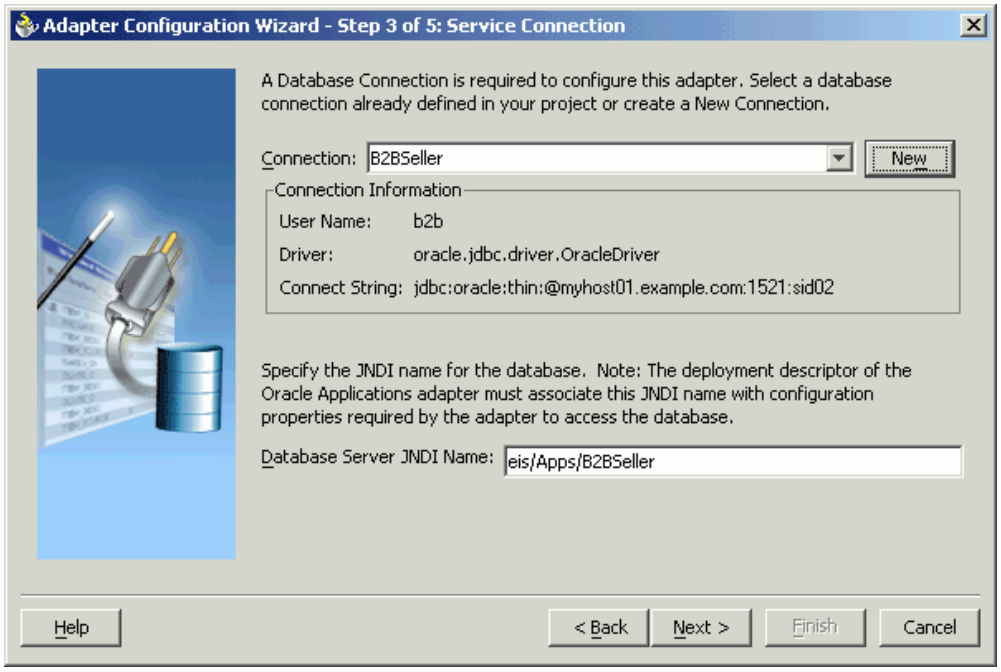
   1. In the **Service Name** field, enter a service name.

   2. In the **Description** field, enter a description for the service. This is an optional field.

Specifying the Service Name

7. Click **Next**. The Service Connection dialog box appears.

Specifying a Database Connection



8. Select an existing database connection from the Connection list. Alternatively, you can click **New** to define a database connection. The Create Database Connection Wizard appears.

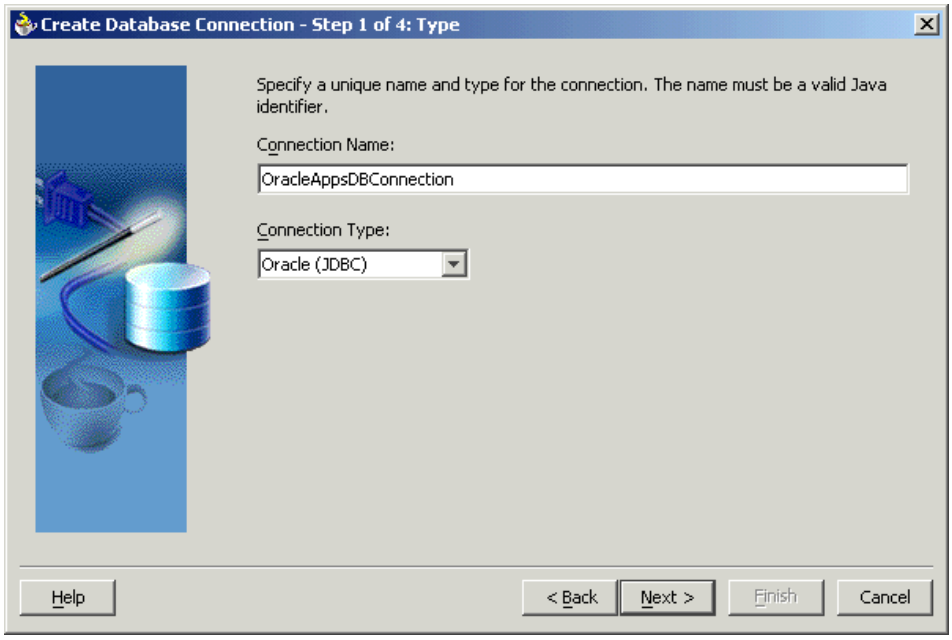
   Note: You need to connect to the database where Oracle Applications is running.

9. Enter the following information in the Type dialog box:

   1. In the **Connection Name** field, specify a unique name for the database connection.

   2. From the **Connection Type** list, select the type of connection for your database connection.

Specifying the Connection Name and Type of Connection



10. Click **Next**. The Authentication dialog box appears.

11. Enter information in the following fields:

    1. In the **UserName** field, specify a unique name for the database connection.

    2. In the **Password** field, specify a password for the database connection.

12. Click **Next**. The Connection dialog box appears.

13. Enter information in the following fields:

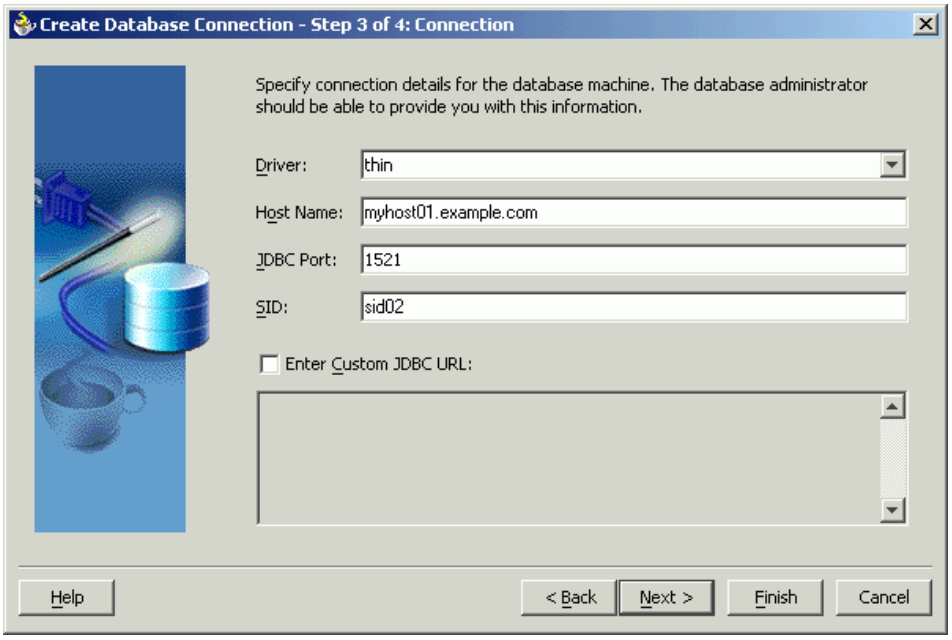
    1. From the **Driver** list, select **Thin**.

    2. In the **Host Name** field, specify the host name for the database connection.

    3. In the **JDBC Port** field, specify the port number for the database connection.

    4. In the **SID** field, specify a unique SID value for the database connection.

Specifying the New Database Connection Information



14. Click **Next**. The Test dialog box appears.

15. Click **Test Connection** to confirm that the specified information establishes a connection with the database.

16. Click **Next**. The Service Connection dialog box returns, providing a summary of the database connection.

17. The JNDI (Java Naming and Directory Interface) name corresponding to the database connection appears automatically in the **Database Server JNDI Name** field. Alternatively, you can specify a JNDI name.

    Note: When you specify a JNDI name, the deployment descriptor of the Oracle Applications adapter must associate this JNDI name with configuration properties required by the adapter to access the database.

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 for production.

Note: For more information about JNDI concepts, refer to *Oracle Application Server Adapter Concepts.*

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

    Once you have completed creating a new connection for the service, you can add an interface table by browsing through the maps available in Oracle Applications.

19. Click **Next** in the Service Connection dialog box.

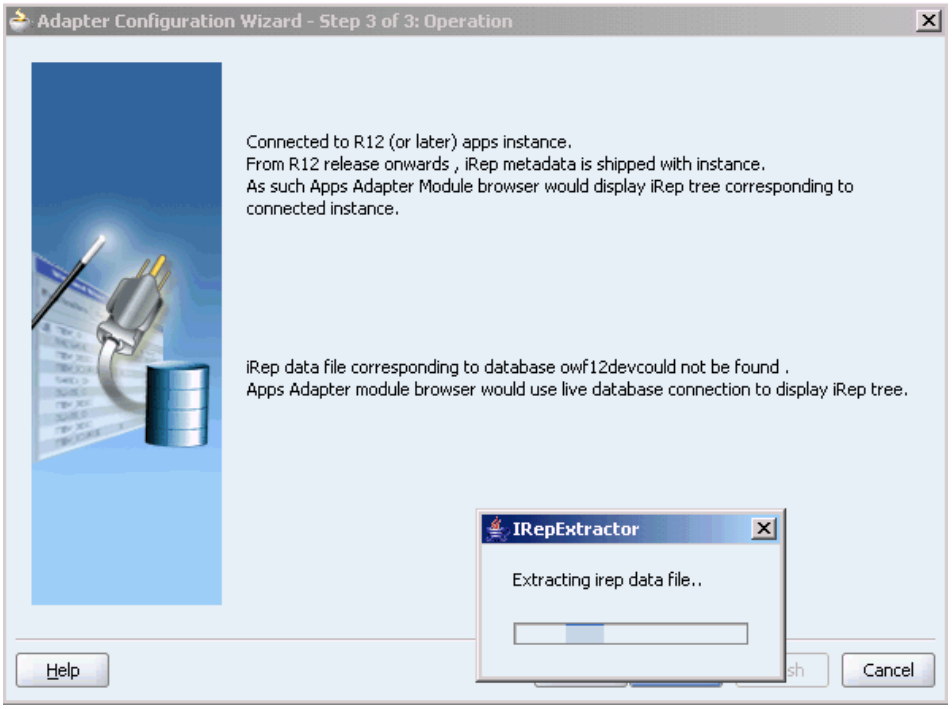
    For Oracle E-Business Suite Release 12:

    If you are connecting to Oracle E-Business Suite Release 12, then the **IREP File not present** dialog box appears indicating that OracleAS Adapter could not find the Oracle Integration Repository data file corresponding to the database you are connecting to Oracle Applications in your workspace. Absence of the data file would make browsing or searching of Integration Repository tree considerably slow. You can choose to extract the data file and create a local copy of the Integration Repository data file. Once it is created successfully, OracleAS Adapter will pick it up automatically next time and retrieve data from your local Integration Repository.

    You can select one of the following options:

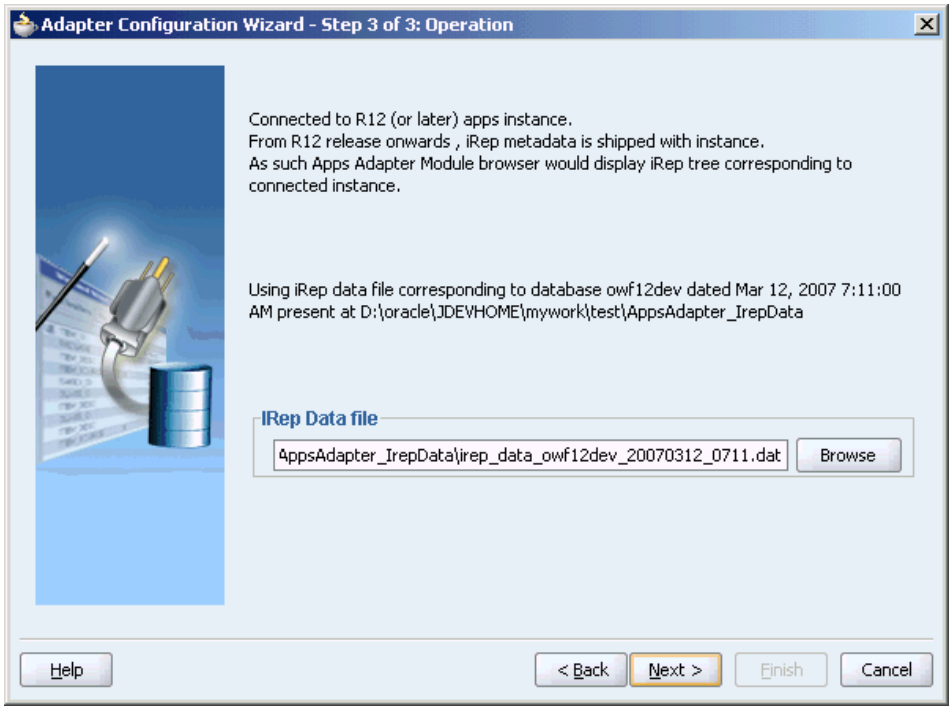
    - Click **Yes** to extract the Integration Repository data file.

Extracting Integration Repository Data File



After the system successfully creates a local copy of the Integration Repository data file, next time when you connect to the database, you will find the **IRep Data File** field appears in the Operation dialog box indicating where your local copy exists with the creation date and time as part of the file name.

Using the Local Integration Repository Data File



- Click **No** to query the Integration Repository data file from the live database you are connecting to display the Integration Repository tree.

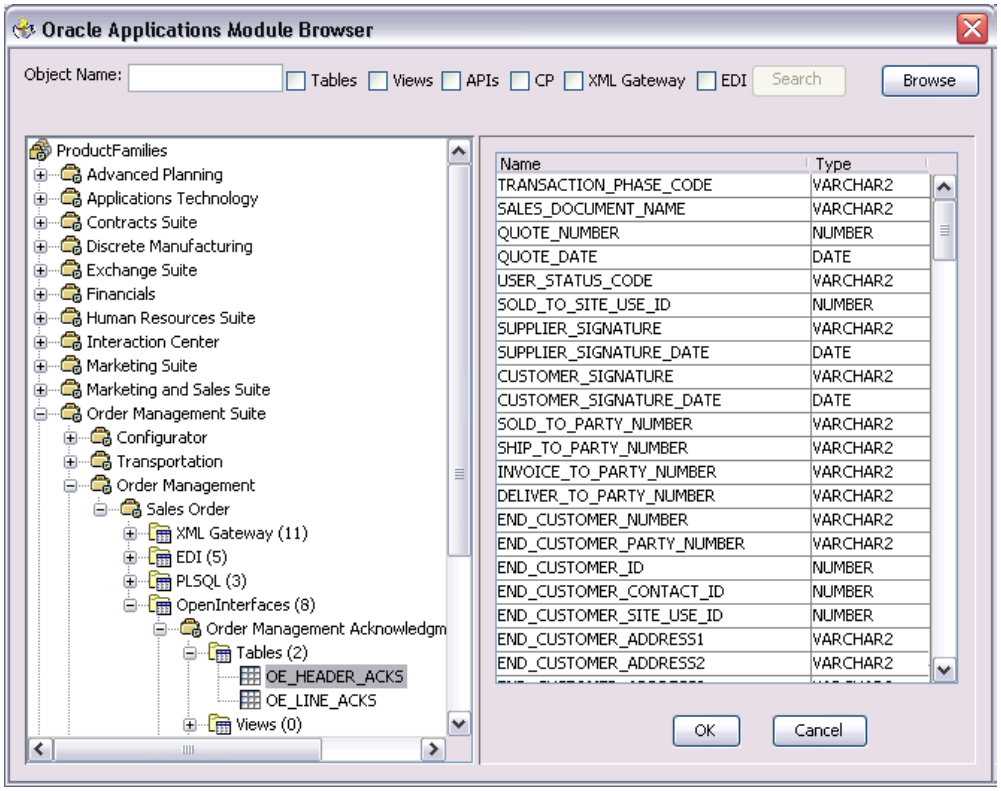
  Note: It is highly recommended that you create a local copy of the Integration Repository data file so that OracleAS Adapter will query the data next time from the local copy in your workspace to enhance the performance.

For Oracle E-Business Suite pre-Release 11.5.10:

If you are connecting to a pre-11.5.10 Oracle Applications instance, you must select the interface type in the Adapter Configuration Wizard. Select **Tables/Views/APIs/Concurrent Programs** to proceed.

20. Click **Get Object** to open the Oracle Applications Module Browser.

Adding a Table from the Oracle Applications Module Browser



Oracle Applications Module Browser includes the various product families that are available in Oracle Applications. For example, Applications Technology or Order Management Suite are product families in Oracle Applications. The product families contain the individual products. For example, Order Management Suite contains the Order Management product. The product contains the business entities associated with the product. For example, the Order Management product contains the Sales Order business entity.

Business entities contain the various application modules that are exposed for integration. These modules are grouped according to the interface they provide. Tables can be found under the OpenInterfaces category.

21. Select a table, then click **OK**.

    Note: You can also search for a table by entering the name of the program in the **Object Name** field. Select the **Tables** check box, then click **Search**.

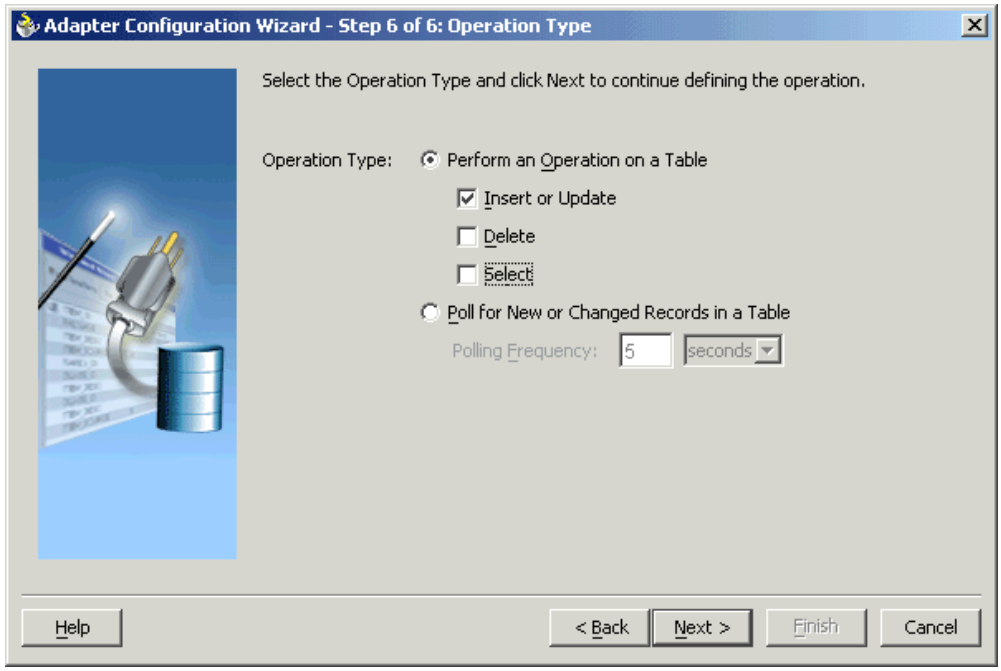
    The table is added to the Operation Objects.

22. Repeat the preceding two steps to add more tables.

23. Click **Next** in the Operation dialog box. The Operation Type dialog box appears.

24. Select **Perform an Operation on a Table**, and then select **Insert or Update.**

Selecting the Type of Operation

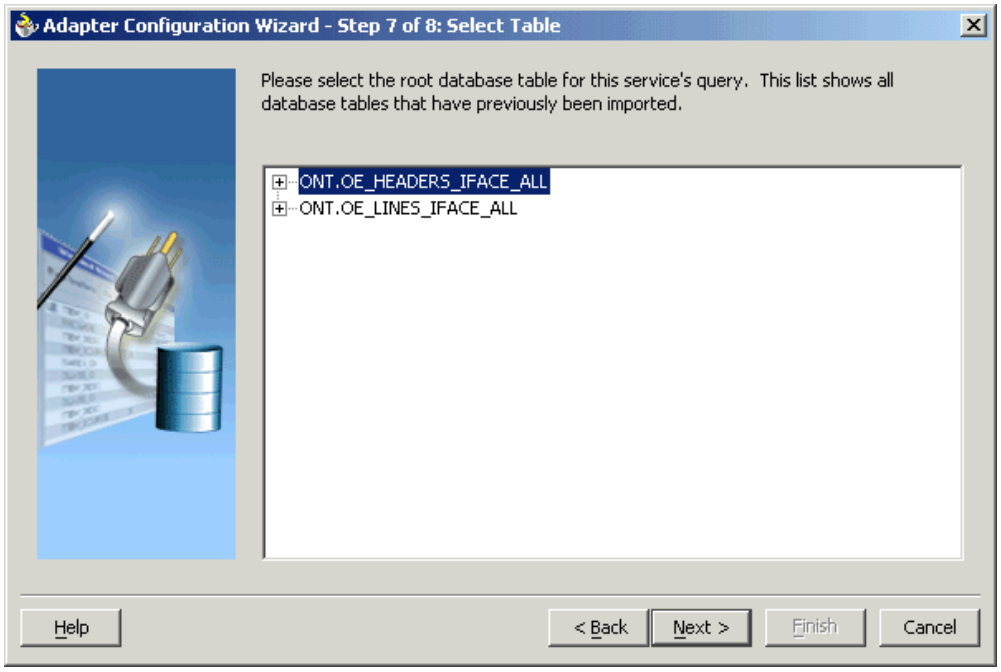


Note: You can only insert data into the interface tables even though the Delete and Select options are enabled.

25. Click **Next.** The Select Table dialog box appears.

26. Select the root database table in the Select Table dialog box, which displays the tables that have been previously imported in this JDeveloper project (including tables that were imported for other partner links). This enables you to reuse configured table definitions in multiple partner links.

Selecting a Root Table

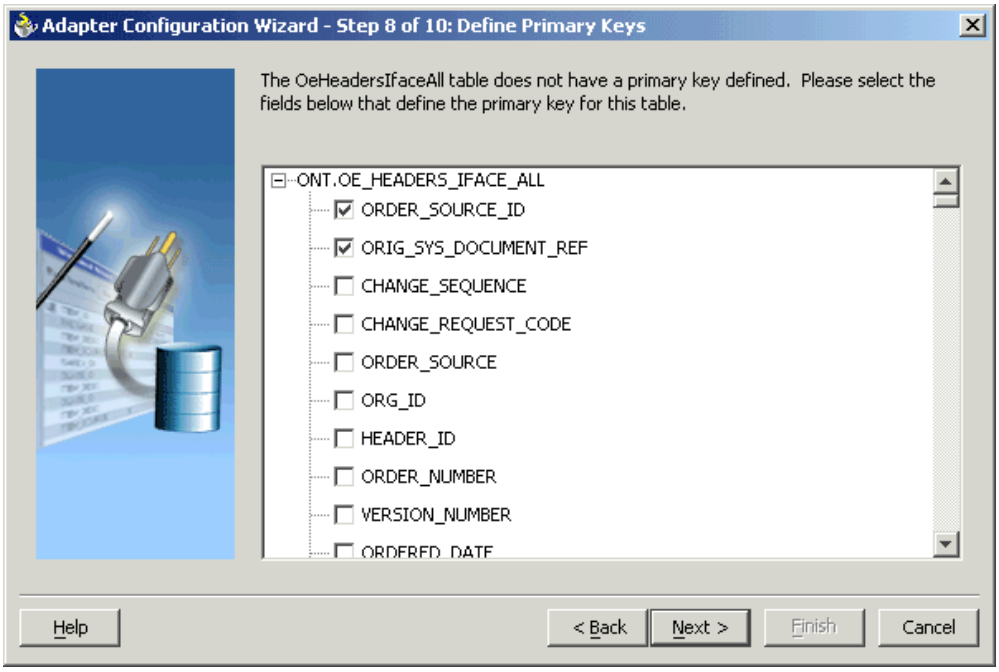


Note: If the required root database table for the operation is not imported, then click **Import Tables**. In addition, you can reimport a table and overwrite the previously configured table definition. However, in this case, you will lose any custom relationships that you may have defined on that table as well as any `WHERE` clause if a root table is imported.

27. Click **Next**. The Define Primary Keys dialog box appears.

28. Select the primary key fields. You can also select multiple fields.

Selecting the Primary Key



29. Click **Next**. The Relationships dialog box appears.

30. Click **Create Relationship** to define a new relationship. The Create Relationship dialog box appears.

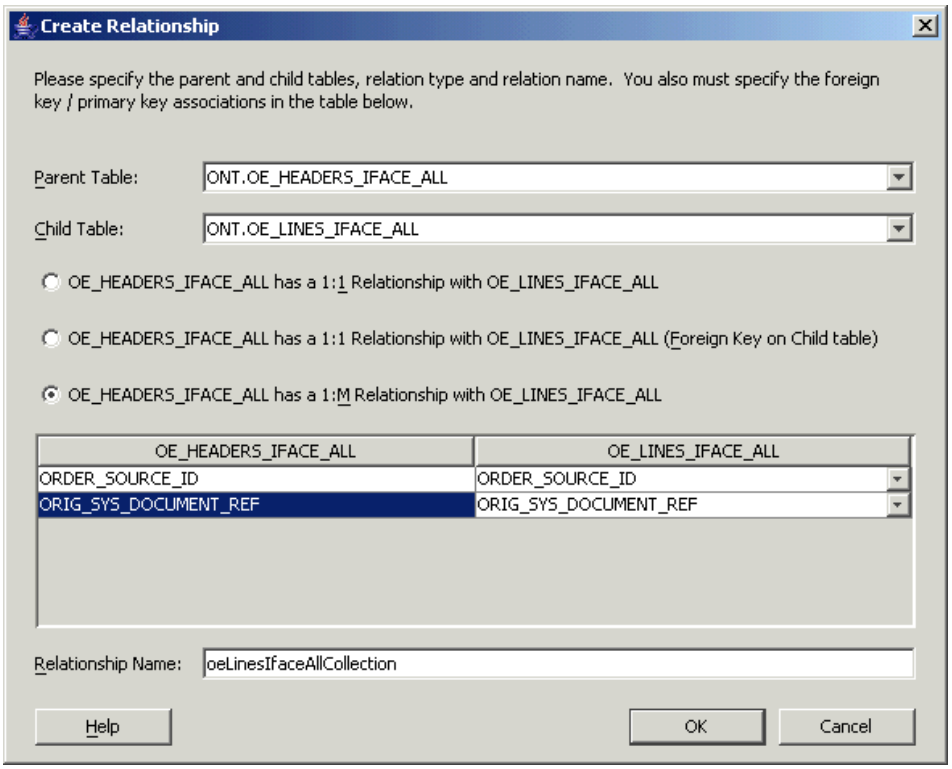
    Note: If foreign key constraints between tables already exist in the database, then two relationships are created automatically while importing tables. One of the relationships is 1:M relationship from the source table, which is the table containing the foreign key constraints, to the target table. The other relationship is a 1:1 back pointer from the target table to the source table.

31. Specify the following information to create a new relationship:

    1. Select the parent and child tables.
    2. Select the mapping type (1:M, 1:1, or 1:1 with Foreign Key on Child Table).
    3. Associate the foreign key fields to the primary key fields.
    4. Enter a name for the relationship you are creating. It is optional to specify a name. By default, a name is generated for the relationship.

Note: You can select only those tables as parent tables that can be accessed from the root table.

Defining Relationships Between Parent and Child Tables



32. Click **OK**. The Relationships dialog box appears.

33. Click **Next**, and then click **Finish** to complete the process of configuring OracleAS Adapter for Oracle Applications.

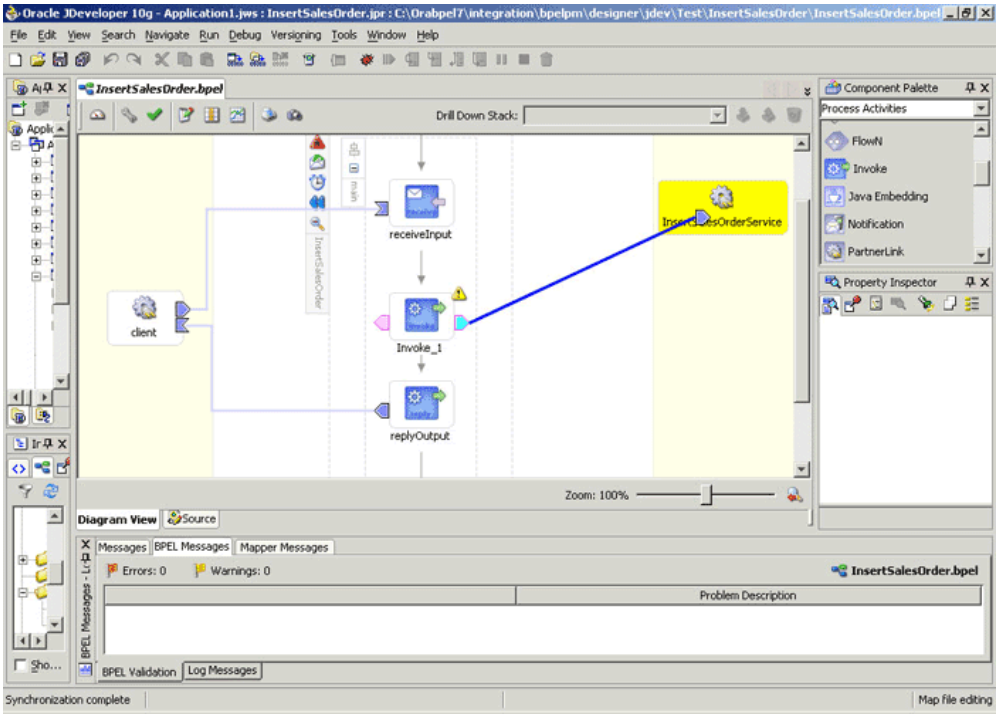
    After adding and configuring the partner link, the next task is to configure the BPEL process.

Configuring the Invoke Activity

To configure the Invoke activity:

1. Drag **Invoke** from the Component palette and drop it at the location where you want to insert the invoke activity in your BPEL process.

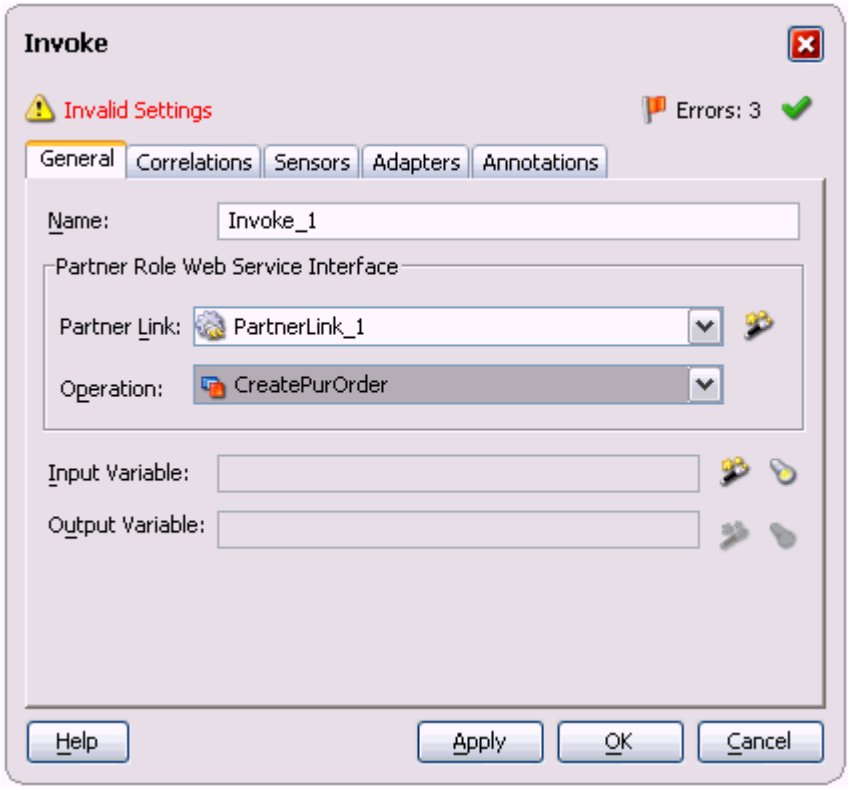
Dragging an Invoke Activity



The Edit Invoke dialog box appears.

2. Double-click **Invoke** in the process map to open the Invoke dialog box. The General tab is selected by default.

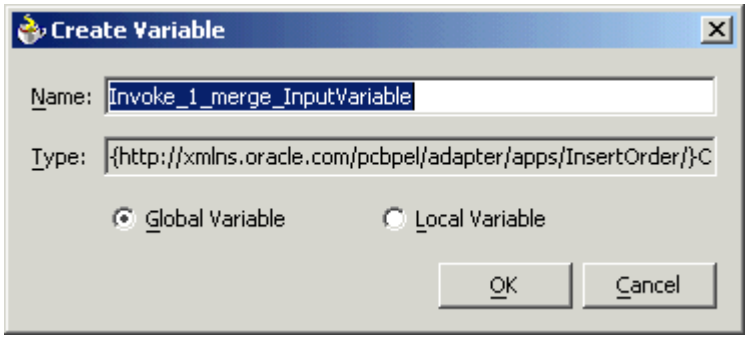
Configuring the Invoke Activity



3. In the **Partner Link** box, select the partner link to invoke. This is the partner link that you configured in the previous section.

4. Click the **Create** icon next to the **Input Variable** field. Enter a descriptive name for the variable in the Create Variable dialog box that appears. You can also accept the default name. Click **OK.**

Creating a Variable



5. In the Invoke dialog box, click **Apply**, and then click **OK**.

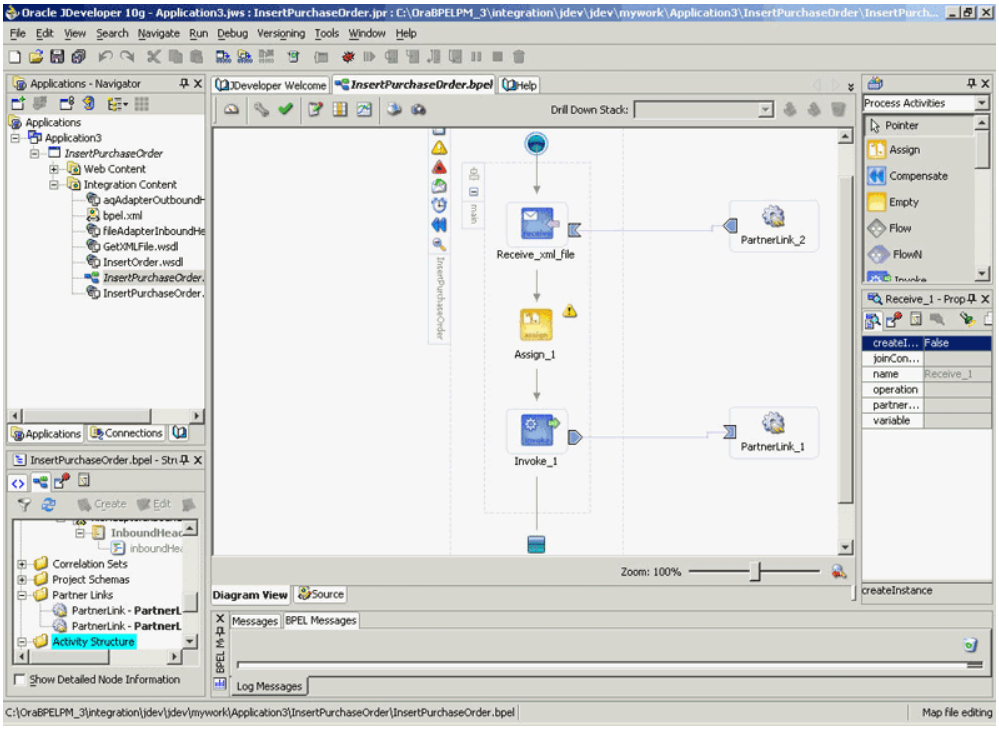
Configuring the Assign Activity

The next task is to add an Assign activity to the process map. This is used to provide values to the input variables.

To configure the Assign activity:

1. Drag and drop the Assign activity to the process map. The Assign activity must be dropped in between the Receive and Invoke activities.

Adding the Assign Activity



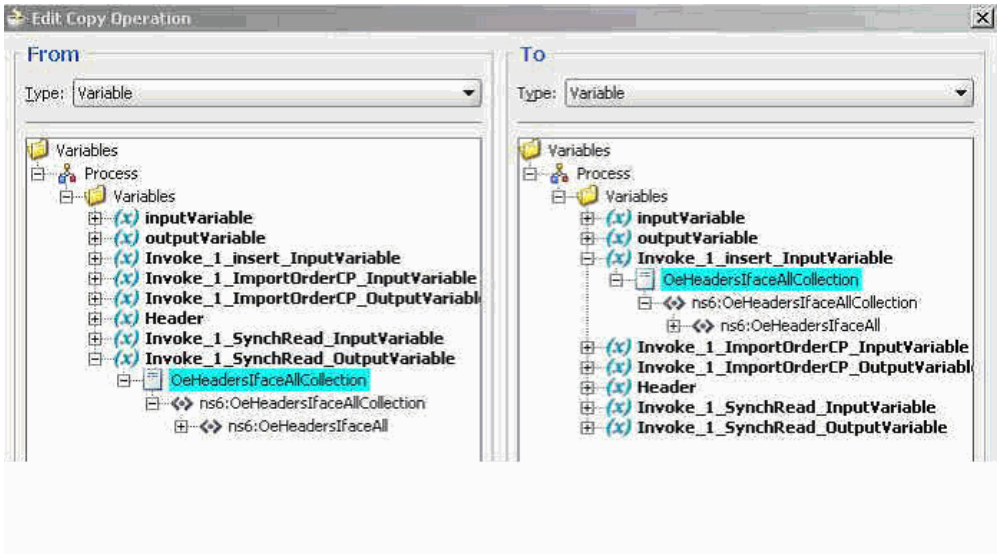
2. You now need to configure the Assign activity. Double-click the **Assign** activity in the process map.

3. The Assign dialog box appears. The Copy Rules tab appears by default. Click **Create**.

4. The Create Copy Rule dialog box appears. In the **To** group, expand the **Variables** node by clicking the plus sign next to it, then select **Expression** from the **From** group to assign values to the input variables.

Selecting the **From** variable and **To** variable will populate the **XPath** input boxes at the bottom of the window.

Assigning Values to Input Variables



5. After assigning values to the input variables, click **OK**.

6. Select **Make** from the Run menu or press Ctrl+F9 to compile the BPEL process to check for errors. The compilation result appears.

Run-Time Tasks for Interface Tables

After designing the BPEL process, the next step is to deploy, run, and monitor it.

1. Deploy the BPEL process., page 5-23

2. Test the BPEL process., page 5-25

Deploying the BPEL Process

You need to deploy the BPEL process to a BPEL server before you can run it. The BPEL process is first compiled, and then deployed to the BPEL server.

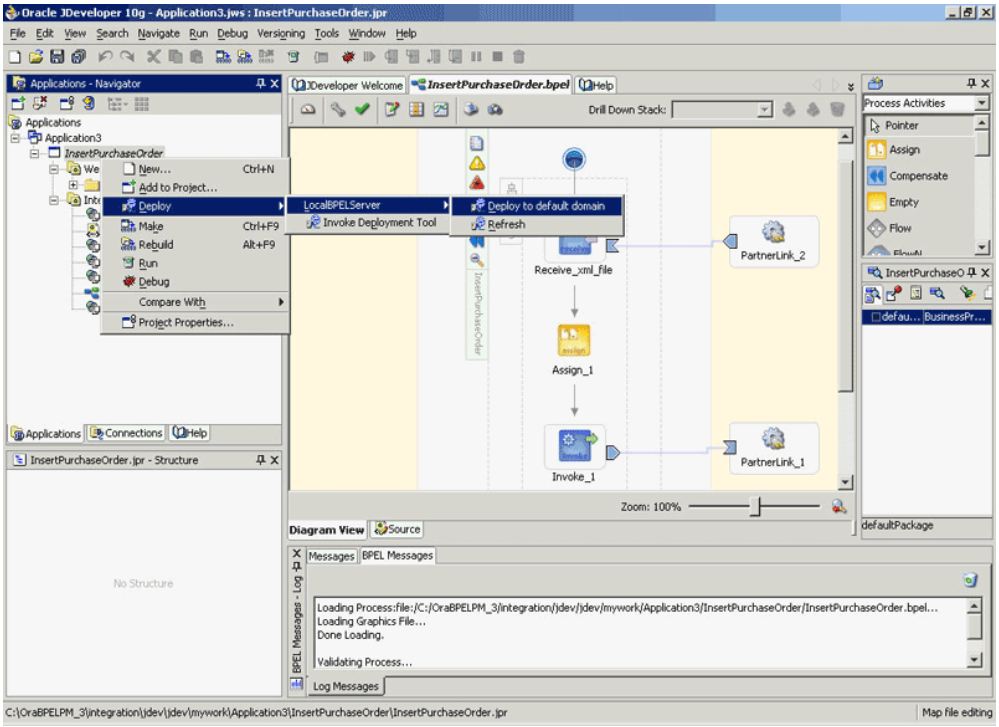
To deploy the BPEL process:

1. Select the BPEL project in the Applications window.

2. Right-click the project name. Select **Deploy** from the menu that appears.

3. Select **Local BPEL Server** followed by **Deploy to Default Domain**, if you are deploying the process on the local BPEL server.

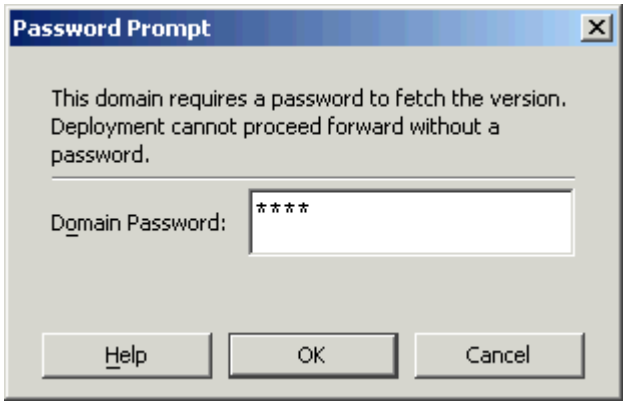
Deploying the BPEL Process



Note: You can select **Invoke Deployment Tool** if you want to deploy to a different BPEL server.

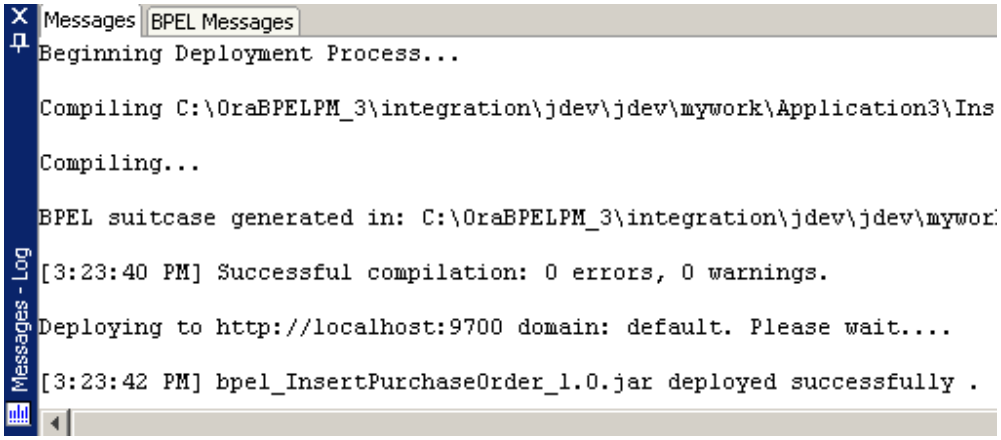
4. The Password Prompt dialog box appears. Enter the password for the default domain in the **Domain Password field**. Click **OK.**

Specifying the Domain Password



5. The BPEL process is compiled and deployed. You can check the progress in the Messages window.

Messages Window



Testing the BPEL Process

Once the BPEL process is deployed, it can be seen in the BPEL console. You can manage and monitor the process from the BPEL console. You can also test the process and the integration interface by manually initiating the process.

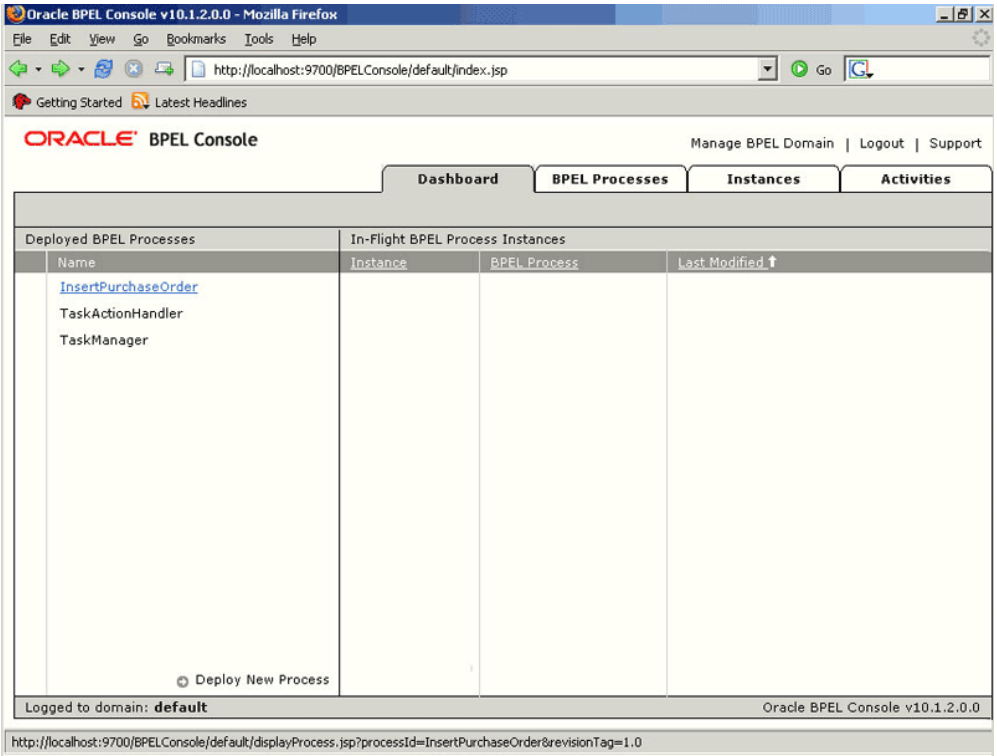
To manually initiate and monitor the BPEL process:

1. To open the BPEL Console, click **Start**, and then select **Programs**. In the Programs menu, select **Oracle - ORACLE_HOME**, **Oracle BPEL Process Manager 10.1.3**, and then select **BPEL Console**.

2. The BPEL console login dialog box appears. Select **Default** in the **Domain** box. Enter the password for the default domain in the **Password** field, and then click **Login.**

3. The Oracle BPEL console page appears. The list of deployed processes is shown under Deployed BPEL Processes.

Deployed BPEL Processes

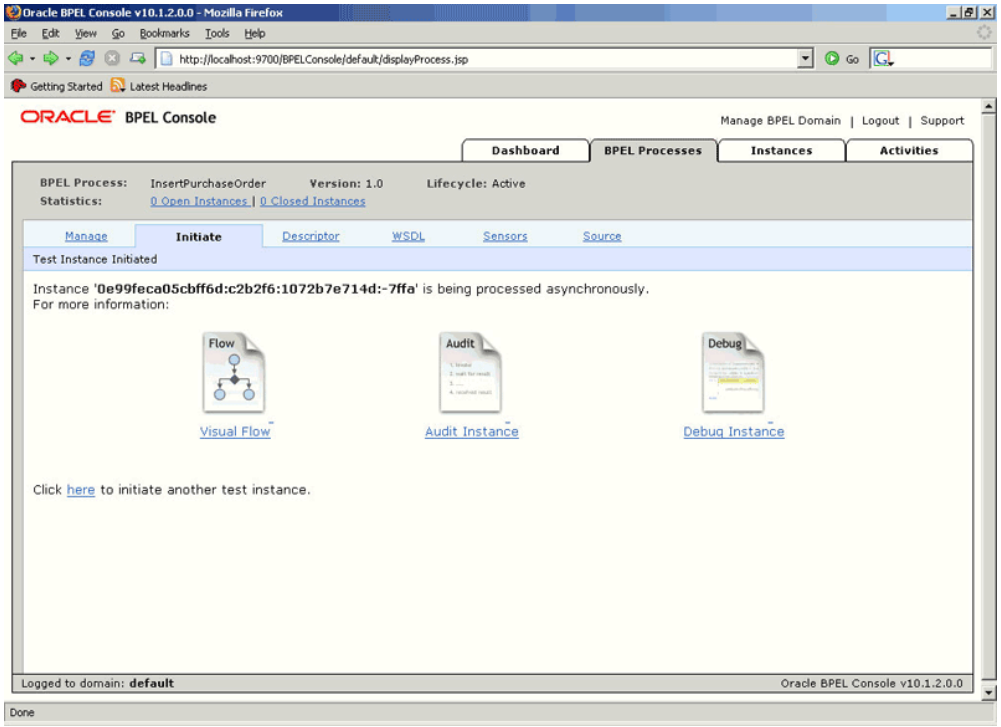


4. Click the BPEL process that you want to initiate. The Initiate page appears. Enter the input values required by the process.

5. Click **Post XML Message** to initiate the process.

6. The BPEL process is now initiated. You can check the process flow by clicking the **Visual Flow** icon.

BPEL Console Initiate Page



7. The audit trail provides information about the steps that have been executed. You can check the audit trail by clicking the **Audit Instance** icon.

> **Note:** To confirm that the records have been written into the open interface tables, you can write the SQL `SELECT` statements and fetch the results showing the latest records inserted into the open interface tables. Alternatively, open the forms of the module for which the record has been inserted, and then check for changes in the form.

Design-Time Tasks for Views

This section describes the steps to configure OracleAS Adapter for Oracle Applications using the Adapter Configuration Wizard in Oracle JDeveloper.

Prerequisites to Configure Views

You need to define a uniqueness criteria, which could be a single key or composite keys.

Following is a list of the procedures required to accomplish the design-time tasks.

1. Create a new BPEL project., page 5-28

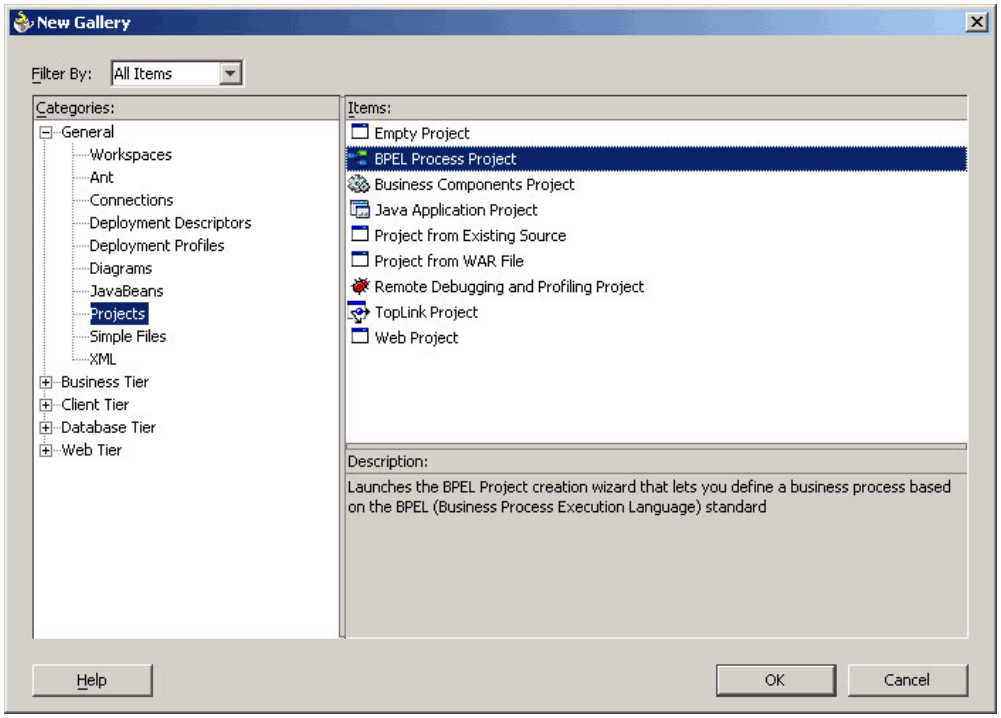
2. Add a partner link., page 5-31
3. Configure the Invoke activity., page 5-45
4. Configure the Assign activity., page 5-48

Creating a New BPEL Project

To create a new BPEL project:

1. Open JDeveloper BPEL Designer.
2. From the **File** menu, select **New**. The New Gallery dialog box appears.
3. Select **All Items** from the **Filter By** list. This displays a list of available categories.
4. Expand the **General** node, and then select **Projects**.
5. Select **BPEL Process Project** from the **Items** list.

Creating a New BPEL Process Project

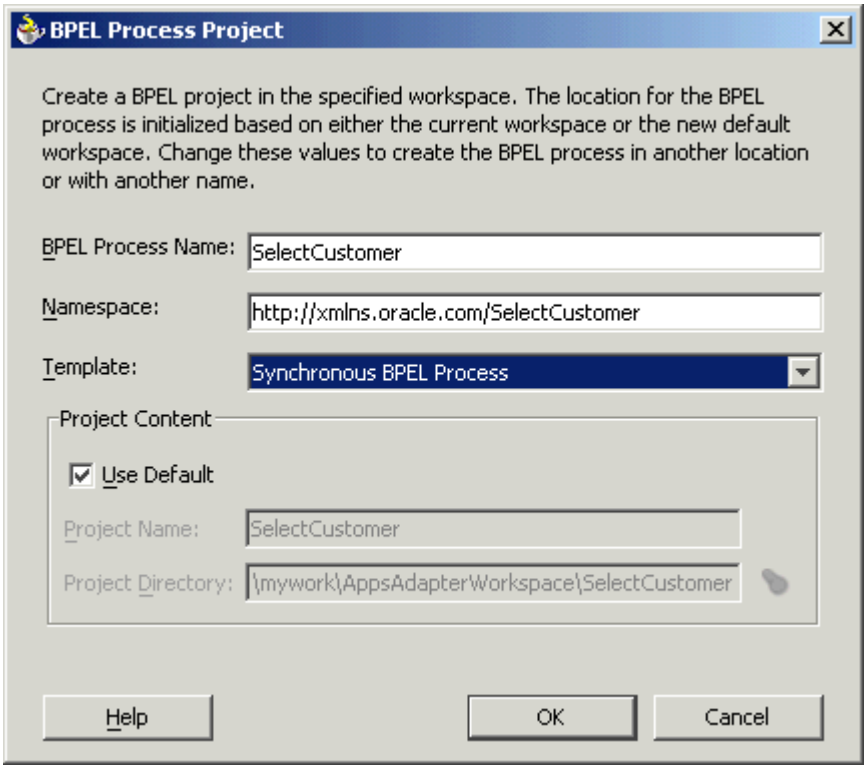


6. Click **OK**. The BPEL Process Project dialog box appears.

7. In the **BPEL Process Name** field, enter a descriptive name, for example, `SelectCustomer`.

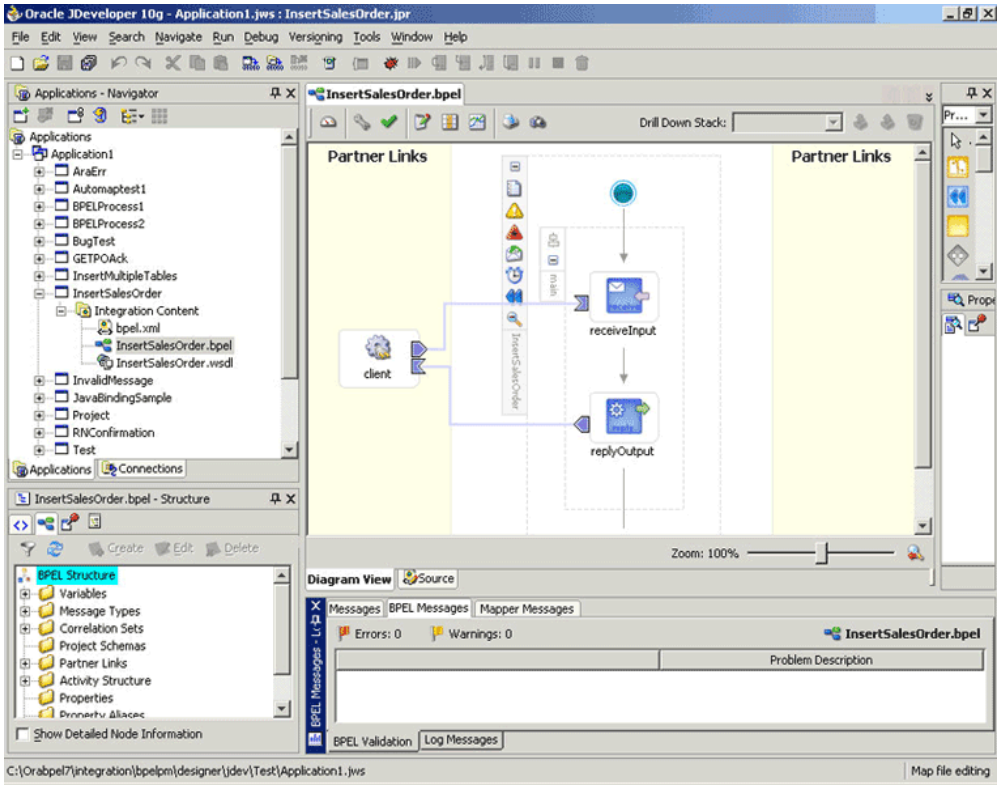
8. From the **Template** list, select **Asynchronous BPEL Process**. Keep the default selection for Use Default in the Project Content section.

Specifying a Name for the New BPEL Process Project



9. Click **OK**. A new asynchronous BPEL process with the required source files including `bpel.xml`, `SelectCustomer.xml`, and `SelectCustomer.wsdl` is created.

New BPEL Process



Adding a Partner Link

This section describes how to create an OracleAS adapter for the application service by adding a partner link to the BPEL process. A BPEL partner link defines the link name, type, and the role of the BPEL process that interacts with the partner service.

To add a partner link:

1. Drag and drop **PartnerLink** into the border area of the process diagram. The Create Partner Link dialog box appears.

2. Click **Define Adapter Service** in the WSDL Settings section. The Adapter Configuration Wizard appears.

3. Click **Next**. The Adapter Type dialog box appears.

4. Select **Oracle Applications** to specify the adapter you want to configure.

Selecting OracleAS Adapter for Oracle Applications

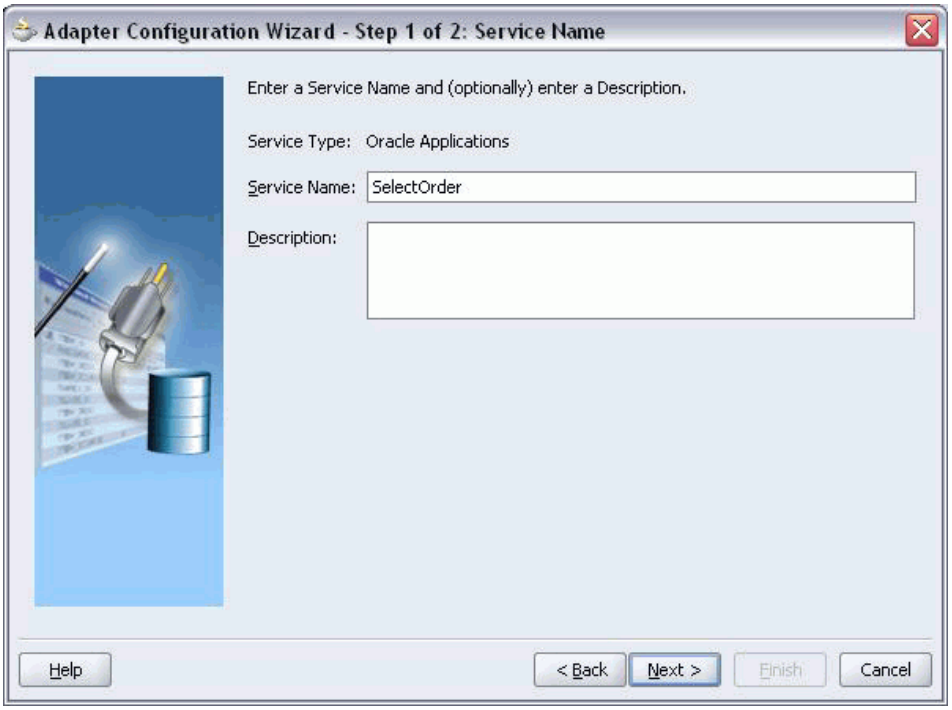
5. Click **Next**. The Service Name dialog box appears.

6. Enter the following information:

   1. In the **Service Name** field, enter a service name.

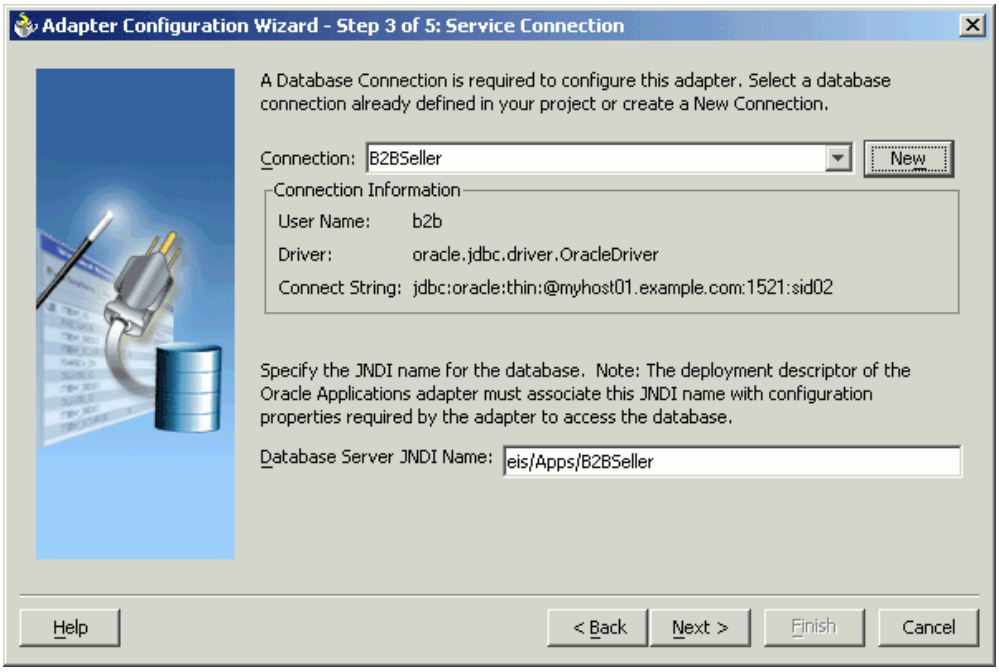
   2. In the **Description** field, enter a description for the service. This is an optional field.

Specifying the Service Name



7. Click **Next**. The Service Connection dialog box appears.

Specifying a Database Connection



8. Select an existing database connection from the Connection list. Alternatively, you can click **New** to define a database connection. The Create Database Connection Wizard appears.

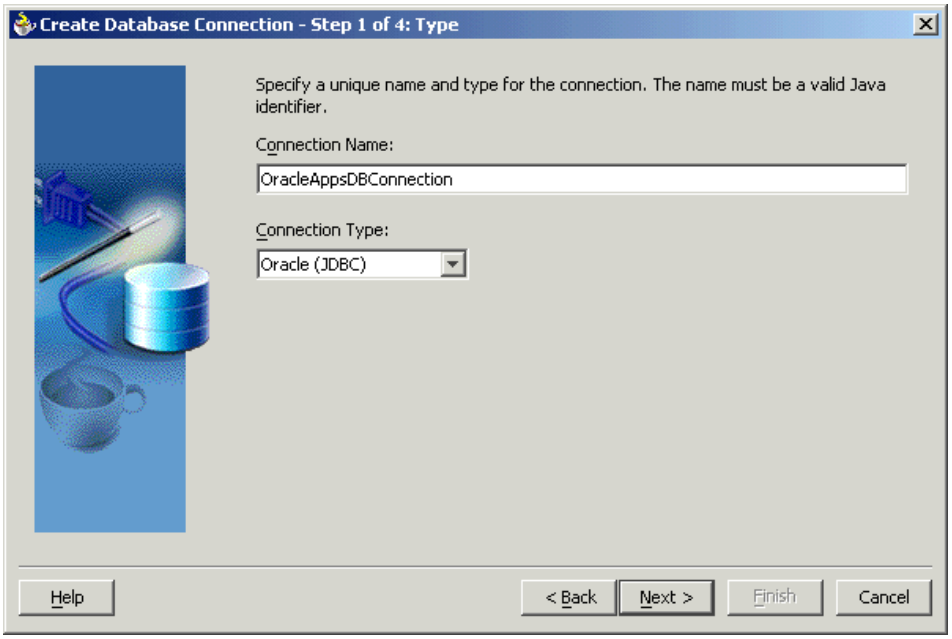
   Note: You need to connect to the database where Oracle Applications is running.

9. Enter the following information in the Type dialog box:

   1. In the **Connection Name** field, specify a unique name for the database connection.

   2. From the **Connection Type** list, select the type of connection for your database connection.

Specifying the Name and Type of Connection



10. Click **Next**. The Authentication dialog box appears.

11. Enter information in the following fields:

    1. In the **UserName** field, specify a unique name for the database connection.

    2. In the **Password** field, specify a password for the database connection.

12. Click **Next**. The Connection dialog box appears.

13. Enter information in the following fields:

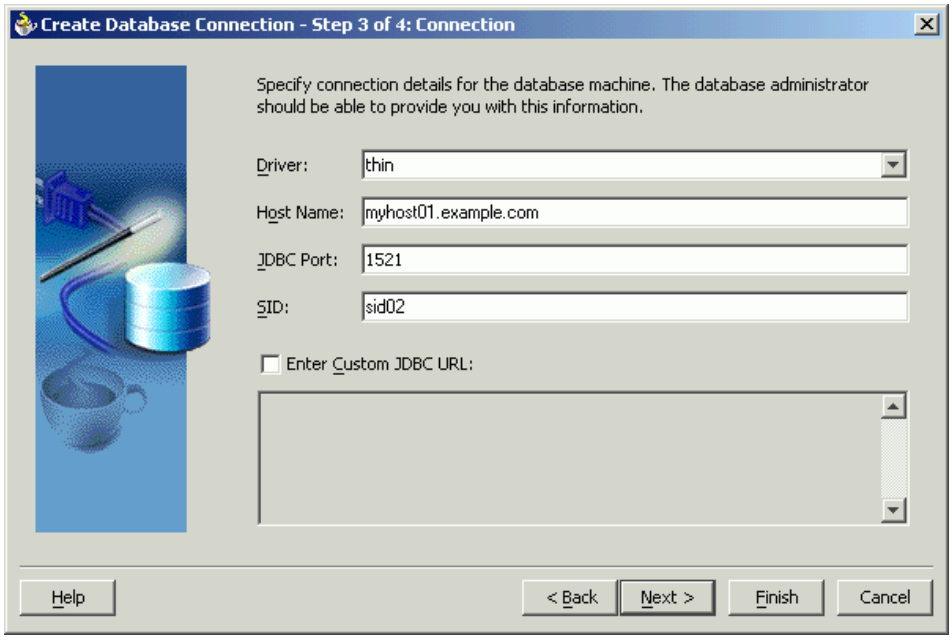
    1. From the **Driver** list, select **Thin**.

    2. In the **Host Name** field, specify the host name for the database connection.

    3. In the **JDBC Port** field, specify the port number for the database connection.

    4. In the **SID** field, specify a unique SID value for the database connection.

Specifying the New Database Connection Information



14. Click **Next**. The Test dialog box appears.

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

16. Click **Next**. The Service Connection dialog box appears, providing a summary of the database connection.

17. The JNDI (Java Naming and Directory Interface) name corresponding to the database connection appears automatically in the **Database Server JNDI Name** field. Alternatively, you can specify a JNDI name.

    Note: When you specify a JNDI name, the deployment descriptor of the Oracle Applications adapter must associate this JNDI name with configuration properties required by the adapter to access the database.

    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 for production.

    Note: *Oracle Application Server Adapter Concepts* for understanding JNDI concepts.

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

    Once you have completed creating a new connection for the service, you can add a view by browsing through the maps available in Oracle Applications.

19. Click **Next** in the Service Connection dialog box.

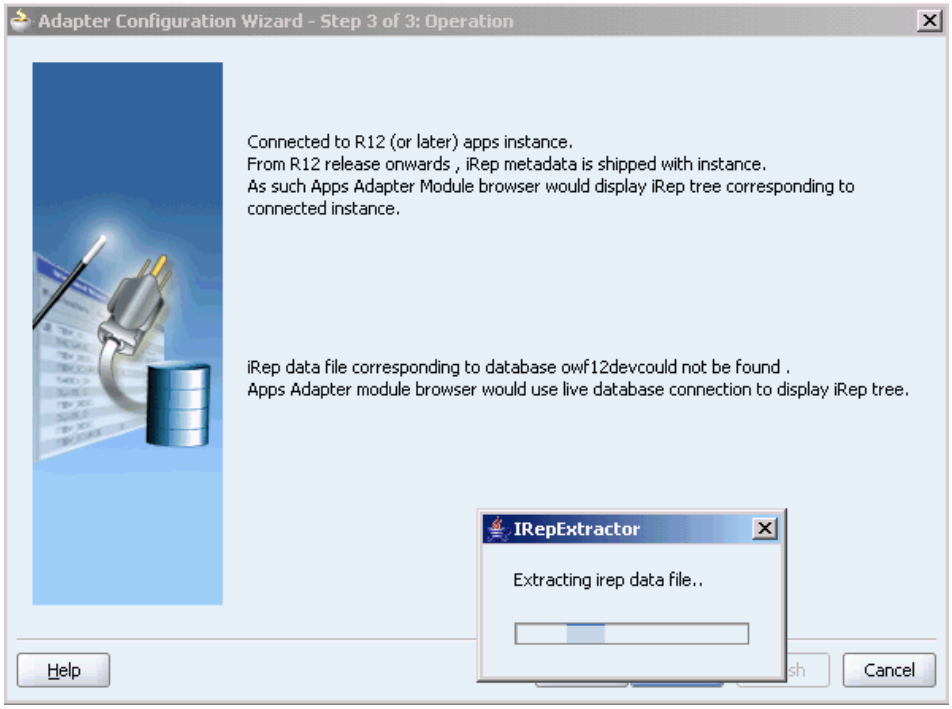
    For Oracle E-Business Suite Release 12:

    If you are connecting to Oracle E-Business Suite Release 12, then the **IREP File not present** dialog box appears indicating that OracleAS Adapter could not find Oracle the Integration Repository data file corresponding to the database you are connecting to Oracle Applications in your workspace. Absence of the data file would make browsing or searching of Integration Repository tree considerably slow. You can choose to extract the data file and create a local copy of the Integration Repository data file. Once it is created successfully, OracleAS Adapter will pick it up automatically next time and retrieve data from your local Integration Repository.

    You can select one of the following options:

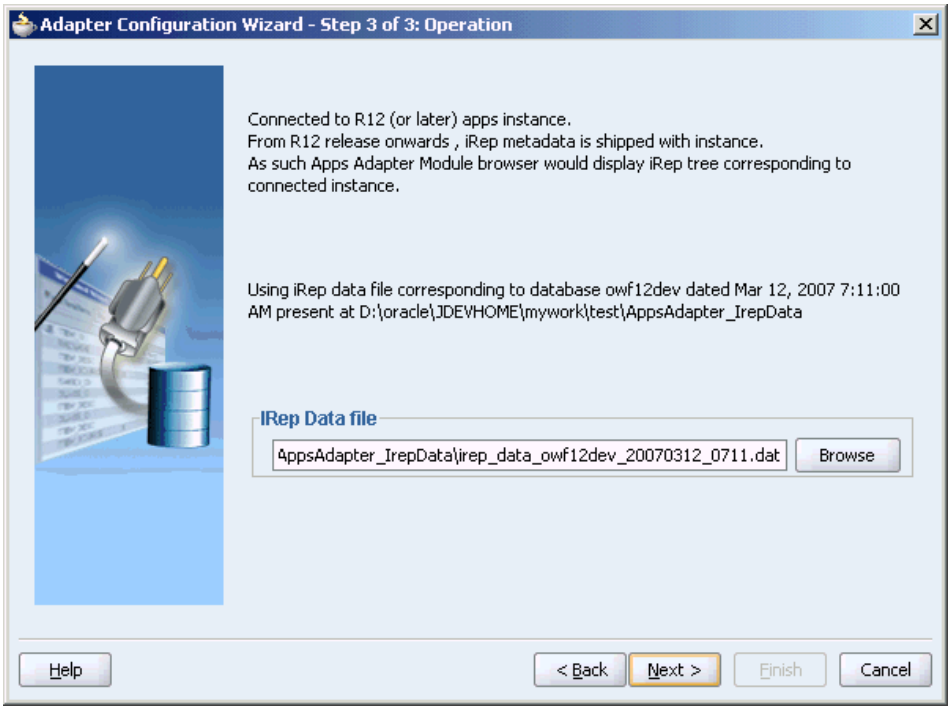
    - Click **Yes** to extract the Integration Repository data file.

Extracting Integration Repository Data File



After the system successfully creates a local copy of the Integration Repository data file, next time when you connect to the database, you will find the **IRep Data File** field appears in the Operation dialog box indicating where your local copy exists with the creation date and time as part of the file name.

Using the Local Integration Repository Data File



- Click **No** to query the Integration Repository data file from the live database you are connecting to display the Integration Repository tree.

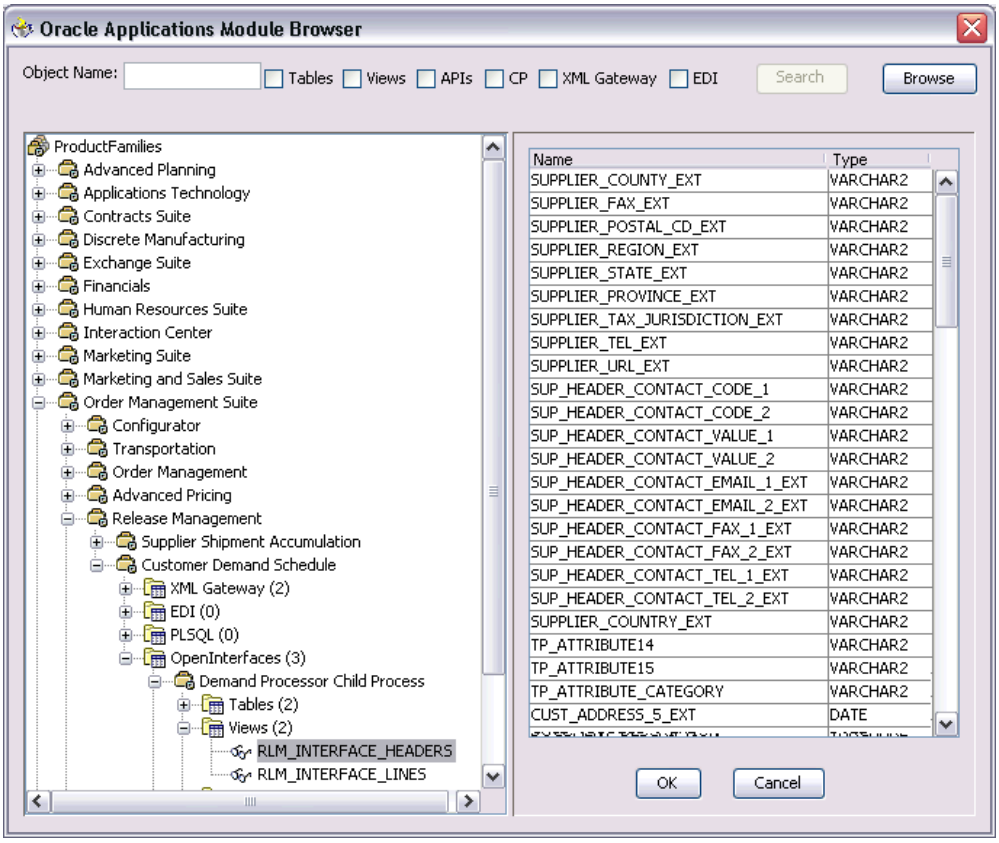
  Note: It is highly recommended that you create a local copy of the Integration Repository data file so that OracleAS Adapter will query the data next time from the local copy in your workspace to enhance the performance.

For Oracle E-Business Suite pre-Release 11.5.10:

If you are connecting to a pre-11.5.10 Oracle Applications instance, you must select the interface type in the Adapter Configuration Wizard. Select **Tables/Views/APIs/Programs** to proceed.

20. Click **Get Object** to open the Oracle Applications Module Browser.

Selecting a View from the Oracle Applications Module Browser



21. Select a view, and then click **OK.**

    Note: You can also search for a view by entering the name of the program in the **Object Name** field. Select the **Views** check box, and then click **Search**.

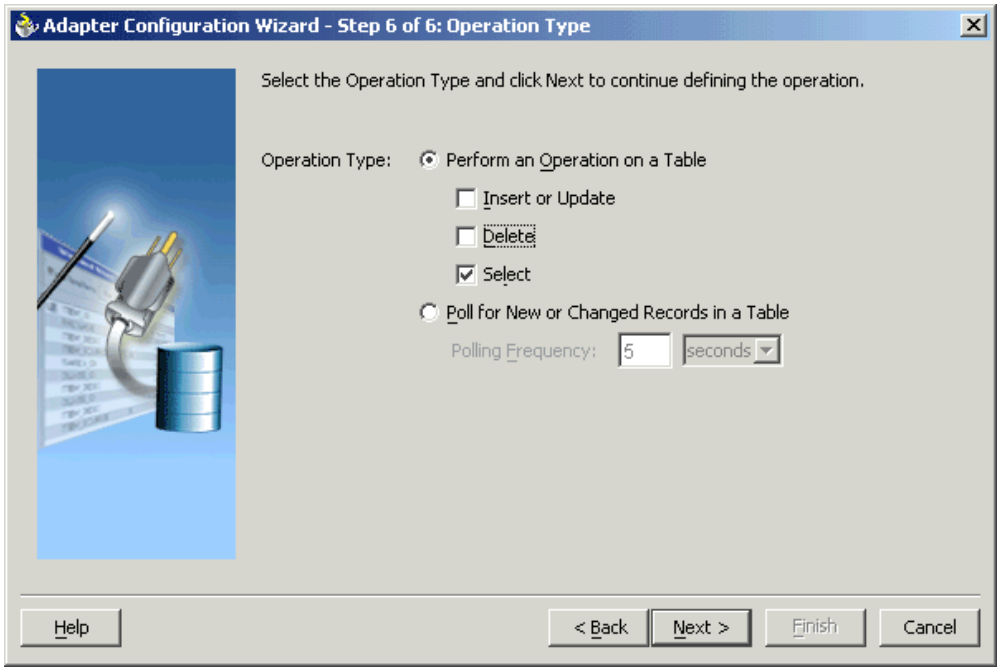
    The view is added to the Operation Objects section.

22. Repeat the preceding two steps to add more views.

23. Click **Next** in the Operation dialog box. The Operation Type dialog box appears.

24. Select **Perform an Operation on a Table**, then select **Select.**

Selecting the Type of Operation



Note: You can perform only the *Select* operation on views.

25. Click **Next**. The Select Table dialog box appears.

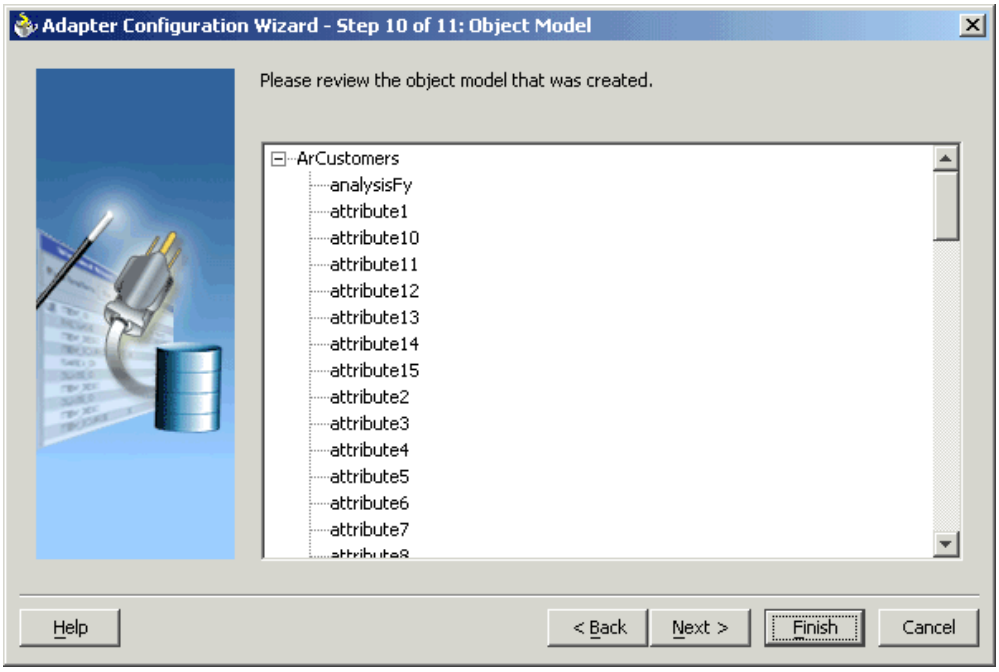
26. Select the required database table, and then click **Next**. The Define Primary Keys dialog box appears.

27. Select the primary key fields. You can also select multiple fields to define primary keys.

28. Click **Next**. The Relationships dialog box appears.

29. Click **Next**. The Object Model dialog box appears, which shows the object models that are created from the imported table definitions, including any relationships that you may have defined.

Available Object Models



30. Click **Next**. The Define WHERE Clause dialog box appears. If your service contains a `SELECT` query, then you can customize the `WHERE` clause of the `SELECT` statement.

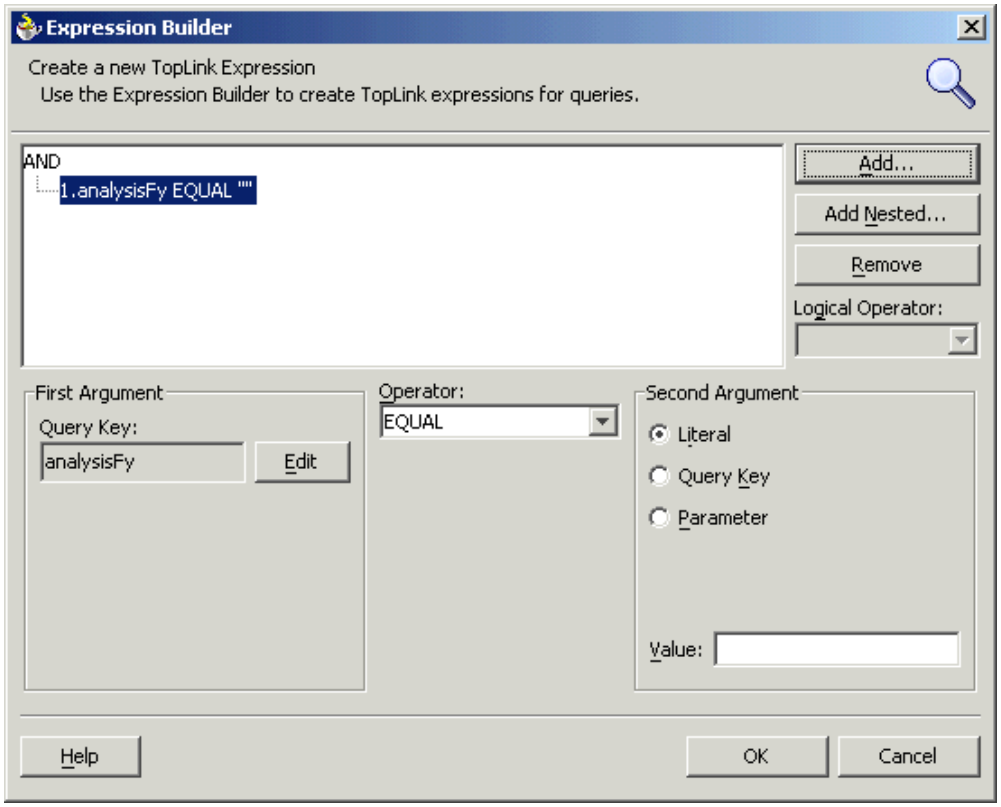
31. Click **Add** to add a new parameter. The Parameter Name dialog box appears.

32. Enter a name for the new parameter, and then click **OK**.

33. Click **Edit** to create an expression. The **Expression Builder** appears.

34. Click **Add** to create a TopLink expression.

Creating an Expression

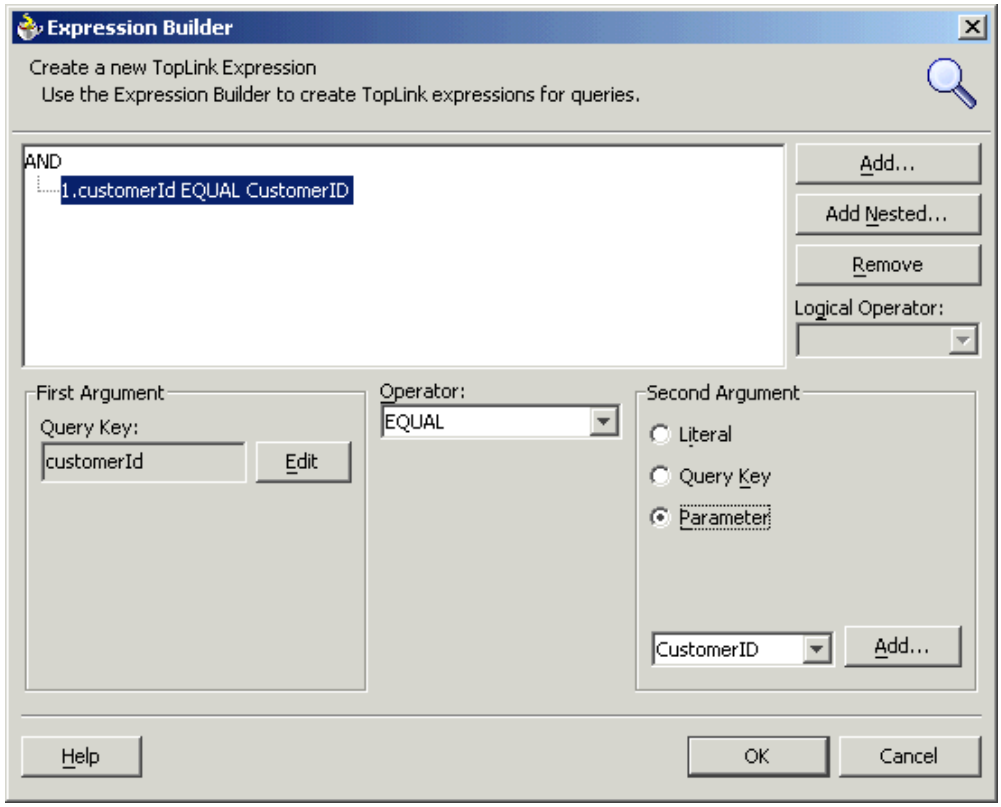


35. Click **Edit**. The Choose dialog box appears.

36. Select the required attribute, and then click **OK**.

37. Select **Parameter** in the Second Argument section.

Selecting the Required Attributes



Note: For more information about configuring expressions using the Expression Builder, refer to the Oracle TopLink page on OTN, [http://www.oracle.com/technology/products/ias/toplink/index.html] and the TopLink documentation in the Oracle Application Server documentation library. [http://www.oracle.com/technology/documentation/appserver.html]

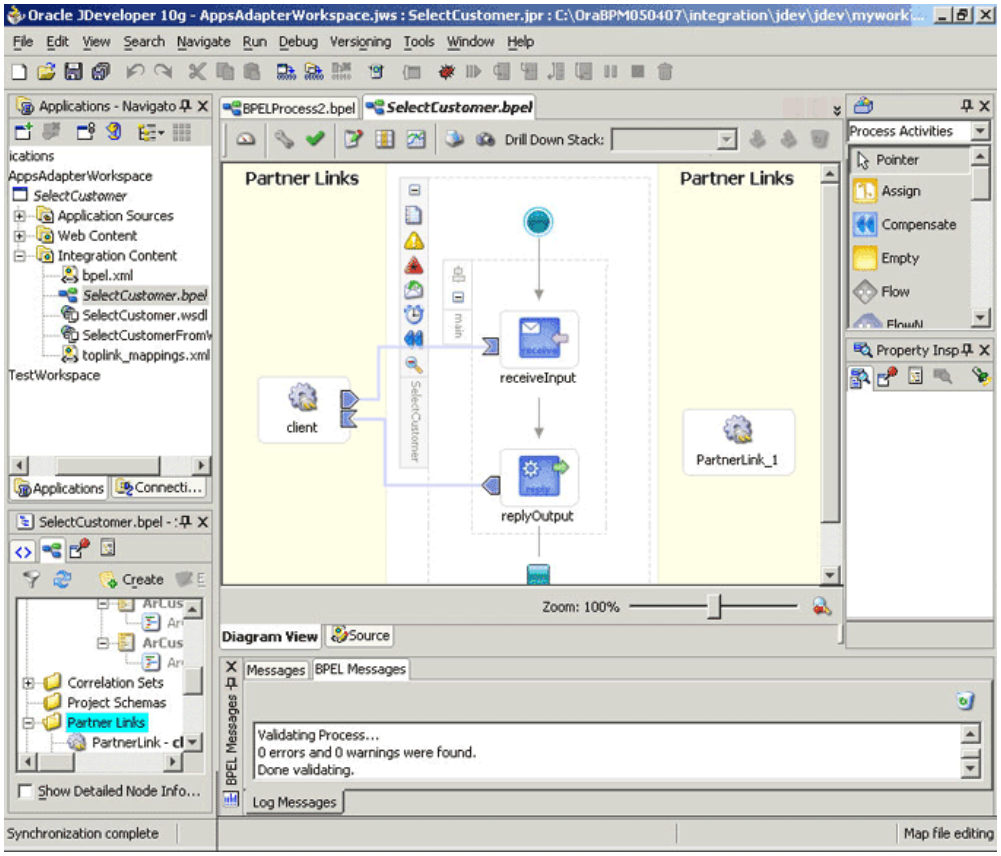
38. Click **OK**. The Define WHERE Clause dialog box appears with the selected values.

39. Click **Next**. The Finish dialog box appears.

40. Click **Finish**. The Create Partner Link dialog box appears.

41. Click **OK**. The new BPEL process is created.

New BPEL Process



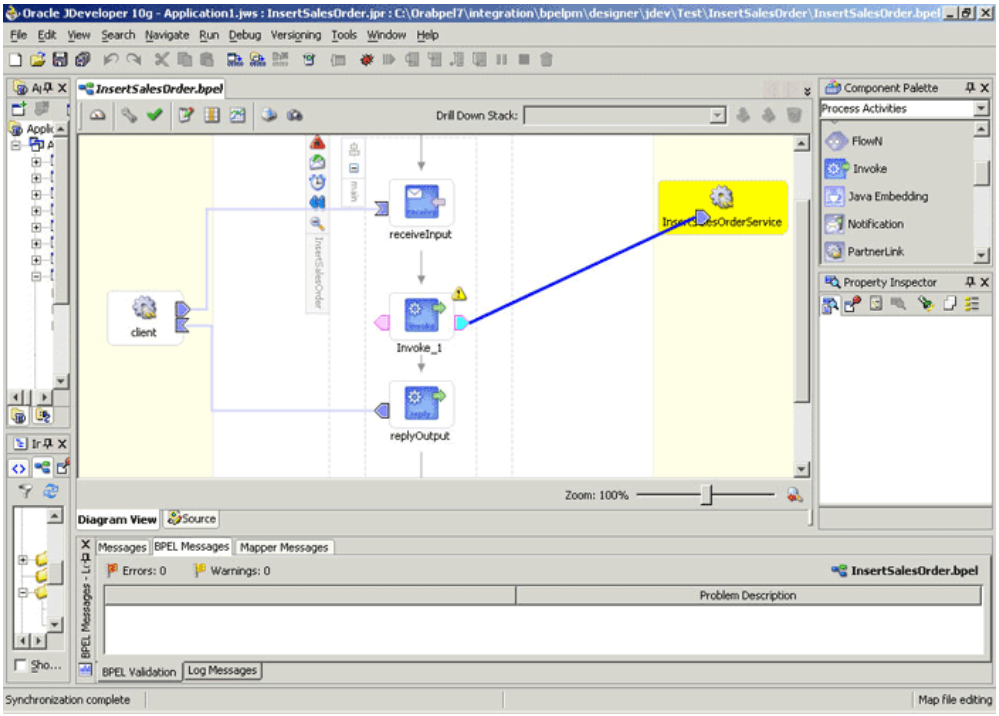
After adding and configuring the partner link, the next task is to configure the BPEL process.

Configuring the Invoke Activity

To configure the Invoke activity:

1. Drag **Invoke** from the palette and drop it at the location where you want to insert the invoke activity in your BPEL process.

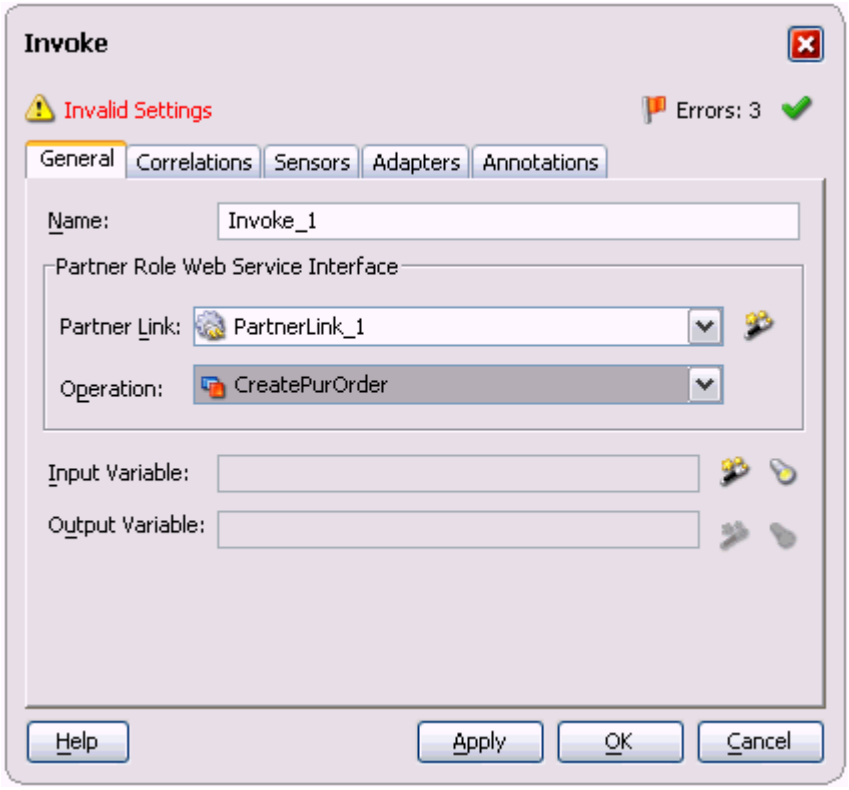
Dragging an Invoke Activity



The Edit Invoke dialog box appears.

2. Double-click **Invoke** in the process map to open the Invoke dialog box. The General tab is selected by default.

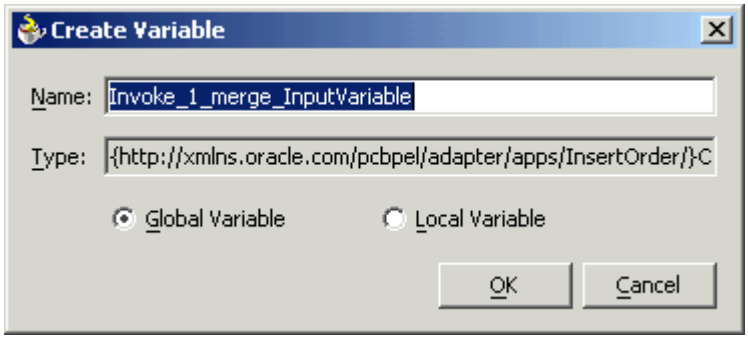
Configuring the Invoke Activity



3. In the **Partner Link** box, select the partner link to invoke. This is the partner link that you configured in the previous section.

4. Click the **Create** icon next to the Input Variable field. Enter a descriptive name for the variable in the Create Variable dialog box that appears. You can also accept the default name. Click **OK.**

Creating a Variable



5. In the Invoke dialog box, click **Apply**, and then click **OK**. The BPEL process is configured.

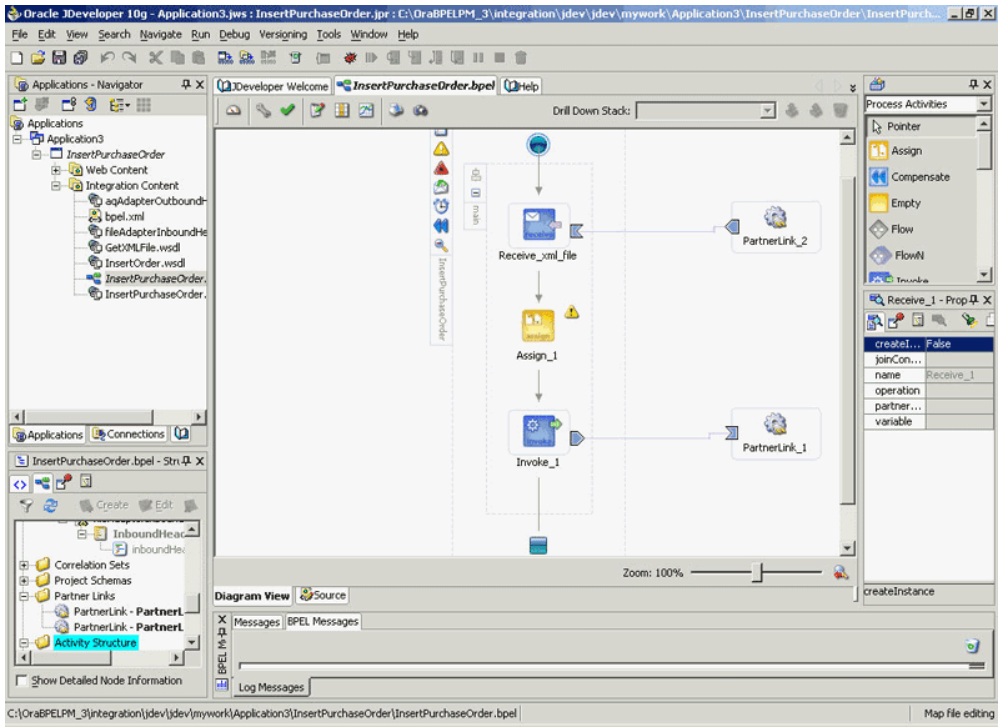
Configuring the Assign Activity

The next task is to add an Assign activity to the process map. This is used to provide values to the input variables.

To configure the Assign activity:

1. Drag and drop the Assign activity to the process map. The Assign activity must be dropped in between the Receive and Invoke activities.

Adding the Assign Activity

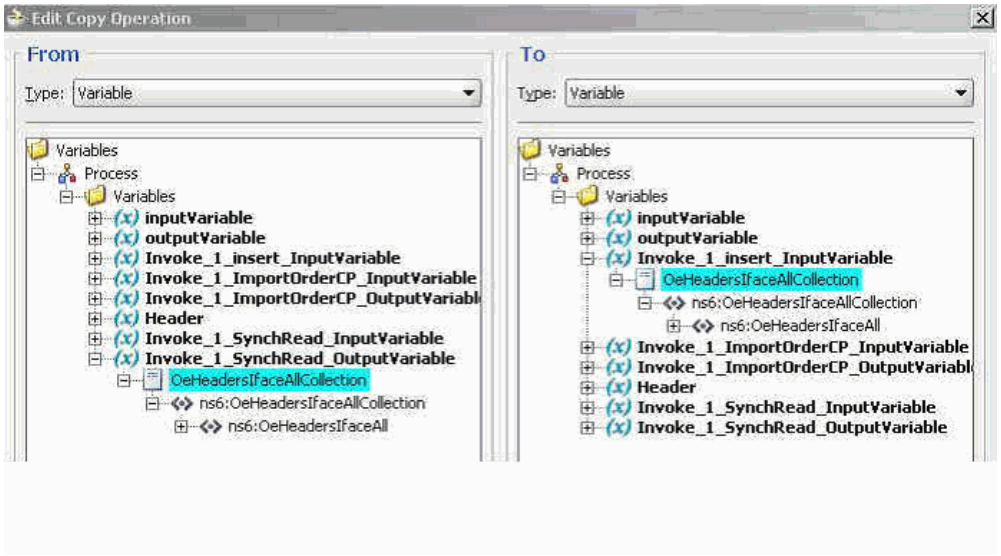


2. You must now configure the Assign activity. Double-click the **Assign** activity in the process map.

3. The Assign dialog box appears. The Copy Rules tab appears by default. Click **Create**.

4. The Create Copy Rule dialog box appears. In the To group, expand the Variables node by clicking the plus sign next to it, then select **Expression** from the From group to assign values to the input variables.

Assigning Values to the Input Variables



5. After assigning values to the input variables, click **OK**.

6. Select **Make** from the Run menu or press Ctrl+F9 to compile the BPEL process to check for errors. The compilation result appears.

Run-Time Tasks for Views

After designing the BPEL process, the next steps are to deploy, run and monitor it.

1. Deploy the BPEL process., page 5-49

2. Test the BPEL process., page 5-51

Deploying the BPEL Process

You need to deploy the BPEL process before you can run it. The BPEL process is first compiled, and then deployed to the BPEL server. The following steps discuss deploying the BPEL process to a BPEL server:

To deploy the BPEL process:

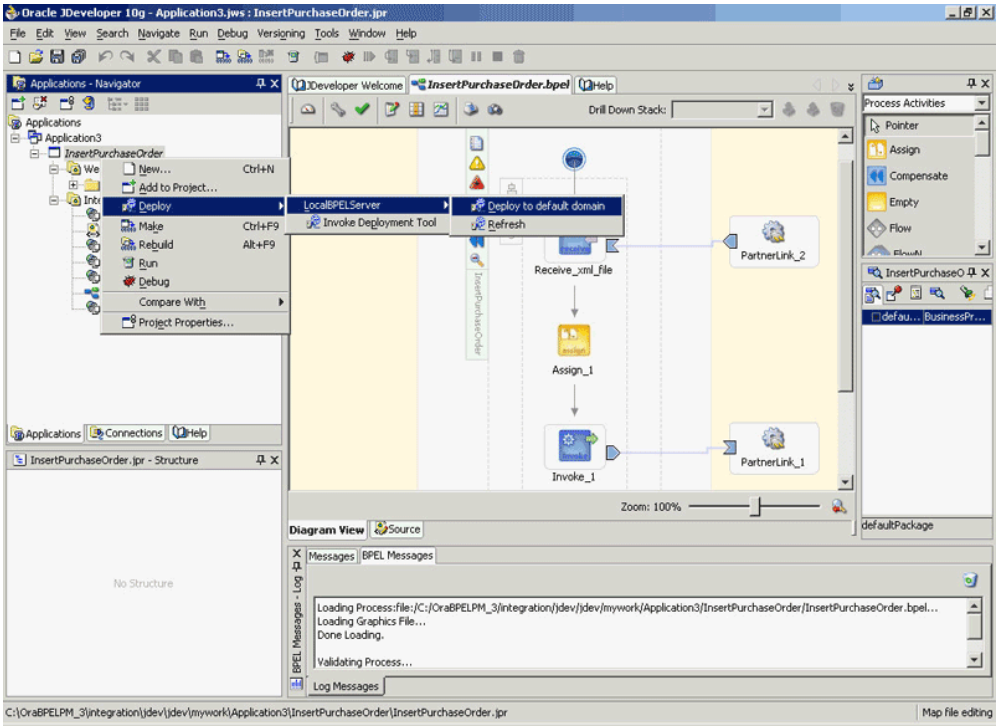
1. Select the BPEL project in the Applications window.

2. Right-click the project name. Select **Deploy** from the menu that appears.

3. Select **Local BPEL Server** followed by **Deploy to Default Domain**, if you are

deploying the BPEL process on the local BPEL server.

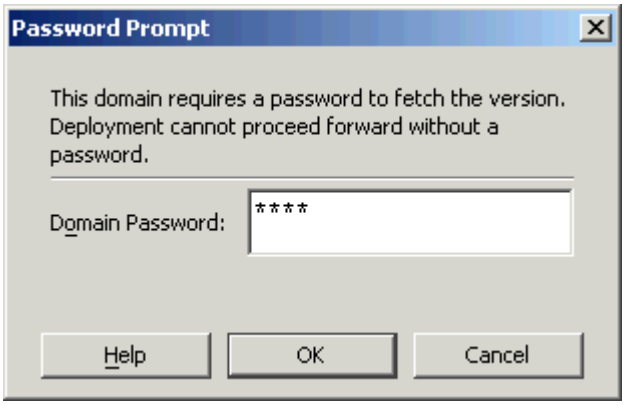
Deploying the BPEL Process



Note: You can select **Invoke Deployment Tool** if you want to deploy to a different BPEL server.

4. The Password Prompt dialog box appears. Enter the password for the default domain in the **Domain Password field**. Click **OK.**

Specifying the Domain Password



5. The BPEL process is compiled and deployed. You can check the progress of the compilation in the Messages window.

Messages Window

```
Messages | BPEL Messages
Beginning Deployment Process...

Compiling C:\OraBPELPM_3\integration\jdev\jdev\mywork\Application3\Ins

Compiling...

BPEL suitcase generated in: C:\OraBPELPM_3\integration\jdev\jdev\mywor

[3:23:40 PM] Successful compilation: 0 errors, 0 warnings.

Deploying to http://localhost:9700 domain: default. Please wait....

[3:23:42 PM] bpel_InsertPurchaseOrder_1.0.jar deployed successfully .
```

Testing the BPEL Process

Once the BPEL process is deployed, it can be seen in the BPEL console. You can manage and monitor the process from the BPEL console. You can also test the process and the integration interface by manually initiating the process. The following steps discuss manually initiating and monitoring the BPEL process:

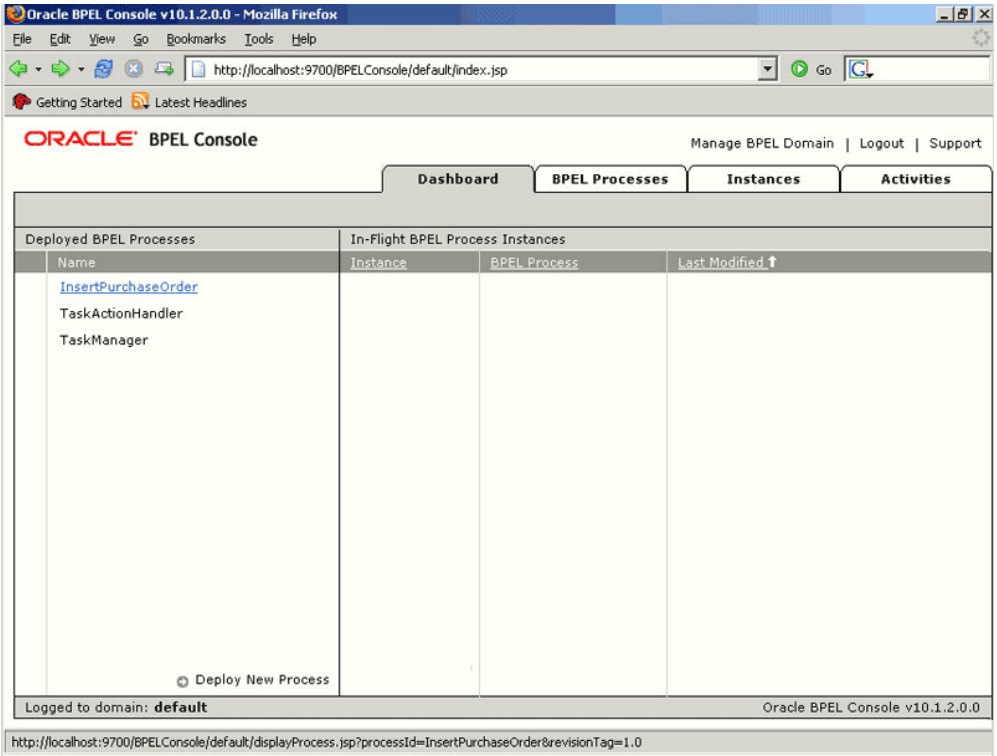
To manually initiate and monitor the BPEL process:

1. To open the BPEL Console, click **Start**, and then select **Programs**. In the Programs menu, select **Oracle - ORACLE_HOME**, **Oracle BPEL Process Manager 10.1.3**, and then select **BPEL Console**.

2. The BPEL Console login page appears. Select **Default** in the **Domain** box. Enter the password for the default domain in the **Password** field, and then click **Login**.

3. The Oracle BPEL Console appears. The list of deployed processes is shown under Deployed BPEL Processes on the BPEL Console page.

Deployed BPEL Processes

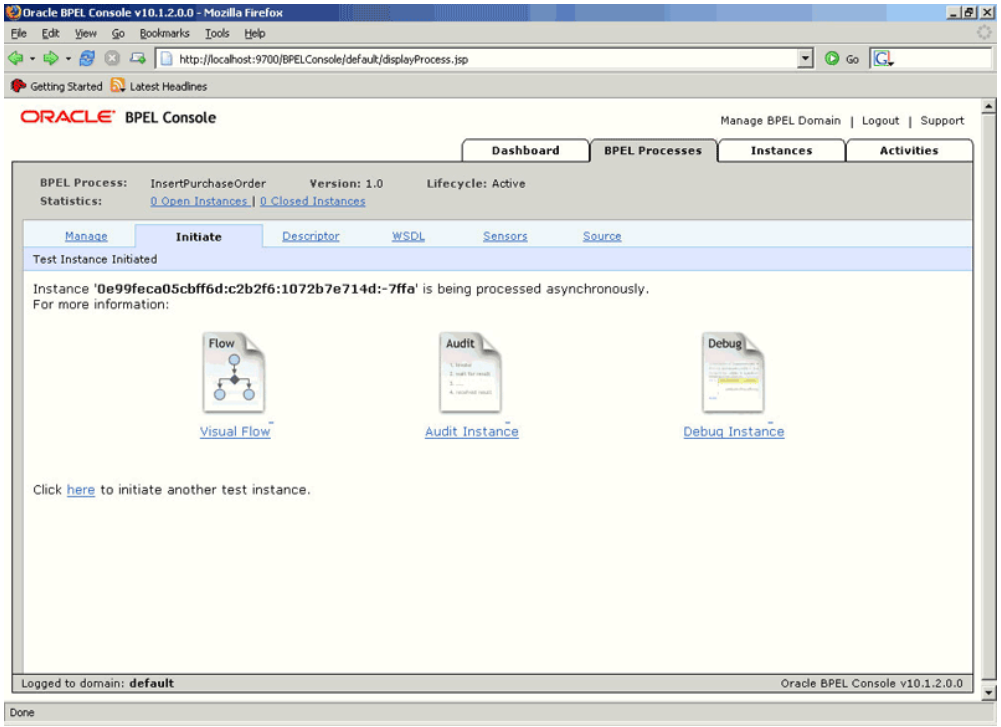


4. Click the BPEL process that you want to initiate. The Initiate page appears. Enter the input values required by the process.

5. Click **Post XML Message** to initiate the process.

6. The BPEL process is now initiated. You can check the process flow by clicking the **Visual Flow** icon on the BPEL Console Initiate page.

BPEL Console Initiate Page



7. The audit trail provides information about the steps that have been executed. You can check the audit trail by clicking the **Audit Instance** icon.

Troubleshooting and Debugging

If you experience problems with your Open Interface integration, you can take the following troubleshooting steps:

- If you're using complex (that is, Boolean) datatypes, ensure that PL/SQL wrappers are applied on the target instance of the concurrent program.
- Examine your transaction data in the Open Interface Tracking Form.
- Check the status of the concurrent programs by Request ID.

If you still experience problems with your integration, you can enable debugging.

Enabling Debugging

You can enable debugging for interface tables using the BPEL Process Manager.

To enable debugging:

1. Log into your BPEL Process Manager domain.
2. Select `yourdomain.collaxa.cube.ws`
3. Select **Debug.**

Debugging information is output to the log file for your domain. To examine the log file in the BPEL Process Manager, navigate to **Home > BPEL Domains >*yourdomain* > Logs.** The log file is ***yourdomain*.log**.

6

Using PL/SQL APIs

This chapter covers the following topics:

- Overview of PL/SQL APIs
- Design-Time Tasks for PL/SQL APIs
- Creating a New BPEL Project
- Adding a Partner Link
- Defining Wrapper APIs
- Declaring Parameters with a DEFAULT Clause
- Configuring the Invoke Activity
- Configuring the Transform Activity
- Run-Time Tasks for PL/SQL APIs
- Deploying the BPEL Process
- Testing the BPEL Process
- Troubleshooting and Debugging

Overview of PL/SQL APIs

OracleAS Adapter for Oracle Applications uses PL/SQL application programming interfaces (APIs) to insert and update data in Oracle Applications. APIs are stored procedures that enable you to insert and update data in Oracle Applications. Additionally, you can use PL/SQL APIs to retrieve data. For example, by using PL/SQL APIs, you can insert a customer record in Oracle Applications.

> **Note:** For more information about PL/SQL procedure limitations, refer to *Oracle Application Server Adapters for Files, FTP, Databases, and Enterprise Messaging User's Guide.*

Design-Time Tasks for PL/SQL APIs

This section describes how to configure the OracleAS Adapter for Oracle Applications to use PL/SQL APIs. It describes the tasks required to configure OracleAS Adapter for Oracle Applications using the Adapter Configuration Wizard in Oracle JDeveloper.

Prerequisites to Configure PL/SQL APIs

OracleAS adapter for Oracle Applications is deployed using the BPEL Process Manager (PM) in Oracle JDeveloper. The BPEL PM creates the WSDL interfaces for the API.

You need to populate certain variables in the BPEL PM in order to provide context information for Oracle Applications. The context information required for an API transaction includes the *username* and *responsibility* of an Oracle Applications user that has sufficient privileges (based on the applications context of Organization ID, Username and Responsibility) to run the program. The default value passed for the username is `SYSADMIN`. The default value passed for responsibility is `SYSTEM ADMINISTRATOR`.

You can change the default values specified in the generated WSDL for the username and responsibility. This is a static way of changing the context information. These values would apply to all invocations of the deployed business process. However, if you need to provide different context information for different invocations of the business process, then you can dynamically populate the header variable with values for username and responsibility. The context information can be specified by configuring an Assign activity before the Invoke activity in the BPEL PM.

Following is a list of the procedures required to accomplish the design-time tasks.

1. Create a new BPEL project., page 6-2
2. Add a partner link., page 6-5
3. Define wrapper APIs., page 6-17
4. Declare parameters with a DEFAULT clause., page 6-20

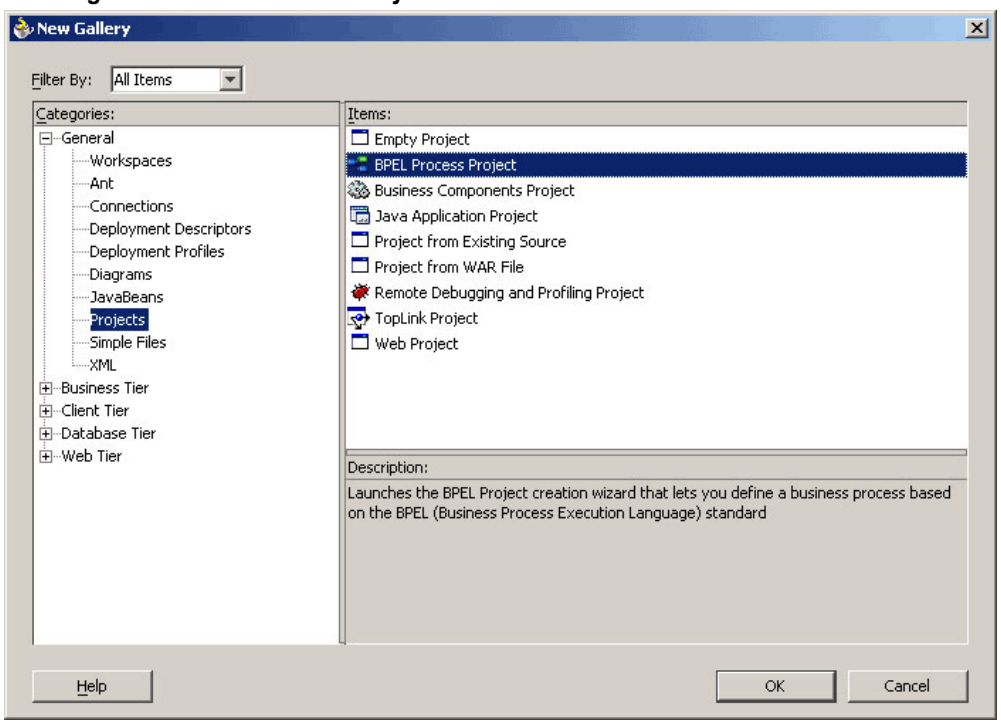
Creating a New BPEL Project

To create a new BPEL project:

1. Open BPEL Designer.
2. From the **File** menu, select **New**. The New Gallery dialog box appears.
3. Select **All Items** from the **Filter By** list. This displays a list of available categories.
4. Expand the **General** node, and then select **Projects**.

5. Select **BPEL Process Project** from the **Items** list.

Creating a New BPEL Process Project

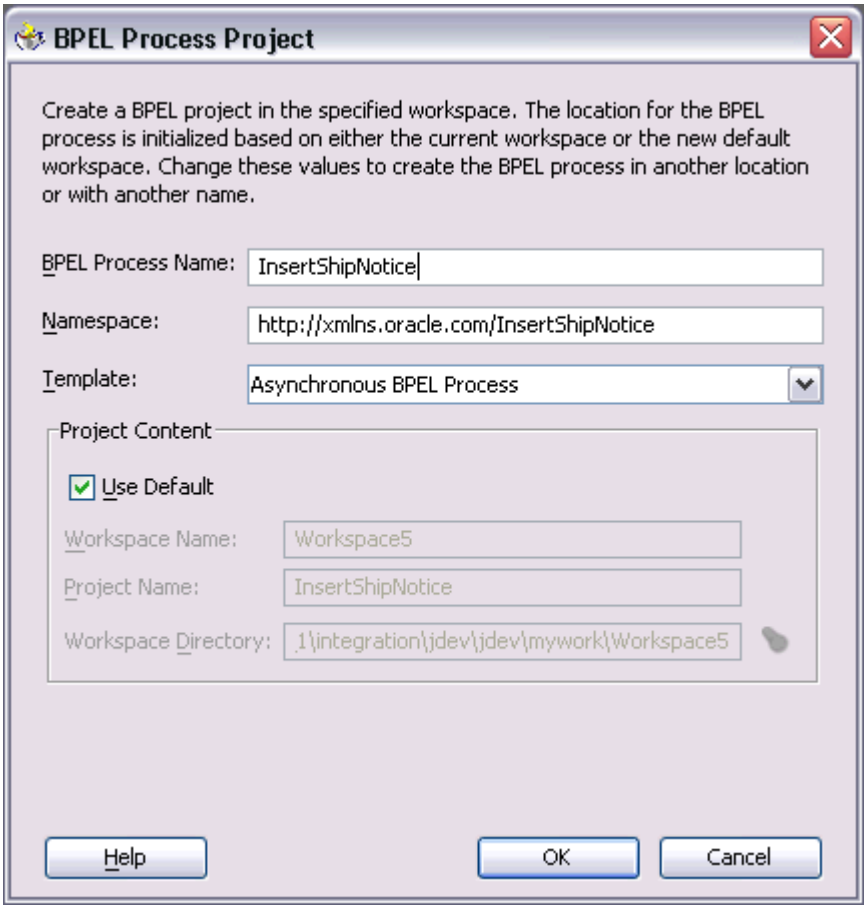


6. Click **OK**. The BPEL Process Project dialog box appears.

7. In the **BPEL Process Name** field, enter a descriptive name. For example, `InsertShipNotice`.

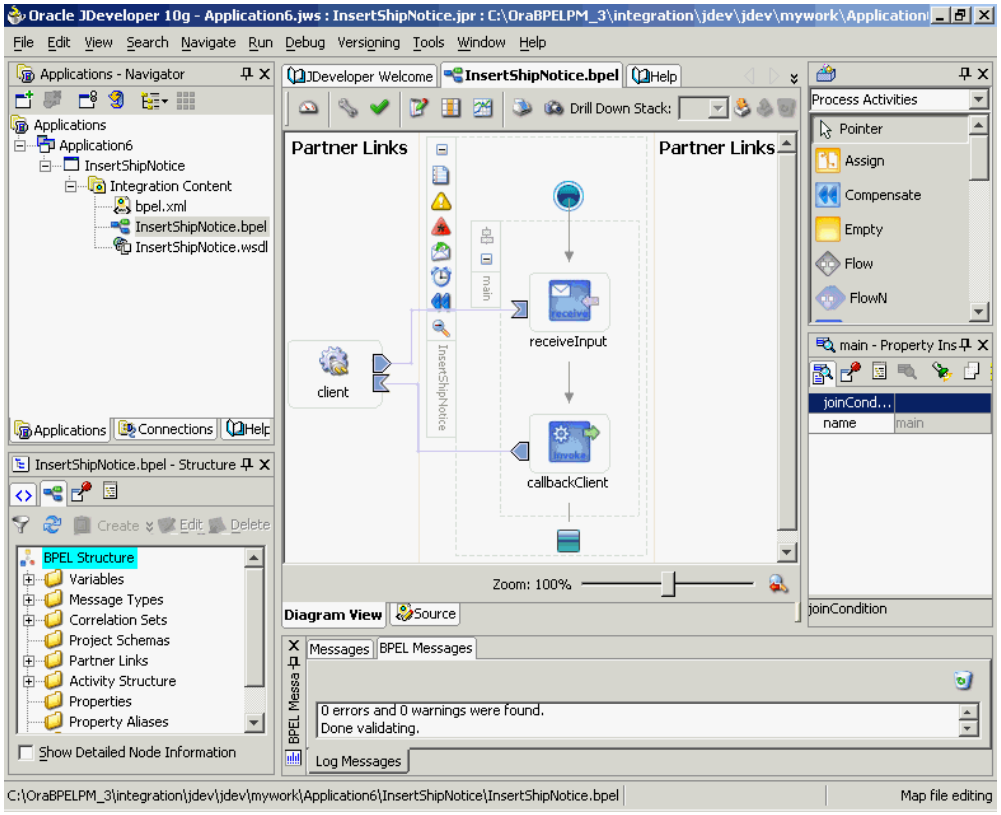
8. From the **Template** list, select **Empty BPEL Process**. Keep the default selection for **Use Default** in the Project Content section.

Specifying a Name for the New BPEL Process Project



9. Click **OK**. An empty BPEL process, with the necessary source files including `bpel.xml`, `InsertShipNotice.xml`, and `InsertShipNotice.wsdl` is created.

New BPEL Process



Adding a Partner Link

The next task is to add a partner link to the BPEL process. This section describes how to create an OracleAS adapter for the application service by adding a partner link to your BPEL process. A BPEL partner link defines the link name, type, and the role of the BPEL process that interacts with the partner service.

To add a partner link:

1. Drag and drop **PartnerLink** into the border area of the process diagram. The Create Partner Link dialog box appears.

2. Click **Define Adapter Service** in the WSDL Settings section. The Adapter Configuration Wizard appears.

3. Click **Next**. The Adapter Type dialog box appears.

4. Select **Oracle Applications** to specify the adapter you want to configure.

Selecting OracleAS Adapter for Oracle Applications

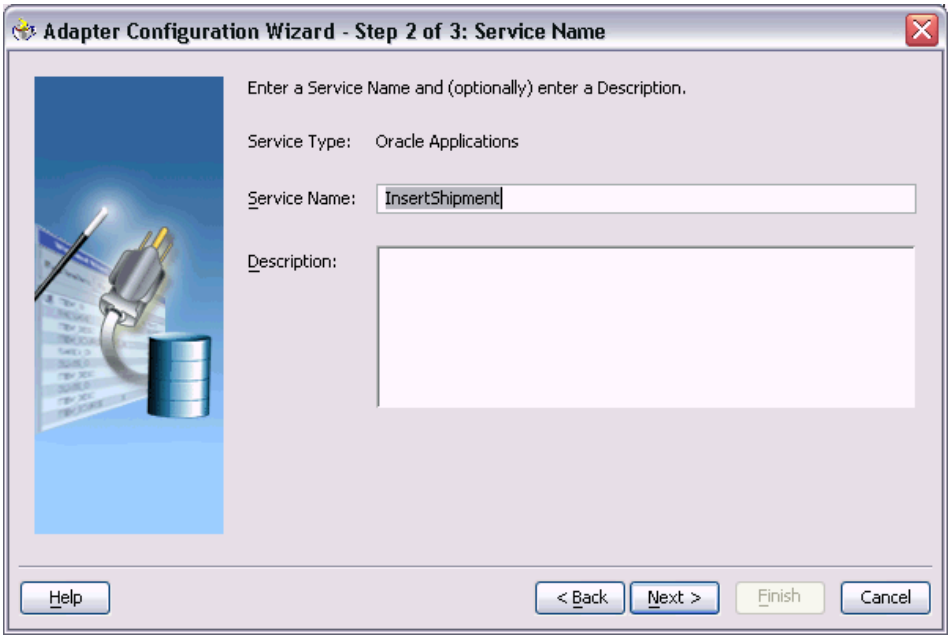
5. Click **Next**. The Service Name dialog box appears.

6. Enter the following information:

   1. In the **Service Name** field, enter a service name.

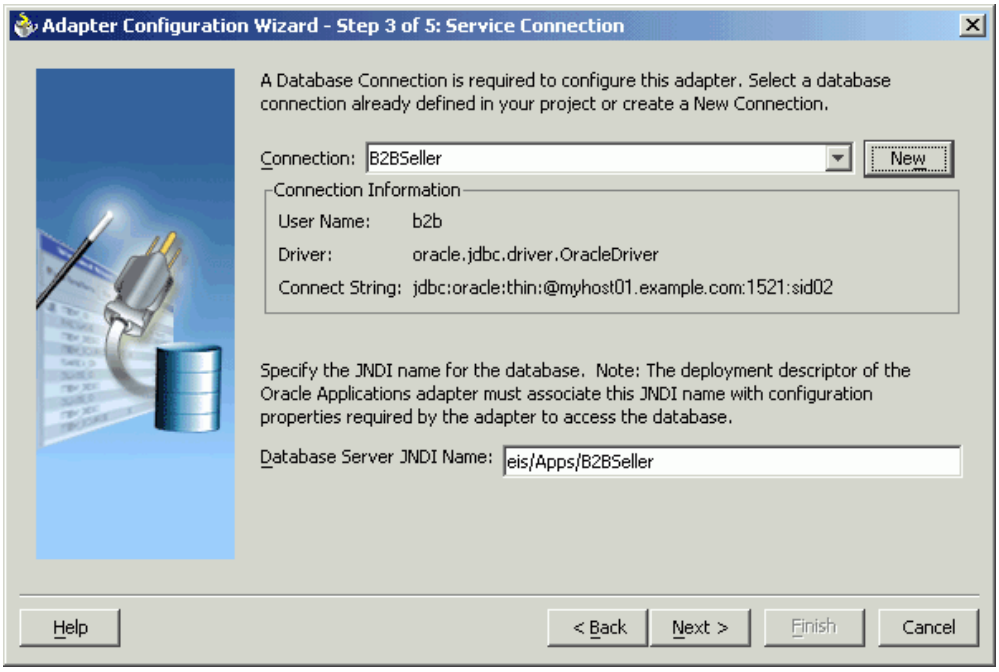
   2. In the **Description** field, enter a description for the service. This is an optional field. The Service Name dialog box appears.

Specifying the Service Name and Description



7. Click **Next**. The Service Connection dialog box appears.

Creating a New Database Connection



8. Click **New** to define a database connection. The Create Database Connection Wizard appears. Alternatively, you can select an existing database connection from the Connection list.

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

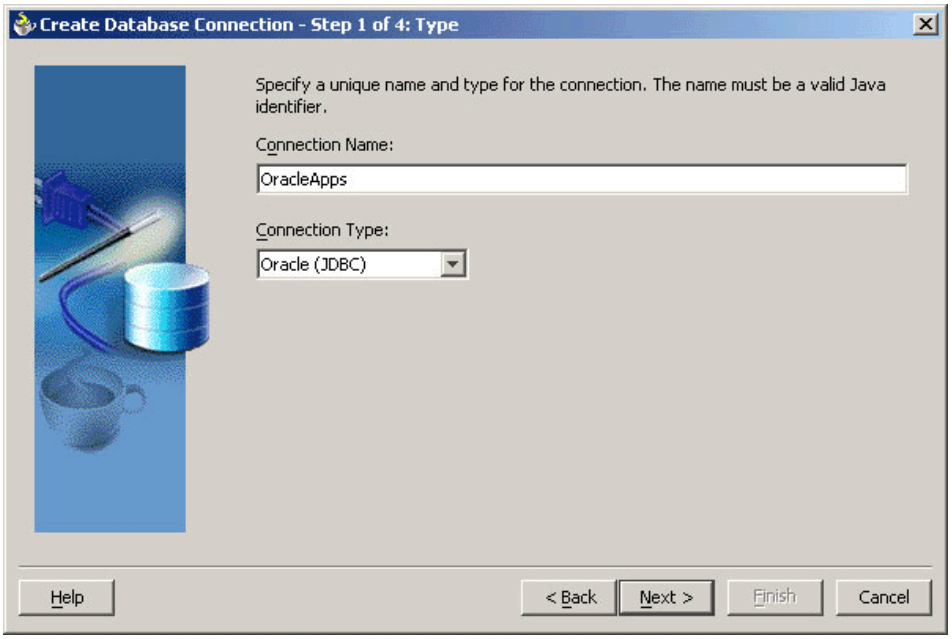
9. Click **Next.** The Type dialog box appears.

10. Enter the following information in the Type dialog box:

    1. In the **Connection Name** field, specify a unique name for the database connection.

    2. From the **Connection Type** list, select the type of connection for your database.

Specifying a Name and Type for the Database Connection



11. Click **Next**. The Authentication dialog box appears.

12. Enter the following information:

    1. In the **UserName** field, specify a unique name for the database connection.

    2. In the **Password** field, specify a password for the database connection.

13. Click **Next**. The Connection dialog box appears.

14. Enter the following information:

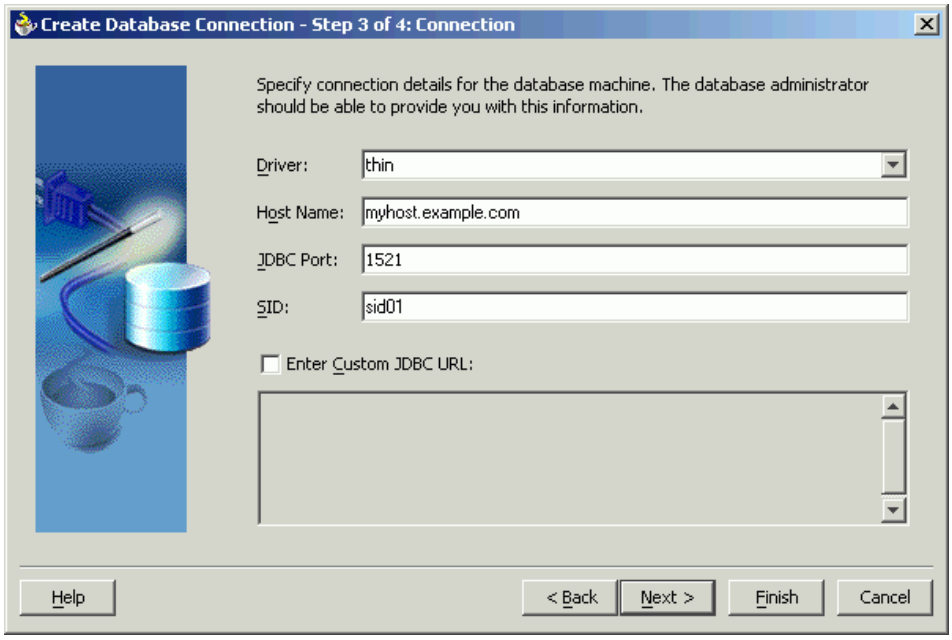
    1. From the **Driver** list, select **Thin**.

    2. In the **Host Name** field, specify the host name for the database connection.

    3. In the **JDBC Port** field, specify the port number for the database connection.

    4. In the **SID** field, specify the unique SID value for the database connection.

Specifying Connection Details for the New Database



15. Click **Next**. The Test dialog box appears.

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

17. Click **Next**. The Service Connection dialog box appears, providing a summary of your database connection.

18. The JNDI (Java Naming and Directory Interface) name corresponding to the database connection appears automatically in the **Database Server JNDI Name** field. Alternatively, you can specify a JNDI name.

    Note: When you specify a JNDI name, the deployment descriptor of the Oracle Applications adapter must associate this JNDI name with configuration properties required by the adapter to access the database.

    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 for production.

    Note: For more information about JNDI concepts, refer to *Oracle Application Server Adapter Concepts.*

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

    Once you have completed creating a new connection for the service, you can add a PL/SQL API by browsing through the list of APIs available in Oracle Applications.

20. Click **Next**.

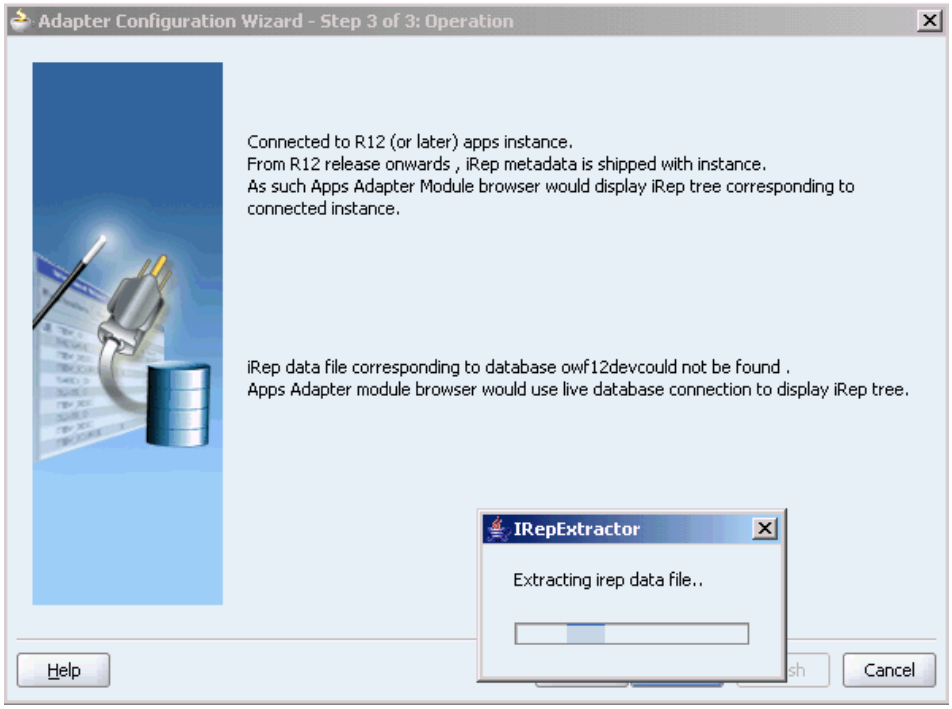
    For Oracle E-Business Suite Release 12:

    If you are connecting to Oracle E-Business Suite Release 12, then the **IREP File not present** dialog box appears indicating that OracleAS Adapter could not find the Oracle Integration Repository data file corresponding to the database you are connecting to Oracle Applications in your workspace. Absence of the data file would make browsing or searching of Integration Repository tree considerably slow. You can choose to extract the data file and create a local copy of the Integration Repository data file. Once it is created successfully, OracleAS Adapter will pick it up automatically next time and retrieve data from your local Integration Repository.

    You can select one of the following options:

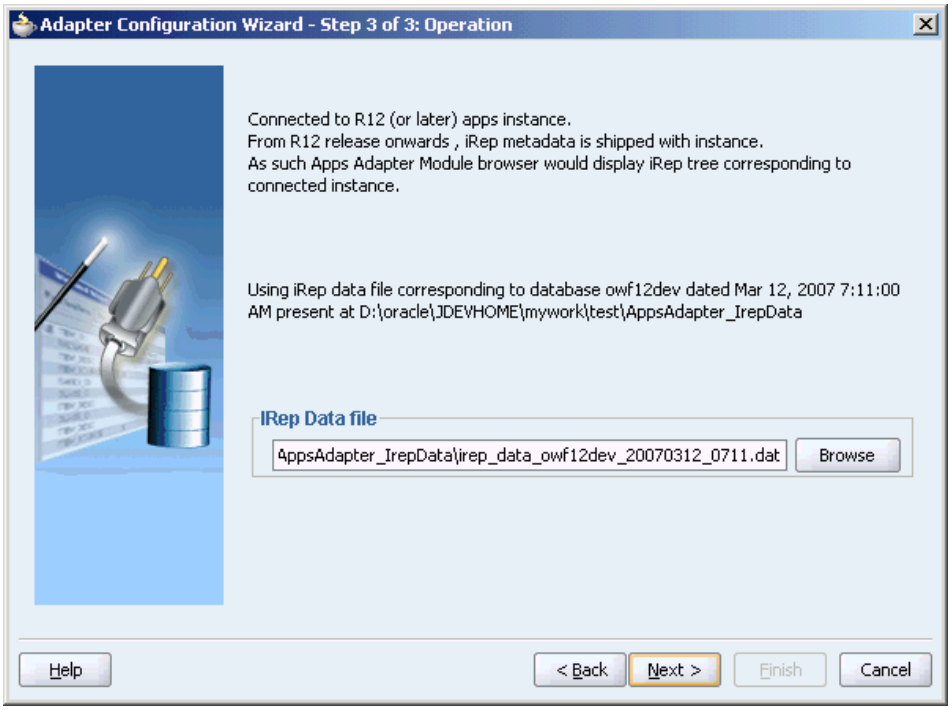
    - Click **Yes** to extract the Integration Repository data file.

Extracting Integration Repository Data File



After the system successfully creates a local copy of the Integration Repository data file, next time when you connect to the database, you will find the **IRep Data File** field appears in the Operation dialog box indicating where your local copy exists with the creation date and time as part of the file name.

Using the Local Integration Repository Data File



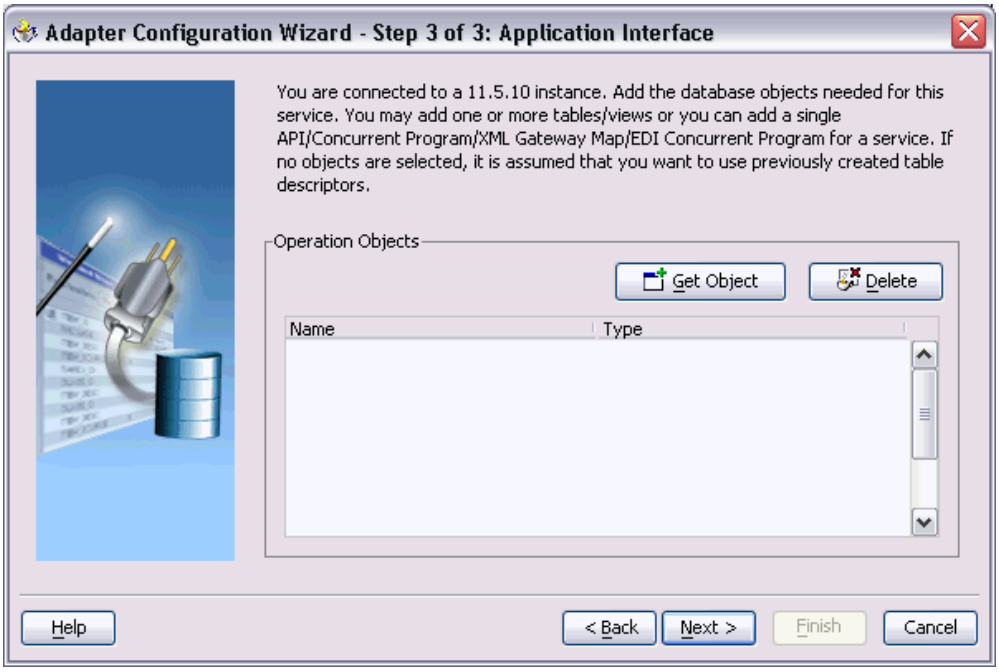
- Click **No** to query the Integration Repository data file from the live database you are connecting to display the Integration Repository tree.

  Note: It is highly recommended that you create a local copy of the Integration Repository data file so that OracleAS Adapter will query the data next time from the local copy in your workspace to enhance the performance.

For Oracle E-Business Suite pre-Release 11.5.10:

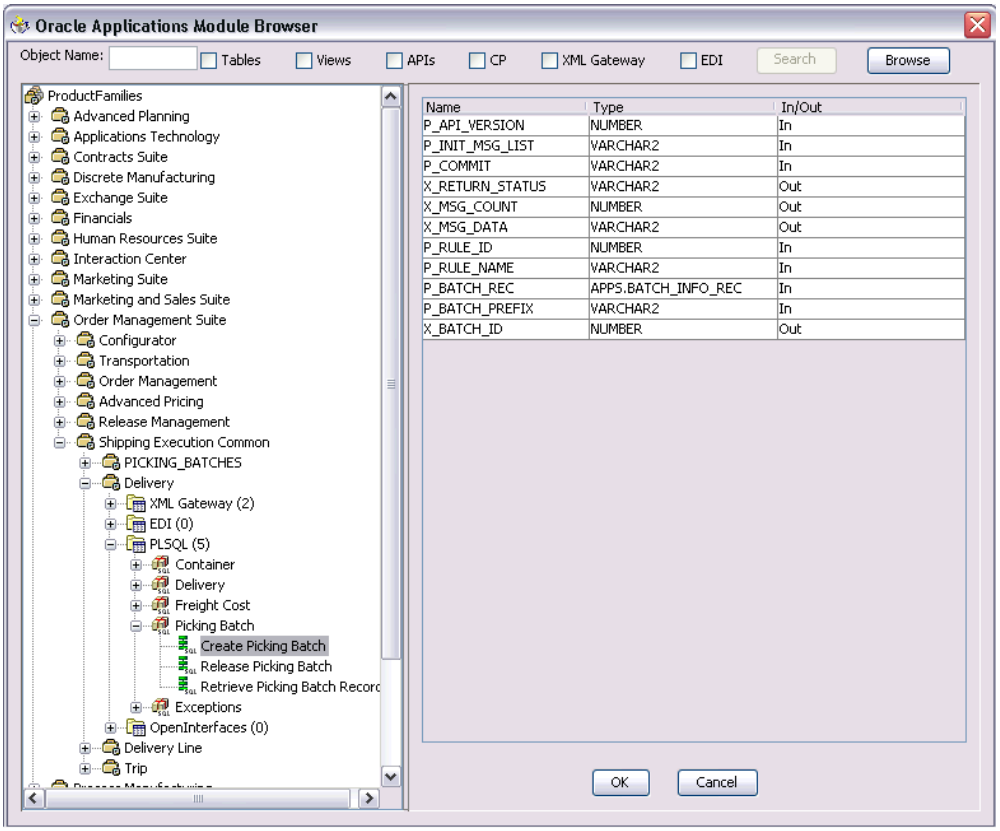
If you are connecting to a pre-11.5.10 Oracle Applications instance, you must select the interface type in the Adapter Configuration Wizard. Select **Tables/Views/APIs/Concurrent Programs** to proceed.

Adding the Database Objects



21. Click **Get Object** in the Application Interface dialog box to open the Oracle Applications Module Browser.

Specifying the PL/SQL API



22. Oracle Applications Module Browser includes the various product families that are available in Oracle Applications. For example, the Marketing Suite or the Order Management Suite are product families in Oracle Applications. The product families contain the individual products. For example, the Order Management Suite contains the Shipping Execution Common product. The product contains the business entities associated with the product. For example, the Shipping Execution Common product contains the Delivery entity.

    Business entities contain the various application modules that are exposed for integration. These modules are grouped according to the interface they provide. PL/SQL APIs can be found under the PLSQL category.

    > **Note:** In previous releases, the Module Browser displayed all of the PL/SQL APIs that were available in a package. Starting in release 11.5.10, the Module Browser displays only the public PL/SQL APIs that are exposed to the Oracle Integration Repository.

    You can specify whether customized APIs within an existing (Integration Repository) PL/SQL package are visible in the Module Browser tree. Right-click the

name of a PL/SQL package, and click **Yes** to display all of its APIs (the default), or click **No** to hide the customized APIs.

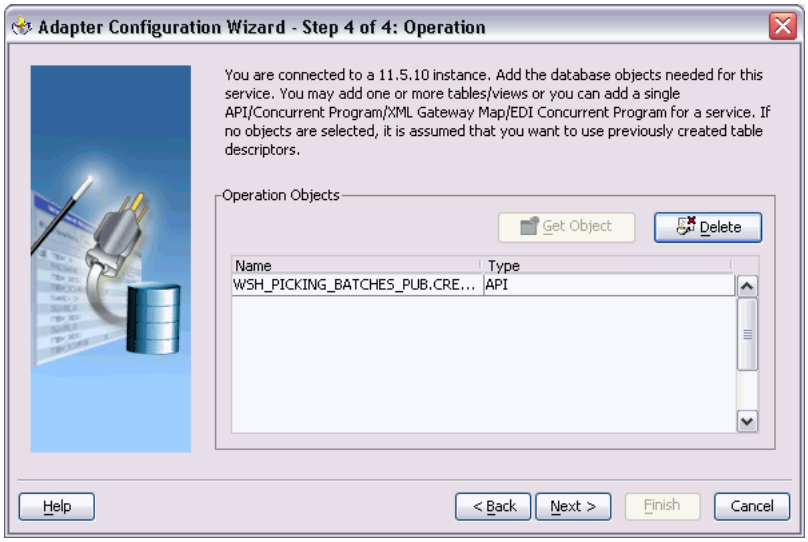
> **Note:** This selection applies separately to each PL/SQL package, and remains in effect only for the duration of the current JDeveloper session.

23. Select the required PL/SQL API. The signature of that given API appears.

24. Click **OK.** You can select only one API for each adapter service.

> **Note:** Use the Search option to quickly find the required objects. Enter the required database object name in the **Object Name** field and select **APIs**. And then, click **Search** to retrieve the required database objects. When searching for a PL/SQL API, enter the package name, and *not* the procedure name. In contrast, when using the DB Adapter Wizard, the user *must* enter the name of the PL/SQL API while searching.

Adding the PL/SQL API



25. Click **Next,** then click **Finish** to complete the process of configuring OracleAS Adapter for Oracle Applications. The wizard generates the WSDL file corresponding to the XML schema. This WSDL file is now available for the partner link.

26. Click **OK**. The partner link is created with the required WSDL settings.

    After adding and configuring the partner link, the next task is to configure the BPEL process.

Defining Wrapper APIs

The Adapter Configuration wizard generates a wrapper API when a PL/SQL API has arguments of data types, such as PL/SQL Boolean, PL/SQL Table, or PL/SQL Record. For generating the wrapper API, Oracle JPublisher is automatically invoked in the background. When a wrapper API is created, besides the WSDL and XSD files, two SQL files are created: one for creating the wrapper API, and necessary datatypes and another for deleting it. The two SQL files are saved in the same directory where the WSDL and XSD files are stored, and are available in the Project view.

Following is a sample code of a PL/SQL API that would require a wrapper to be generated:

```
package pkg is
    type rec is record (...);
    type tbl is table of .. index by ..;
    procedure proc(r rec, t tbl, b boolean);
end;
```

If the preceding PL/SQL API is selected in the wizard, then a wrapper API will be created, and loaded into the database. This wrapper API will be used instead of the originally selected PL/SQL API. For this reason, the content of the WSDL and XSD files represent the wrapper procedure, not the procedure originally selected.

The following are the types that will be created for the wrapper API:

- Object type for PL/SQL RECORD
- Nested table of the given type for PL/SQL TABLE. For example, the nested table of NUMBER.
- INTEGER substituted for PL/SQL BOOLEAN

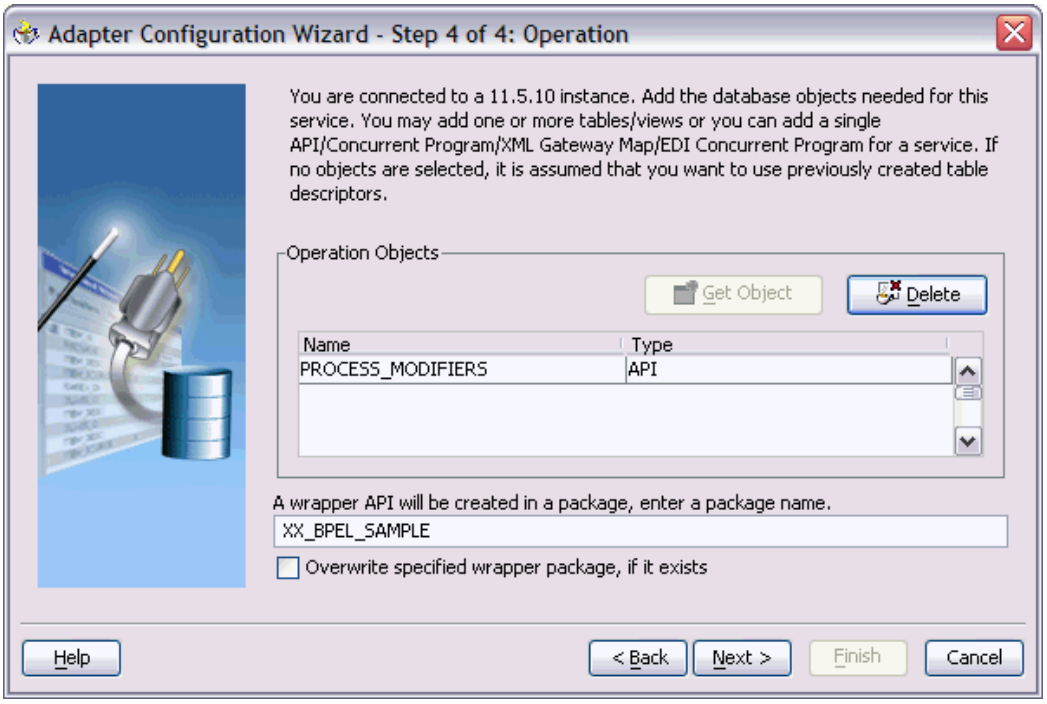
The generated SQL file that creates the wrapper API also creates the required schema objects. The types of the wrapper API's parameters will be those of the new schema object types. The wrapper package will contain conversion APIs to convert between the base PL/SQL type and the new schema object types.

> **Note:** The `SQLJUTL` package contains the BOOL2INT and INT2BOOL conversion functions used for PL/SQL BOOLEAN arguments whose data types have been changed to INTEGER.

The wrapper API is created in a package. This package is named XX_BPEL_*servicename*. *servicename* is the name of the service that you entered in step 6 of Adding a Partner Link, page 6-6. If this package already exists, then the wizard prompts for a different package name, or to select a checkbox, to overwrite the existing package. Overwriting an existing package causes all PL/SQL APIs in the specified package to be lost. When the wizard creates a package for the wrapper, only one API, that is, the wrapper API, is contained in it.

> **Note:** Despite specifying to overwrite an existing package, if the wrapper API already exists in the specified package, the wizard will not re-create the wrapper API, as it would take some time. This means no SQL files will be created, Oracle JPublisher will not be run, and the WSDL and XSD files will be for the existing wrapper API. The Finish page of the wizard will indicate that these actions will take place, but it is possible that they will not, depending on whether the wrapper already exists.

Entering a Package Name for the Wrapper API



> **Note:** The package name for the wrapper has a limit of 30 characters, and the wrapper API name has a limit of 29 characters. Thus, if the package name or the wrapper API name is longer than the maximum limit, it will be truncated accordingly.

The name of the wrapper API depends on whether the PL/SQL API that was originally selected is in a package or not. If the original PL/SQL API is a root-level API, that is, it does not belong in a package, then the name of the wrapper API will be, TOPLEVEL$ *original_api_name.* If the originally selected PL/SQL API is in a package, then the name of the wrapper API will be *original_package_name*$*original_api_name.*

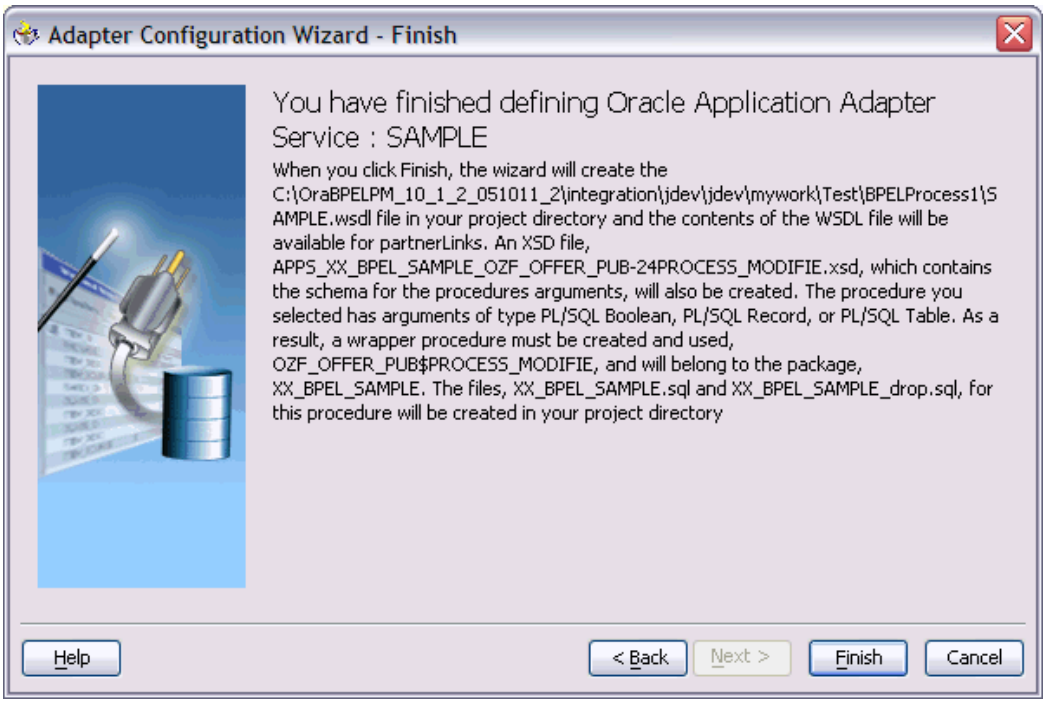
Wrapper APIs follow the naming convention of Oracle JPublisher. For example, if the original PL/SQL API was called `CREATE_EMPLOYEE` and was a root-level API, then the wrapper API would be named `TOPLEVEL$CREATE_EMPLOYEE`. If the original PL/SQL API is in a package called `EMPLOYEE`, then the wrapper API would be named `EMPLOYEE$CREATE_EMPLOYEE`.

The Finish page of the wizard is different when a wrapper API needs to be created. The Finish page informs you that a wrapper API is needed, and in addition, lists the name of the wrapper package, wrapper API, and the SQL files that will be created.

> **Note:** When a wrapper API needs to be created, it may take a while before the wizard completes. However, the processing time for

subsequent PL/SQL APIs in the same package would be much shorter.

The Finish Page



Note: The `REF CURSOR` type is not supported out of the box. However, for detailed steps to generate an adapter service for a PL/SQL API which takes `REF CURSOR` type, see the section "Support for REF CURSOR" in *Oracle BPEL Process Manager Developer's Guide* on OTN.

Overloaded APIs are not supported in release 11.5.10.

Declaring Parameters with a DEFAULT Clause

You can declare parameters of a stored procedure with a DEFAULT clause, so that when you invoke the procedure without that parameter, BPEL Process Manager will supply a default value for that parameter. For example:

```
PROCEDURE addEmployee (name VARCHAR2, country VARCHAR2 DEFAULT 'US')
```

This procedure can be invoked in the following two ways:

- `addEmployee ('John Smith') // country => 'US'`
- `addEmployee ('John Smith', 'France') // country => 'France'`

Omitting Parameters With a DEFAULT Clause

You can omit elements for parameters with default values in the instance XML. The procedure will be invoked without these parameters, allowing their default values to be used, as shown in the following example of input and runtime invocation.

Input

```
<db:InputParameters xmlns:db="…">
    <name>John Smith</name>
</db:InputParameters>
```

Runtime Invocation

```
BEGIN addEmployee (name=>?); END; // country => 'US'
```

If the input includes a value for the defaulted parameter, the value in the input will be used, rather than the default, as follows:

Input

```
<db:InputParameters xmlns:db="…">
    <name>John Smith</name>
    <country>France</country>
</db:InputParameters>
```

Runtime Invocation

```
BEGIN addEmployee (name=>?, country=>?); END; // country => 'France'
```

Omitting Parameters Without a DEFAULT Clause

The element in the XSD for parameters with a default clause is annotated with a special tag to indicate that the parameter has a default clause, as shown in the following example.

```
<element name="country" … db:default="true" …/>
```

This new functionality allows elements for parameters without a default clause also to be omitted in the instance XML. In these cases, the parameter is still included in the invocation of the stored procedure. A value of NULL is bound by default. Following is an example of a declaration where neither parameter has a DEFAULT clause:

```
PROCEDURE addEmployee (name VARCHAR2, country VARCHAR2)
```

In BPEL Process Manager release 10.1.2, elements for both parameters were required in the instance XML. If an element was omitted, it was presumed to have a DEFAULT clause, so the parameter was not included in the invocation of the procedure. In this case, the missing parameter resulted in a PL/SQL error stating that an incorrect number of arguments was passed to the procedure.

In the current BPEL Process Manager release, the missing parameter will be included in the invocation of the procedure. A `NULL` value will be bound, as shown in the following example:

Input

```
<db:InputParameters xmlns:db="…">
    <name>John Smith</name>
</db:InputParameters>
```

Runtime Invocation

```
BEGIN addEmployee (name =>?, country=>?); END; // country => NULL
```

Even though the element for *country* was not provided in the instance XML, it still appears in the call to the procedure. In this case, *country* will be `NULL`.

DEFAULT Clause Handling in Wrapper Procedures:

If a procedure contains a special type requiring a wrapper to be generated, the `default` clauses on any of the original parameters will *not* be carried over to the wrapper, as shown in the following example:

```
PROCEDURE needsWrapper(isTrue BOOLEAN, value NUMBER DEFAULT 0)
```

Assuming that the procedure in the preceding example was defined at the top level, outside of a package, the generated wrapper will appear, as shown in the following example:

```
TOPLEVEL$NEEDSWRAPPER (isTrue INTEGER, value NUMBER)
```

In the preceding example, the BOOLEAN type has been replaced by INTEGER. The default clause on the value parameter is missing. In the current release, parameters of generated wrapper procedures will never have a default clause, even if these parameters did in the original procedure. If the element is missing in the instance XML, instead of defaulting to `0`, then the value of the parameter will be `NULL`, as shown in the following example:

Input

```
<db:InputParameters xmlns:db="…">
    <isTrue>1</isTrue>
</db:InputParameters>
```

Runtime Invocation

```
BEGIN TOPLEVEL$NEEDSWRAPPER (isTrue =>?, value =>?); END; // value =>
NULL
```

To fix this, you can edit the generated SQL file, restoring the default clauses. You should then, run the SQL file to reload the wrapper definitions into the database schema. In addition, you should modify the generated XSD.

Following are the steps to fix the default clause with the wrapper generated for `needsWrapper()`:

1. Change the signature in the following manner in the generated wrapper SQL file, *from:*

   ```
   TOPLEVEL$NEEDSWRAPPER (isTrue INTEGER, value NUMBER)
   ```

   To:

   ```
   TOPLEVEL$NEEDSWRAPPER (isTrue INTEGER, value NUMBER DEFAULT 0)
   ```

2. Reload the modified wrapper SQL file mentioned in the preceding example into the appropriate database schema. For BOOLEAN parameters with DEFAULT clause, you need to map as follows:

   ```
   DEFAULT TRUE(base) to DEFAULT 1 (wrapper)
   DEFAULT FALSE (base) to DEFAULT 0 (wrapper)
   ```

   For example, if the base stored procedure is `PROCEDURE needsWrapper(isTrue BOOLEAN TRUE, value NUMBER DEFAULT 0)`, then the generated wrapper would be `TOPLEVEL$NEEDSWRAPPER (isTrue INTEGER, value NUMBER)`.

You should manually fix the store procedure to be `TOPLEVEL$NEEDSWRAPPER (isTrue INTEGER DEFAULT 1, value NUMBER DEFAULT 0)`

3. If a parameter has a default clause, then its corresponding element in the XSD must have an extra attribute, `db:default="true"`. For example, if a parameter has a default clause `TOPLEVEL$NEEDSWRAPPER(isTrue INTEGER DEFAULT 1, value NUMBER DEFAULT 0)`, then the elements in the XSD for `isTrue` and `value` need to have the following new attributes:

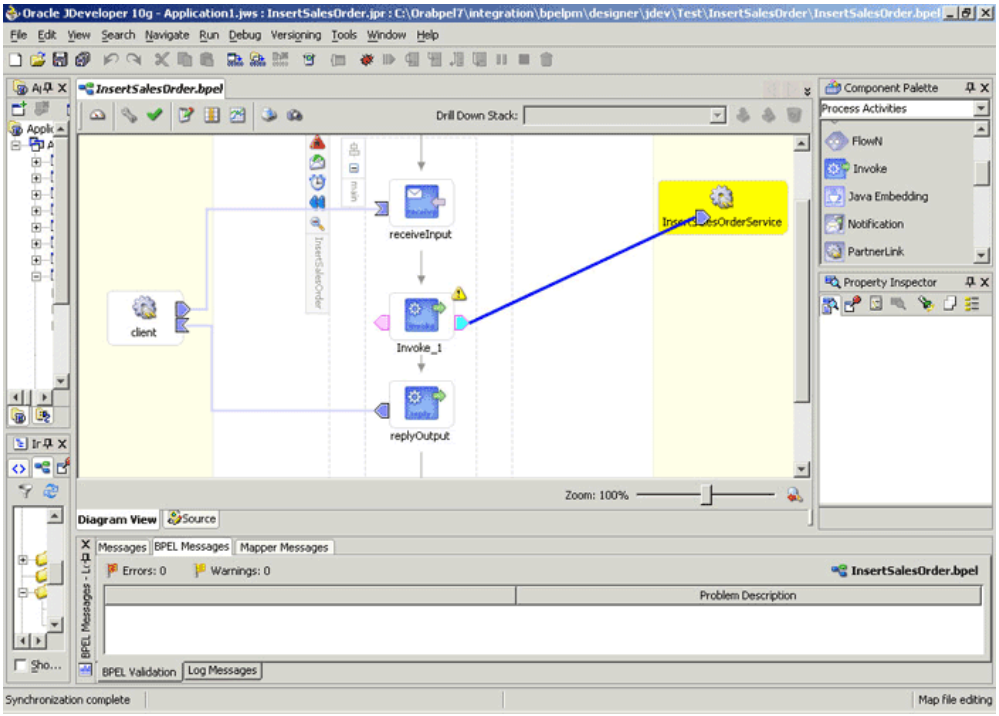
```
<element name="ISTRUE" ... db:default="true" .../>
<element name="VALUE" ... db:default="true" .../>
```

Configuring the Invoke Activity

To configure the Invoke activity:

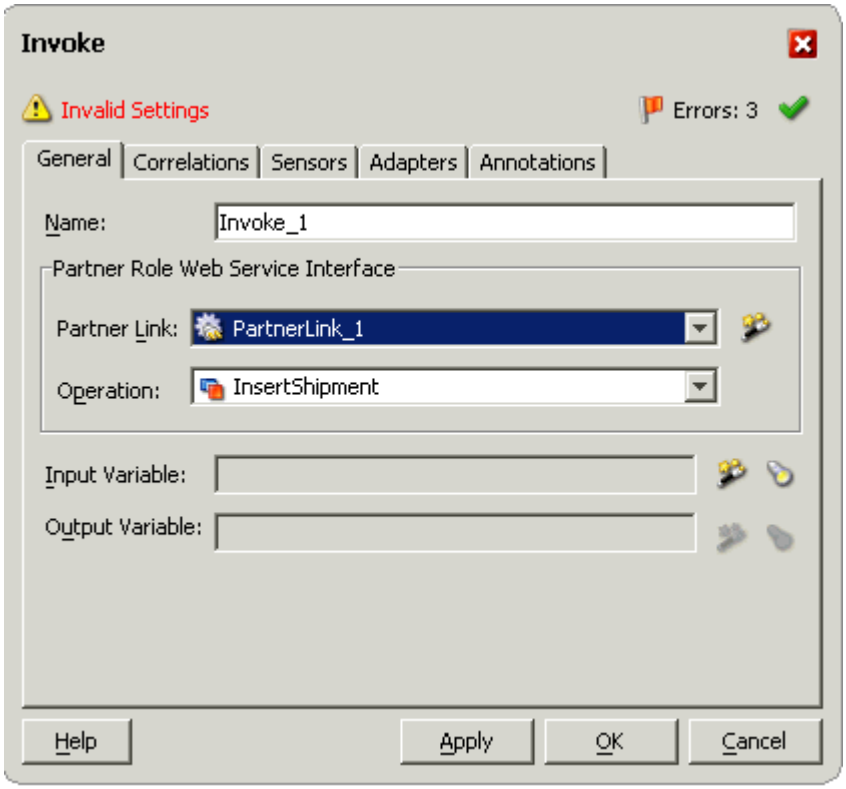
1. Drag **Invoke** from the Component palette and drop it at the location where you want to insert the invoke activity in your BPEL process.

Dragging the Invoke Activity



2. Double-click **Invoke** in the process map to open the Invoke dialog box. The General tab is selected by default.

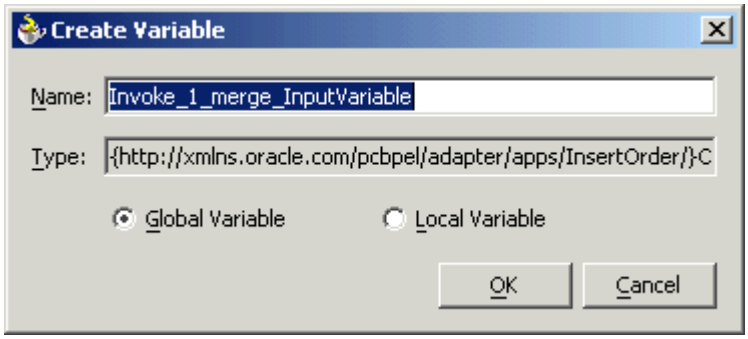
Configuring the Invoke Activity



3. In the **Partner Link** box, select the partner link to invoke. This is the partner link that you configured in the previous section. The **Operation** is automatically selected.

4. Click the **Create** icon next to the Input Variable field. Enter a descriptive name for the variable in the Create Variable dialog box that appears. You can also accept the default name. Click **OK.**

Creating a Variable



5. Click the **Create** icon next to the Output Variable field. Enter a descriptive name for the variable in the Create Variable dialog box that appears. You can also accept the default name. Click **OK**.

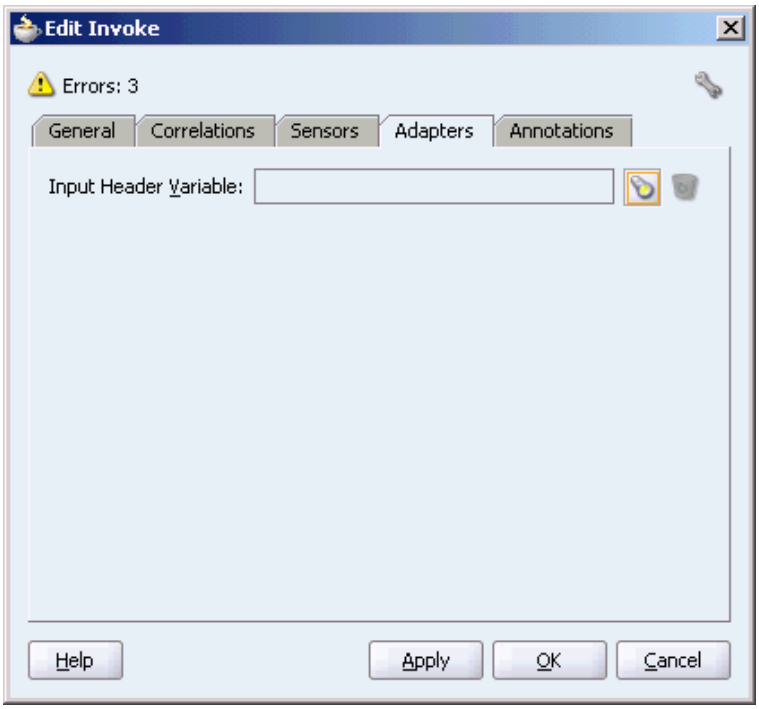
6. In the Invoke dialog box, click **Apply**, and then click **OK**. The invoke activity is configured.

Note: You can define an **Input Header Variable** under the **Adapters** tab of the Invoke dialog box. This variable can be used to provide context information for Oracle Applications in supporting multiple languages and multiple organization setups.

To Create the Header Variable:

1. Click on Adapters tab in the Edit Invoke dialog box and click **Browse Variable...** icon for the Input Header Variable field.

Specifying Input Header Variable

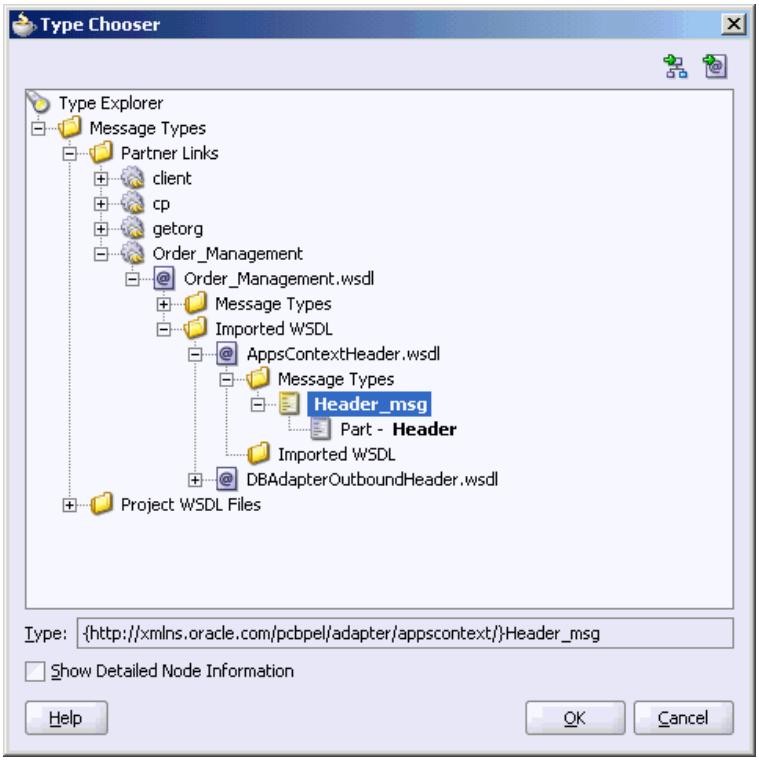


2. In the Variable Chooser dialog box, right-click on the Variables node and select Create Variable option from the menu.

3. Select **Message Type** and click **Browse Message Type...** icon to open Type Chooser dialog box.

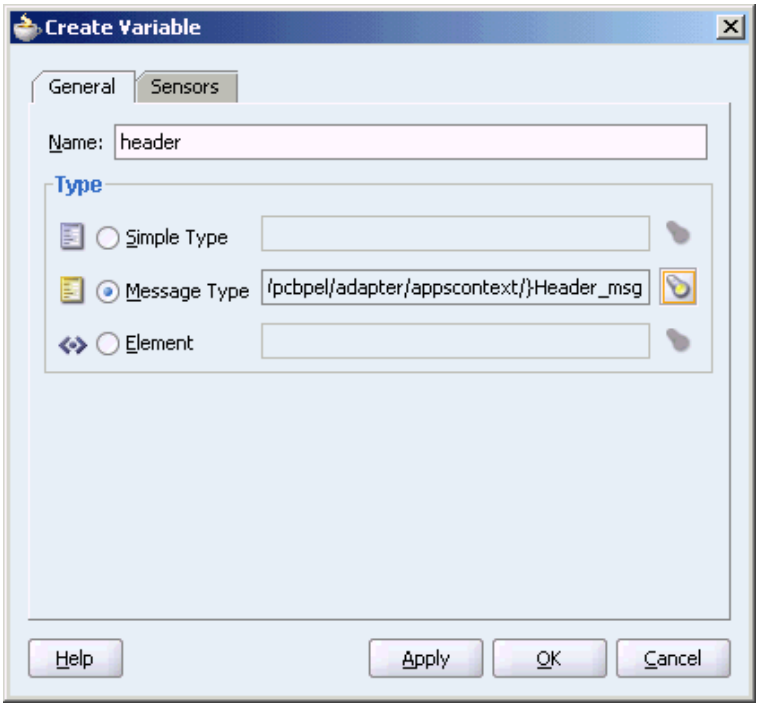
4. Expand the Partner Link node and locate the Header_msg node `{http://xmlns.oracle.com/pcbpel/adapter/appscontext/}Header_msg` for your partner link. The Header_msg node should be under the path, Partner Links node >*Your Partner Link WSDL* > Imported WSDL > AppsContextHeader.wsdl > Message Types > Header_msg.

Declaring Header Variable



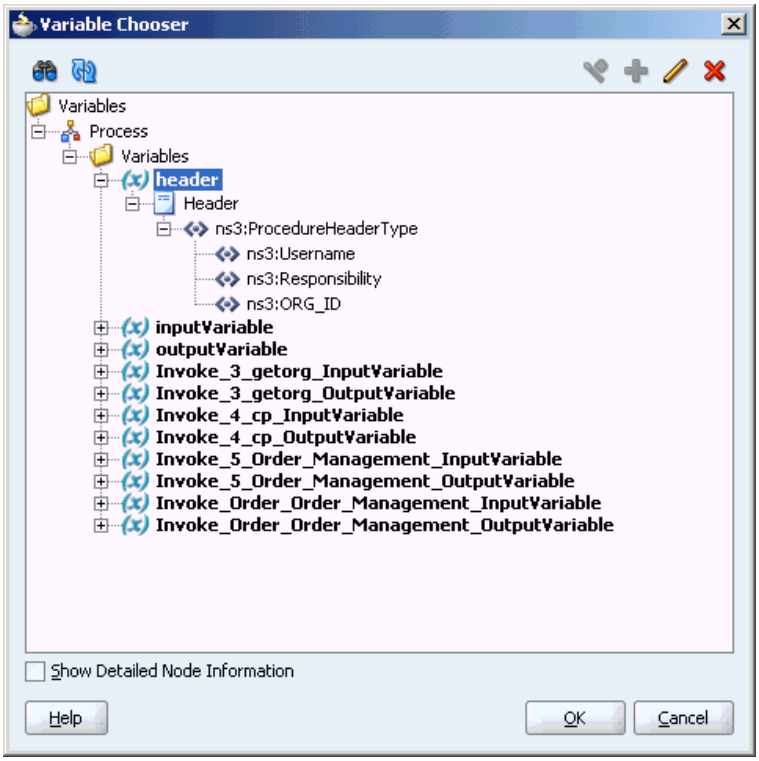
5. Click **OK** to return to the Create Variable dialog box with your selected message type populated.

Populating Selected Header Variable



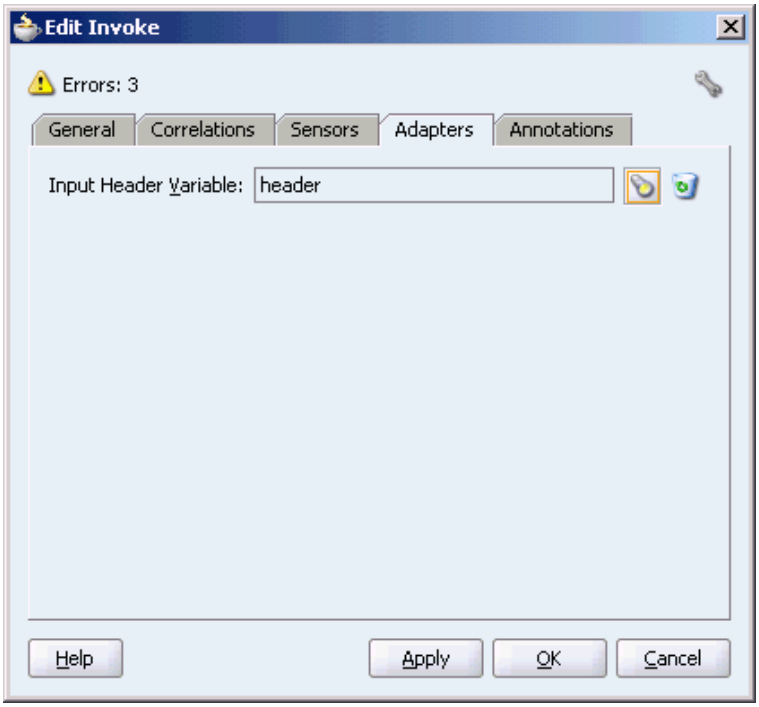
6. To view your header variable with Username, Responsibility, and ORG_ID, click **Browse Message Type...** icon to open Variable Chooser dialog box. Locate the header variable to view the variable hierarchical structure with these three parameters needed for applications context.

Viewing Header Variable Structure



7. Click **OK** to return to the Edit Invoke dialog with the selected header variable populated for the Input Header Variable field. Click **Apply** to complete the header creation.

Populating Input Header Variable



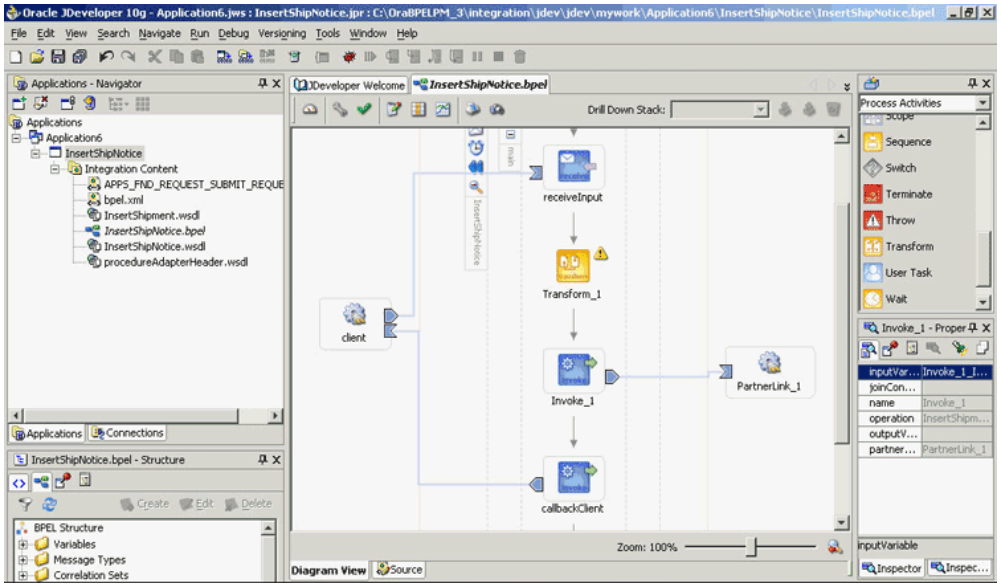
Configuring the Transform Activity

The Transform activity can be used to configure the parameters for the input and output variables. The Transform activity can also be used if variable values need to be transformed before updating them in Oracle Applications.

To configure the Transform activity:

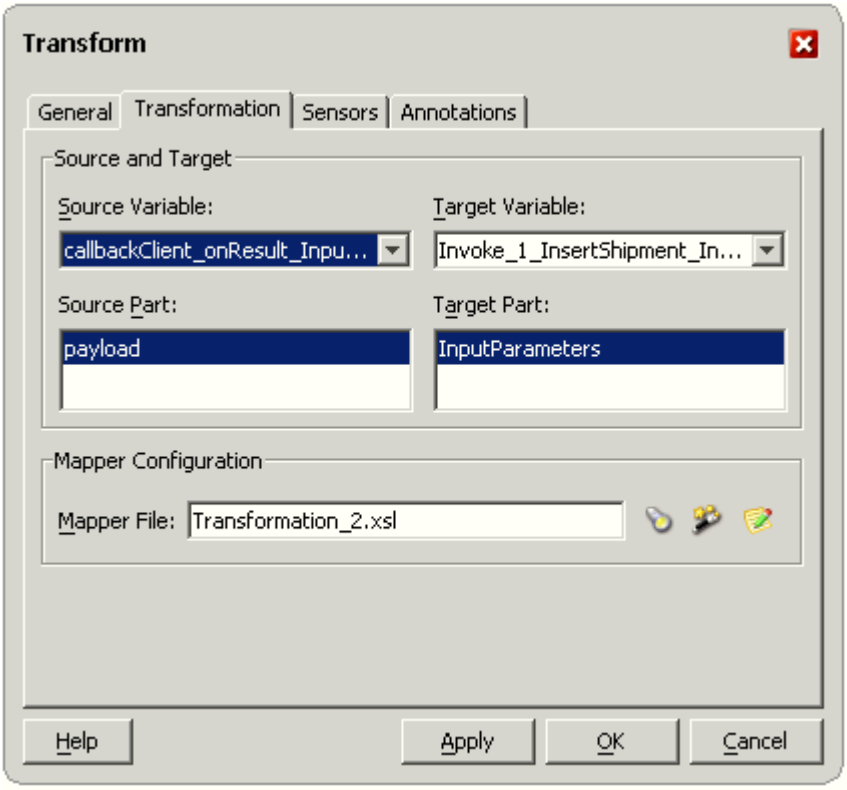
1. Drag and drop **Transform** into the process map window. The **Transform** activity should be placed in between **Receive** and **Invoke.**

Adding the Transform Activity



2. Double-click **Transform** in the process map to open the Transform dialog box. The Transformation tab is selected by default.

Configuring the Transform Activity



3. Select the **Source Variable** and **Target Variable** from the respective boxes. Elements are mapped from the **Source Variable** to the **Target Variable**.

4. Select the **Source Part** of the variable from which to map elements. For example, the source part may be a payload schema consisting of a purchase order request.

5. Select the **Target Part** of the variable to which to map elements. For example, the target part may be a payload schema consisting of a purchase order acknowledgment.

6. Click the **Create** icon next to the Mapper File field to create a new transformation mapping file. Mapper File specifies the file in which you create the mappings using the XSLT Mapper Transformation tool.

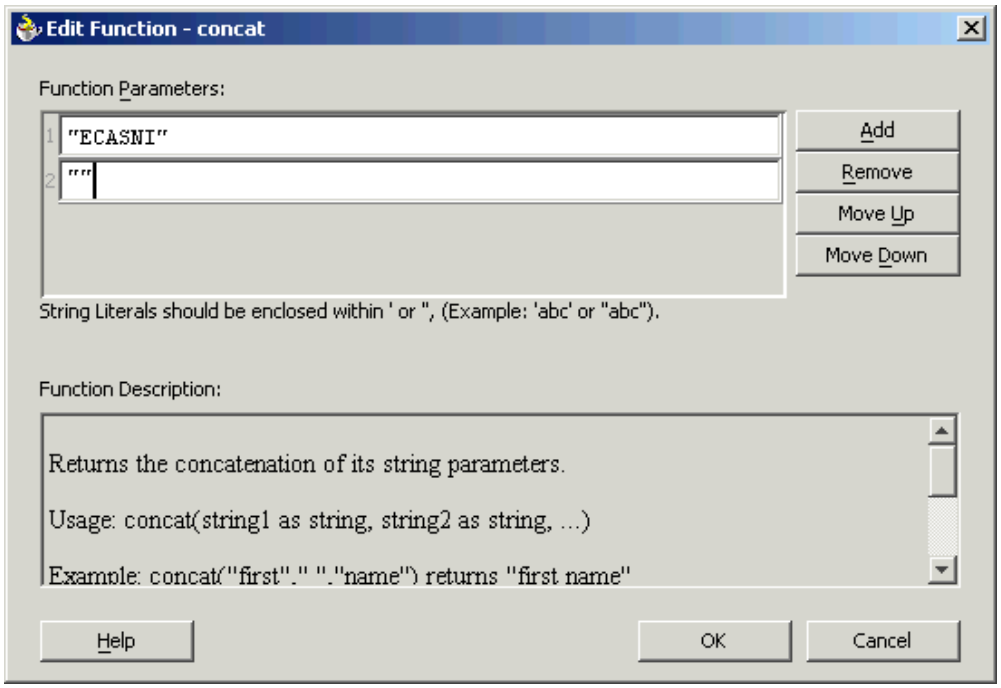
7. The transformation mapping file appears. The **Design** view appears by default.

8. You can define the parameter values in the **Design** view. Drag a string function to the Design area. Connect the function to the appropriate parameter for which you want to define a value.

Note: You can use an input parameter value from the source variable, transform it using a string function, and use it as the input parameter value for the target variable.

9. Double-click the icon for the function. The Edit Function dialog box appears.

Supplying the Function Parameters



10. Repeat steps 8 and 9 for all the parameters that you need to supply.

Run-Time Tasks for PL/SQL APIs

After designing the BPEL process, the next step is to deploy, run and monitor it.

1. Deploy the BPEL Process., page 6-33
2. Test the BPEL Process., page 6-35

Deploying the BPEL Process

You must deploy the BPEL process before you can run it. The BPEL process is first compiled, and then deployed to the BPEL server.

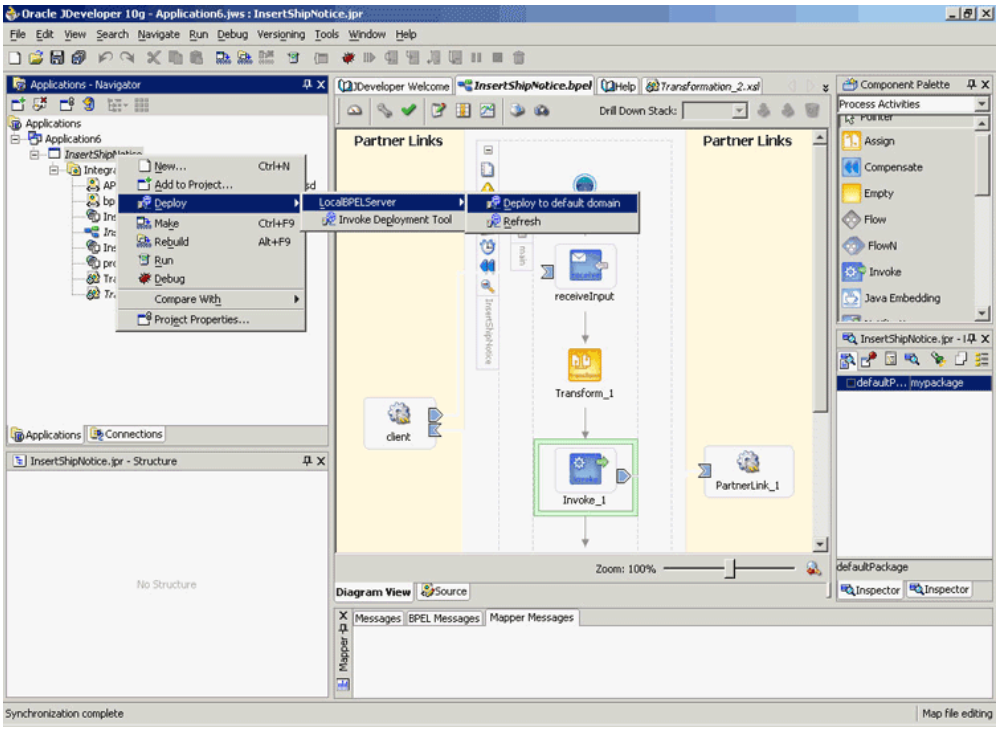
To deploy the BPEL process:

1. Select the BPEL project in the Applications window.

2. Right-click the project name, and then select **Deploy** from the menu that appears.

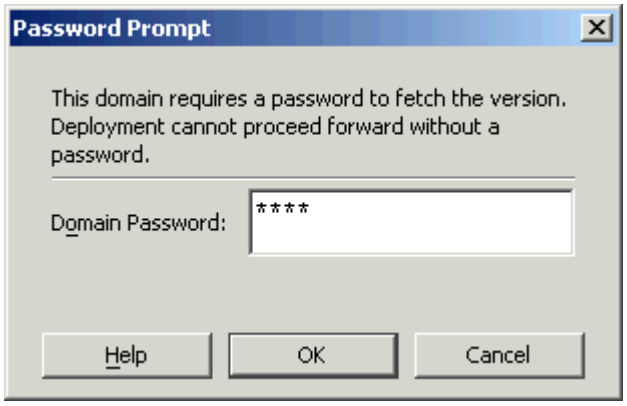
3. Select **Local BPEL Server** followed by **Deploy to Default Domain**, if you are deploying the process on the local BPEL server.

Deploying the BPEL Process



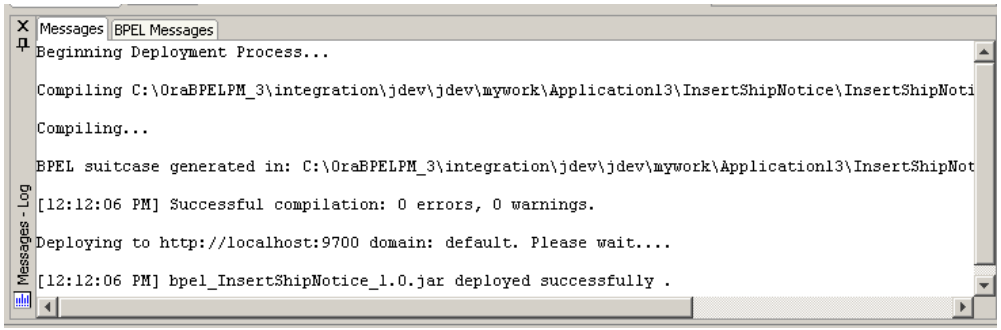
4. The Password Prompt dialog box appears. Enter the password for the default domain in the **Domain Password** field, and then click **OK**.

Specifying the Domain Password



5. The BPEL process is compiled and deployed. You can check the progress in the Messages window.

Messages Window



Testing the BPEL Process

Once the BPEL process is deployed, it can be seen in the BPEL console. You can manage and monitor the process from the BPEL console. You can also test the process and the integration interface by manually initiating the process.

To test the BPEL process:

1. To open the BPEL console, click **Start**, and then choose **Programs**. In the Programs menu, select **Oracle - ORACLE_HOME**, **Oracle BPEL Process Manager 10.1.3**, and then select **BPEL Console**.

   The BPEL console login page appears.

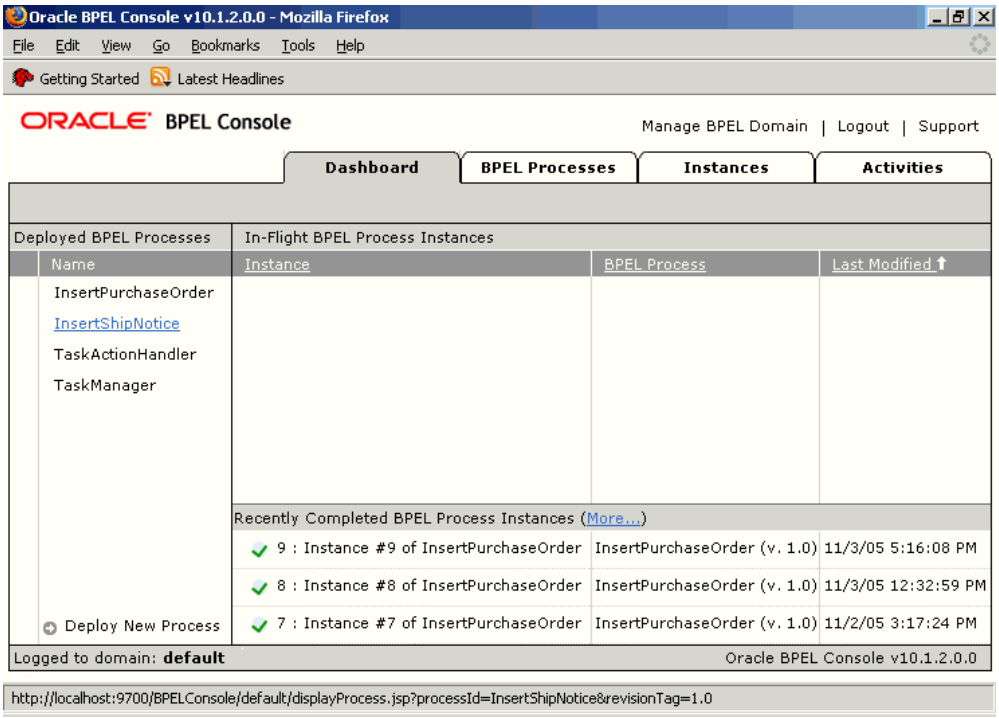
2. Select **Default** in the **Domain** box. Enter the password for the default domain in the

Password field, and then click **Login**.

The Oracle BPEL console appears.

3. A list of deployed processes is shown under Deployed BPEL Processes.

Deployed BPEL Processes

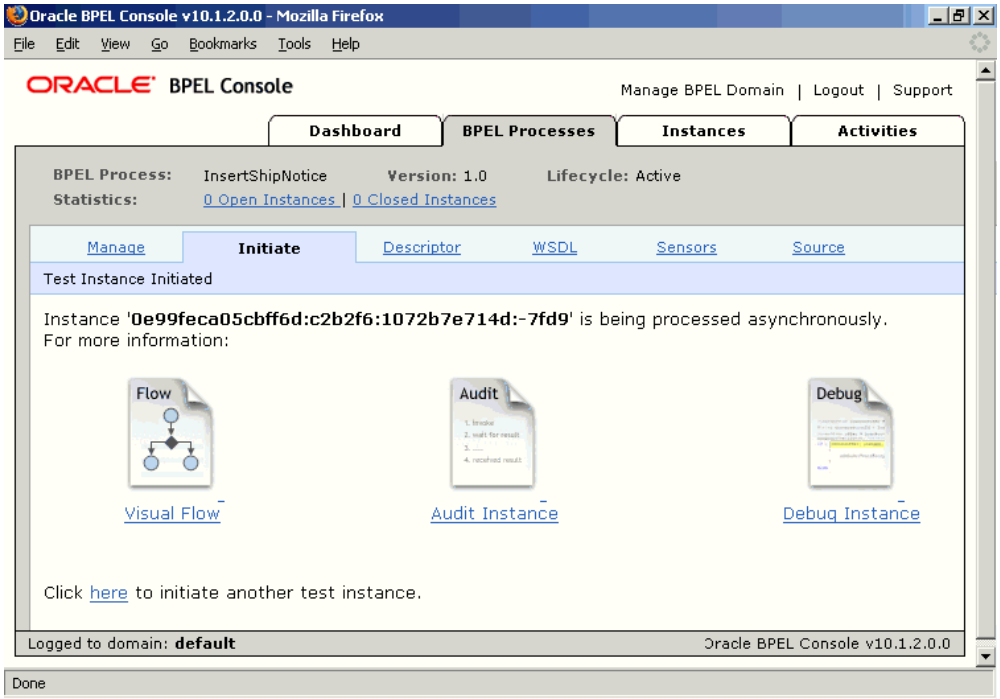


4. Click the BPEL process that you want to initiate. The Initiate page appears. Enter the input string required by the process.

5. Click **Post XML Message** to initiate the process.

6. The BPEL process is now initiated. You can check the process flow by clicking the **Visual Flow** icon.

BPEL Console Initiate Page



Note: To confirm that the records have been written into the Oracle Applications, you can write SQL `SELECT` statements and fetch the results showing the latest records inserted into Oracle Applications. Alternatively, you can go to the specific module in Oracle Applications and verify the appropriate changes in the records.

Troubleshooting and Debugging

If you experience problems with your PL/SQL API integration, you can take the following troubleshooting steps:

- If you designed your PL/SQL API integration in one instance of your Oracle Applications environment, and plan to run it in another instance, be sure to rerun the SQL scripts in the second instance to create the necessary PL/SQL wrappers.
- Ensure that the JNDI location in **oc4j-ra.xml** is properly configured.
- Use **encrypt.bat** to encrypt the passwords in **oc4j-ra.xml** if needed.

If you still experience problems with your integration, you can enable debugging.

Enabling Debugging

You can enable debugging for PL/SQL APIs using the BPEL Process Manager.

To enable debugging:

1. Log into your BPEL Process Manager domain.
2. Select `yourdomain.collaxa.cube.ws`
3. Select **Debug.**

Debugging information is output to the log file for your domain. To examine the log file in the BPEL Process Manager, navigate to **Home > BPEL Domains >*yourdomain* > Logs.** The log file is ***yourdomain*.log**.

7

Using e-Commerce Gateway

This chapter covers the following topics:

- Overview of e-Commerce Gateway Integration
- Design-Time Tasks for e-Commerce Gateway
- Creating a New BPEL Project
- Adding a Partner Link
- Configuring the Invoke Activity
- Configuring the Transform Activity
- Run-Time Tasks for e-Commerce Gateway
- Deploying the BPEL Process
- Testing the BPEL Process
- Verifying Records in Oracle Applications

Overview of e-Commerce Gateway Integration

Oracle e-Commerce Gateway provides a common, standards-based approach for Electronic Data Interchange (EDI) integration between Oracle Applications and third party applications. EDI is the computer to computer exchange of business documents in a standard format. The EDI format is commonly used for e-commerce transactions between businesses.

Oracle e-Commerce Gateway is the EDI integration enabler for Oracle Applications. It provides a single integration framework for you to conduct e-business using EDI standards with everyone in your global supply chain. Oracle e-Commerce Gateway provides an application integration infrastructure that is flexible enough to accommodate the integration requirements of any and all applications that must integrate with Oracle Applications. This allows for seamless flow of information in an ever expanding trading partner base.

Oracle e-Commerce Gateway includes pre-built transactions of key business documents

that can be implemented simply by defining a trading partner and enabling the transaction in test or production mode. You can implement a single transaction, a group of transactions, or a business flow. You can implement the pre-built transactions as is or configure them to meet your specific industry needs.

Oracle e-Commerce Gateway uses a metadata driven approach to dynamically generate outbound and consume inbound flat files based on user defined trading partner, mapping, transformation, and data validation rules. You can change a rule by changing the metadata stored in the repository. The updated rule takes effect at run-time without any code modifications.

> **Note:** For detailed information about Oracle e-Commerce Gateway, see *Oracle e-Commerce Gateway User's Guide.* This guide is a part of the Oracle Applications documentation library. Oracle Applications documentation can be accessed from the following link:
>
> http://www.oracle.com/technology/documentation/applications.html

OracleAS Adapter for Applications can be configured to use e-Commerce Gateway to interact with third party applications. e-Commerce Gateway, like XML Gateway, is primarily used for Business-to-Business (B2B) integration. While XML transactions are mostly based on a single transaction and are event based, EDI transactions are more batch oriented.

Design-Time Tasks for e-Commerce Gateway

OracleAS adapter for Oracle Applications is deployed using the BPEL Process Manager (PM) in Oracle JDeveloper. The BPEL PM creates the WSDL interfaces for the e-Commerce Gateway.

This section describes configuring the OracleAS Adapter for Oracle Applications to use e-Commerce Gateway. It describes the tasks required to configure OracleAS Adapter for Oracle Applications using the Adapter Configuration Wizard in Oracle JDeveloper.

Prerequisites to Configuring e-Commerce Gateway

You need to populate certain variables in the BPEL PM in order to provide context information for Oracle Applications. The context information required for an EDI transaction includes the *username* and *responsibility* of an Oracle Applications user that has sufficient privileges to run the program. The default value passed for the username is SYSADMIN. The default value passed for responsibility is SYSTEM ADMINISTRATOR.

You can change the default values specified in the generated WSDL for the username and responsibility. This is a static way of changing the context information. These values would apply to all invocations of the deployed business process. However, if you need to provide different context information for different invocations of the business process, then you can dynamically populate the header variable with values for username and responsibility. The context information can be specified by

configuring an Assign activity before the Invoke activity in the BPEL PM.

Following is a list of the procedures required to accomplish the design-time tasks.

1. Create a new BPEL project., page 7-3
2. Add a partner link., page 7-6
3. Configure the Invoke activity., page 7-16
4. Configure the Transform activity., page 7-24

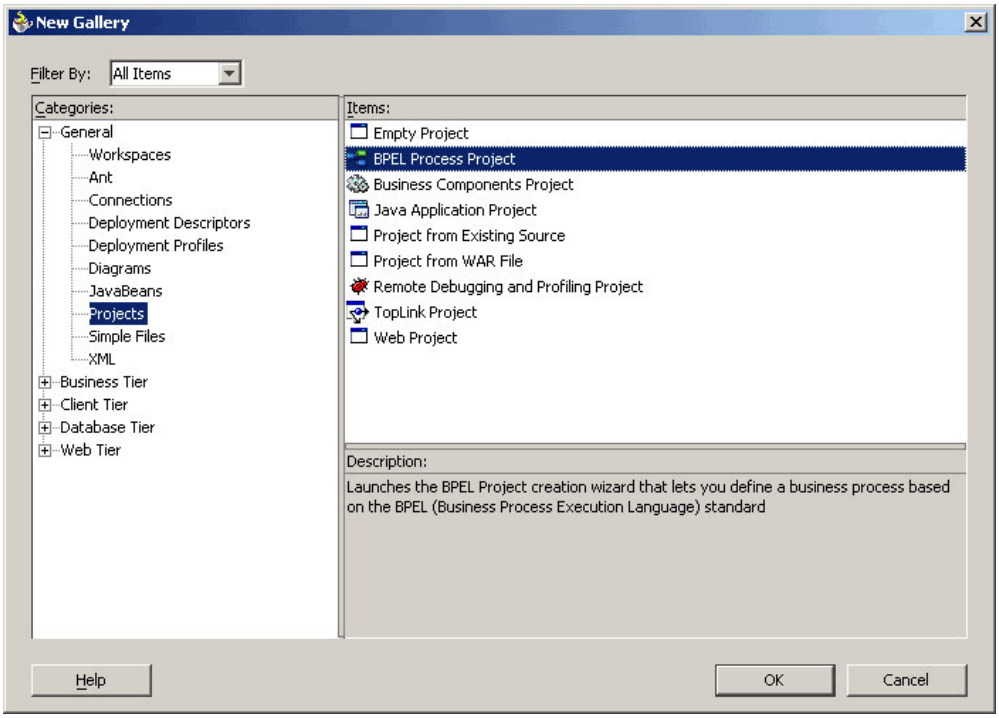
Creating a New BPEL Project

The first configuration task is to create a new BPEL project.

To create a new BPEL project:

1. Open JDeveloper BPEL Designer.
2. From the **File** menu, select **New**. The New Gallery dialog box appears.
3. Select **All Items** 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.

Creating a New BPEL Process Project

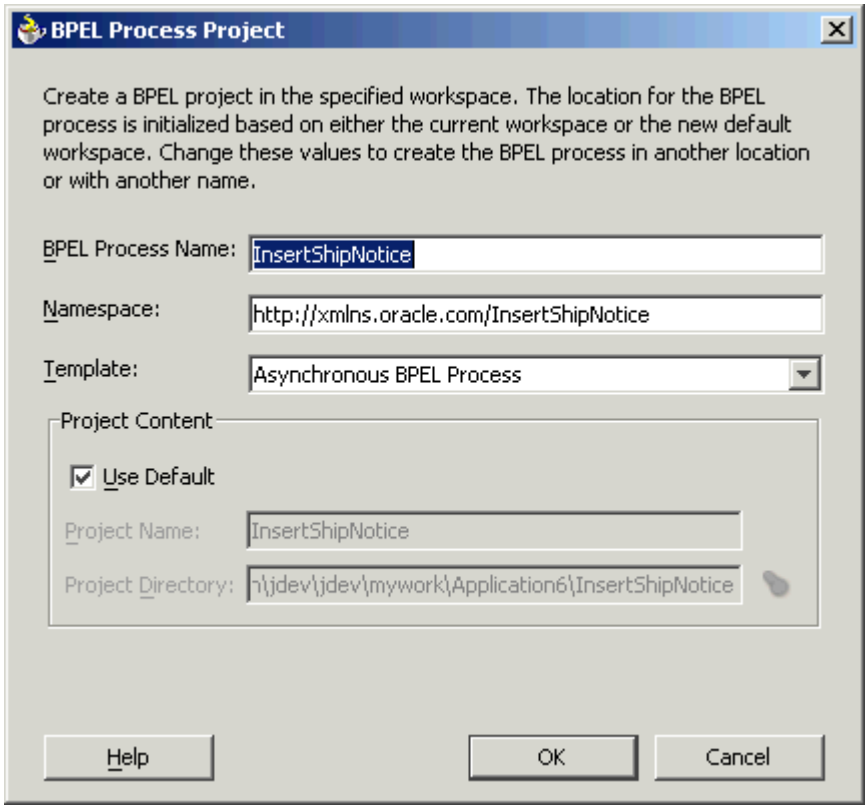


6. Click **OK**. The BPEL Process Project dialog box appears.

7. In the **BPEL Process Name** field, enter a descriptive name. For example, `InsertShipNotice`.

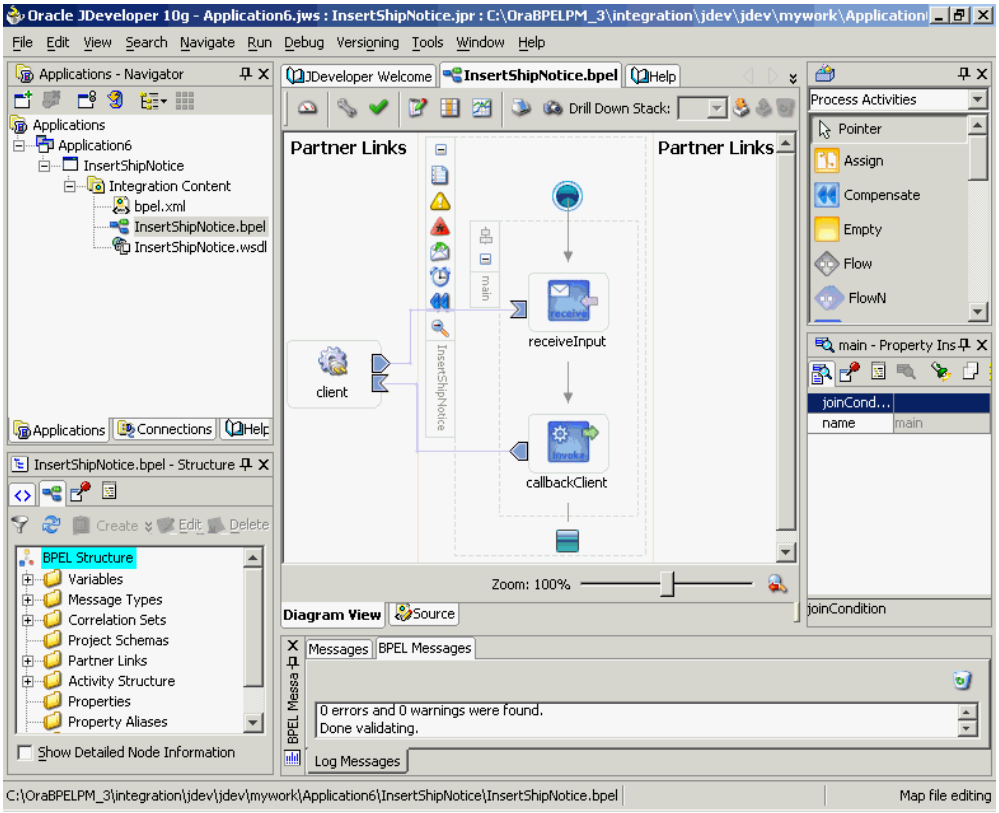
8. From the **Template** box, select **Asynchronous BPEL Process**. Keep the default selection for **Use Default** under Project Content.

Specifying a Name for the New BPEL Process Project



9. Click **OK**. A new BPEL process, with the required source files including `bpel.xml`, `InsertShipNotice.bpel` and `InsertShipNotice.wsdl` is created.

New BPEL Process Project



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.

To add a partner link:

1. Drag and drop **PartnerLink**, from the Component Palette, into the border area of the process diagram. The Create Partner Link dialog box appears.

2. Click the **Define Adapter Service** icon in WSDL Settings. The Adapter Configuration Wizard appears.

3. Click **Next**. The Adapter Type dialog box appears.

4. Select **Oracle Applications.**

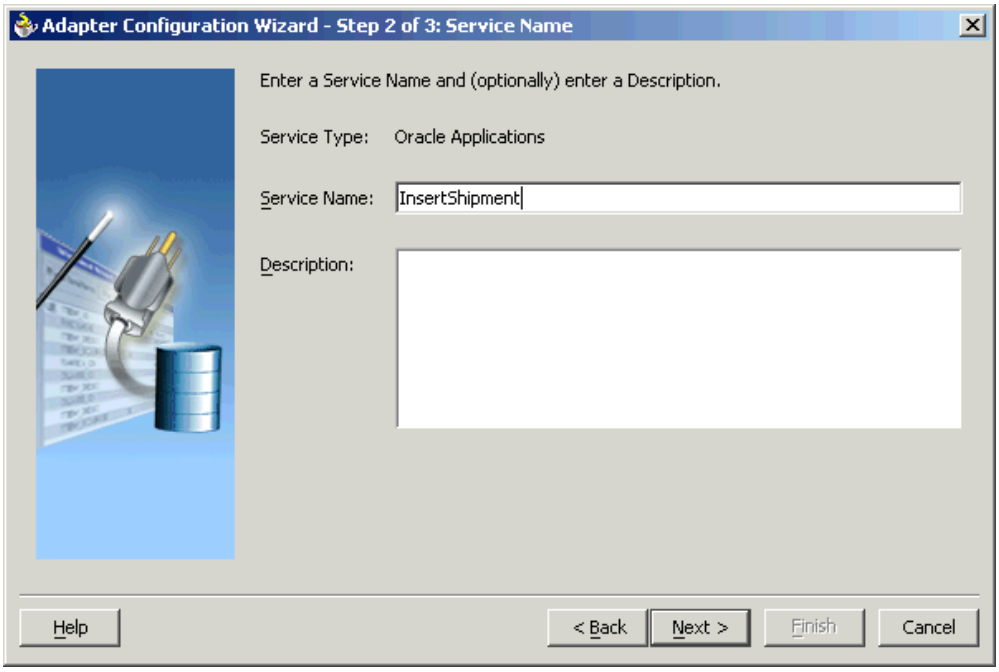
Selecting OracleAS Adapter for Oracle Applications

5. Click **Next**. The Service Name dialog box appears. Enter the following information:

   1. In the **Service Name** field, enter a service name.

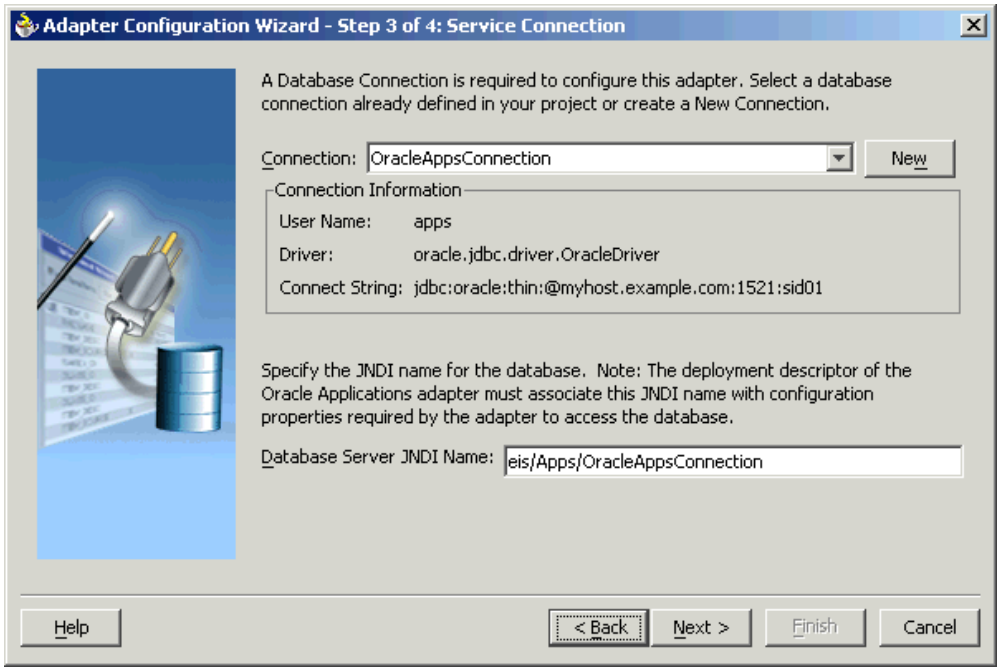
   2. In the **Description** field, enter a description for the service. This is an optional field.

Specifying the Service Name



6. Click **Next**. The Service Connection dialog box appears.

Creating a New Database Connection



7. Click **New** to define a database connection. The Create Database Connection Wizard appears. Alternatively, you can select an existing database connection from the Connection list.

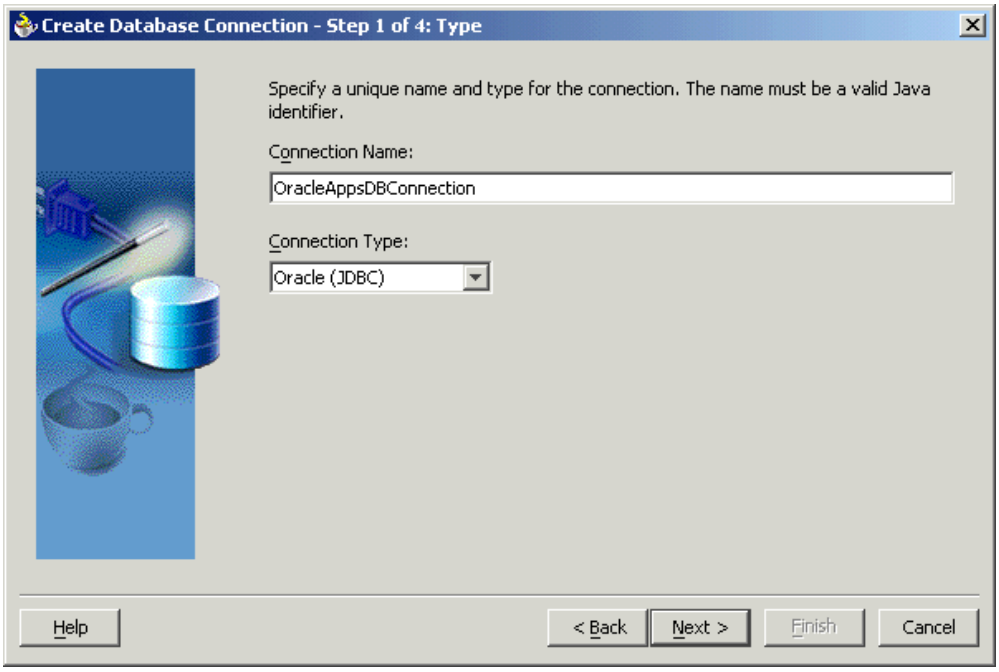
   Note: You need to connect to the database where Oracle Applications is running.

8. Enter the following information in the Type dialog box:

   1. In the **Connection Name** field, specify a unique name for the database connection.

   2. From the **Connection Type** box, select the type of connection for your database connection.

Specifying the Connection Name and Type of Connection



9. Click **Next**. The Authentication dialog box appears.

10. Enter information in the following fields:

    1. In the **UserName** field, specify a unique name for the database connection.

    2. In the **Password** field, specify a password for the database connection.

11. Click **Next**. The Connection dialog box appears.

12. Enter information in the following fields:

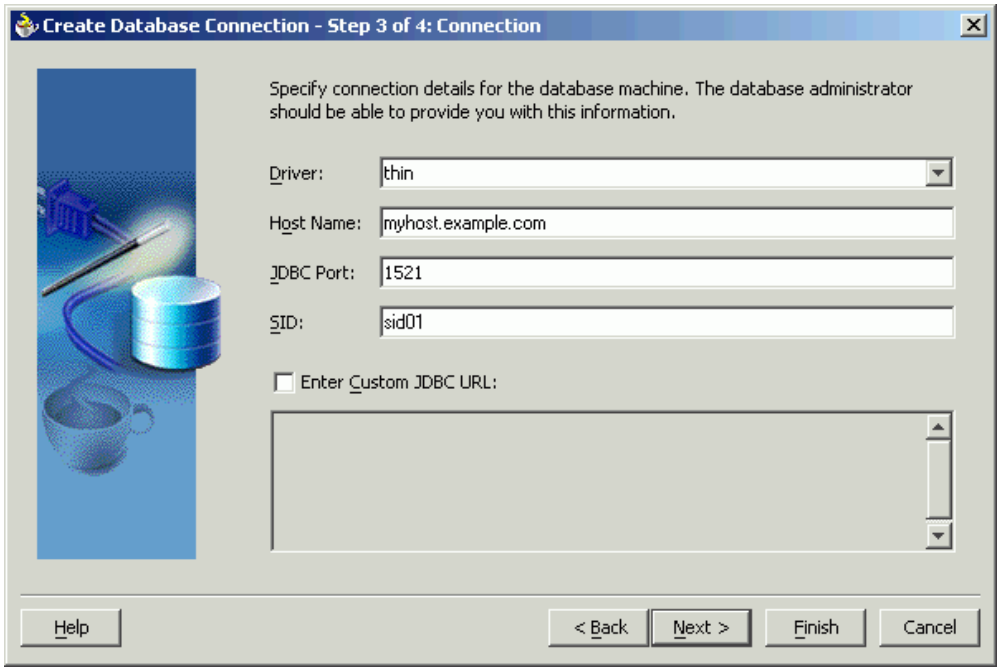
    1. From the **Driver** list, select **Thin**.

    2. In the **Host Name** field, specify the host name for the database connection.

    3. In the **JDBC Port** field, specify the port number for the database connection.

    4. In the **SID** field, specify a unique SID value for the database connection.

Specifying the New Database Connection Information



13. Click **Next**. The Test dialog appears.

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

15. Click **Next**. The Service Connection dialog box appears, providing a summary of the database connection.

16. The JNDI (Java Naming and Directory Interface) name corresponding to the database connection appears automatically in the **Database Server JNDI Name** field. Alternatively, you can specify a different JNDI name.

    Note: When you specify a JNDI name, the deployment descriptor of the Oracle Applications adapter must associate this JNDI name with configuration properties required by the adapter to access the database.

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 for production.

Note: For more information about JNDI concepts, see *Oracle*

Application Server Adapter Concepts.

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

    Once you have completed creating a new connection for the service, you can select an e-Commerce Gateway interface by browsing through the modules available in Oracle Applications.

18. Click **Next** in the Service Connection dialog box.

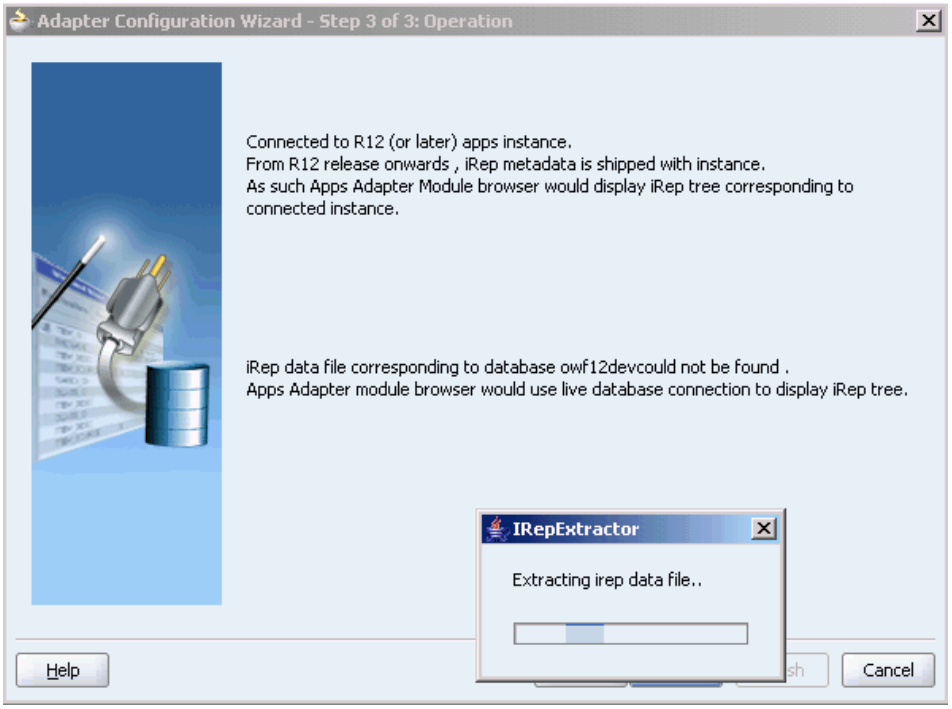
    For Oracle E-Business Suite Release 12:

    If you are connecting to Oracle E-Business Suite Release 12, then the **IREP File not present** dialog box appears indicating that OracleAS Adapter could not find the Oracle Integration Repository data file corresponding to the database you are connecting to Oracle Applications in your workspace. Absence of the data file would make browsing or searching of Integration Repository tree considerably slow. You can choose to extract the data file and create a local copy of the Integration Repository data file. Once it is created successfully, OracleAS Adapter will pick it up automatically next time and retrieve data from your local Integration Repository.

    You can select one of the following options:

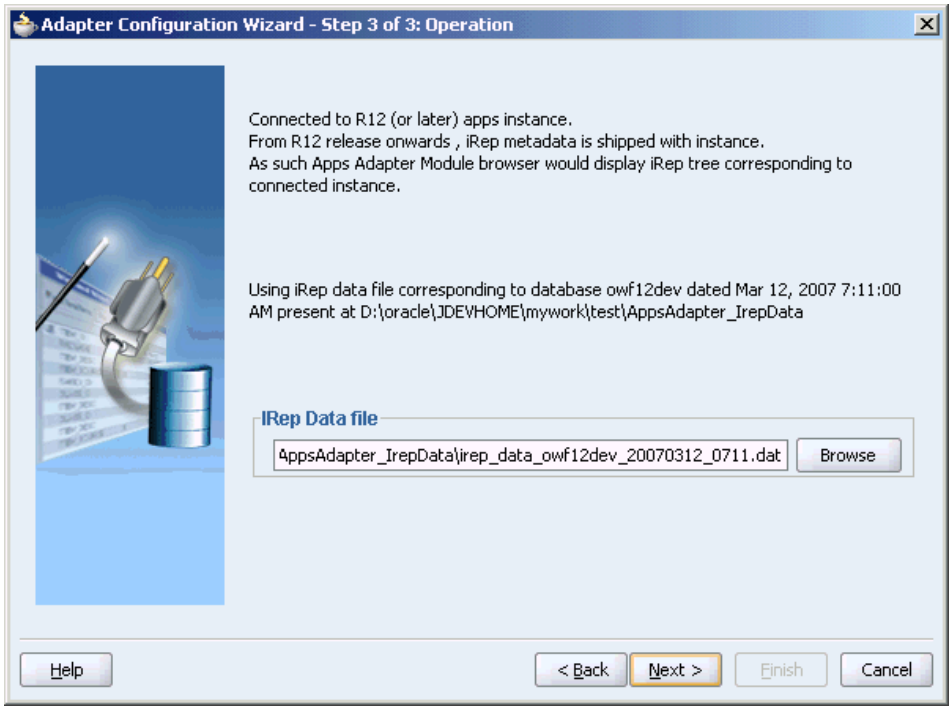
    - Click **Yes** to extract the Integration Repository data file.

Extracting Integration Repository Data File



After the system successfully creates a local copy of the Integration Repository data file, next time when you connect to the database, you will find the **IRep Data File** field appears in the Operation dialog box indicating where your local copy exists with the creation date and time as part of the file name.

Using the Local Integration Repository Data File



- Click **No** to query the Integration Repository data file from the live database you are connecting to display the Integration Repository tree.

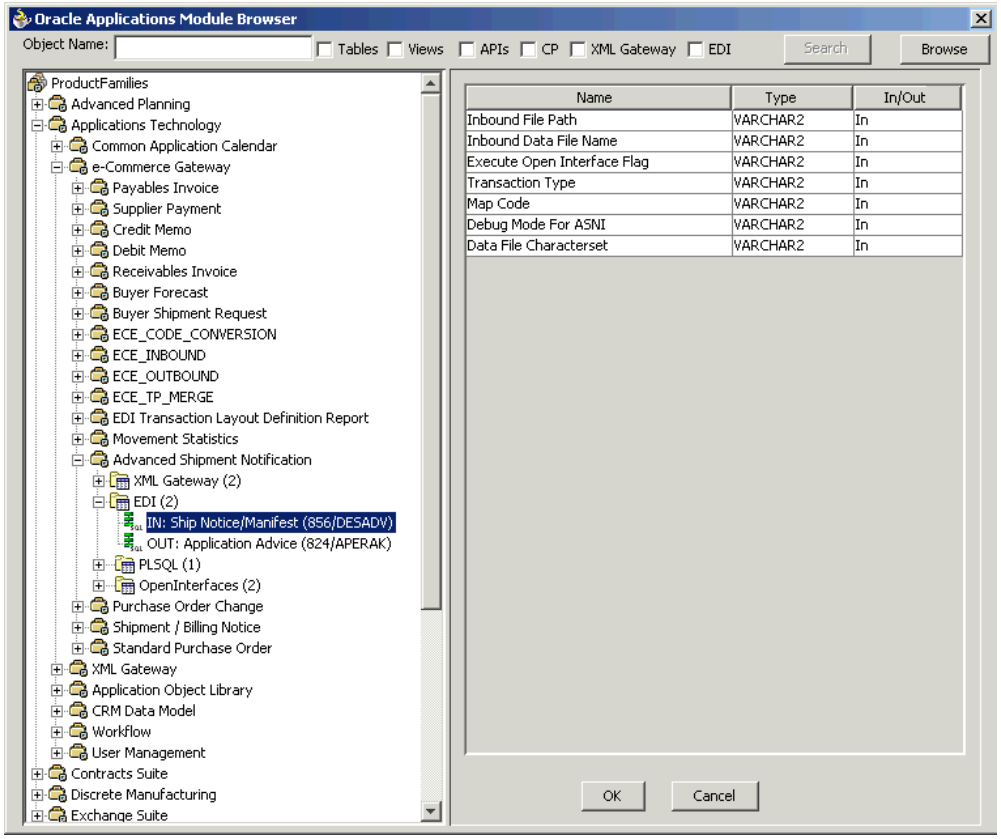
  Note: It is highly recommended that you create a local copy of the Integration Repository data file so that OracleAS Adapter will query the data next time from the local copy in your workspace to enhance the performance.

For Oracle E-Business Suite pre-Release 11.5.10:

If you are connecting to a pre-11.5.10 Oracle Applications instance, you must select the interface type in the Adapter Configuration Wizard. Select **Inbound into Oracle Applications** or **Outbound from Oracle Applications** depending on whether data is inbound into Oracle Applications or outbound from Oracle Applications. Next, click **Get CP** to choose the EDI concurrent program from the Oracle Applications Module Browser. Select **EDI Gateway** to proceed.

19. Click **Get Object** to open the Oracle Applications Module Browser.

Specifying the EDI Concurrent Program



20. Oracle Applications Module Browser includes the various product families that are available in Oracle Applications. For example, Applications Technology or Order Management Suite are product families in Oracle Applications. The product families contain the individual products. For example, Applications Technology contains the e-Commerce Gateway product. The product contains the business entities associated with the product. For example, the e-Commerce Gateway product contains the Advanced Shipment Notification business entity.

    Business entities contain the various application modules that are exposed for integration. These modules are grouped according to the interface they provide. EDI programs can be found under the EDI category.

21. Select an inbound or outbound EDI program. Click **OK**. You can select only one EDI program for each adapter service.

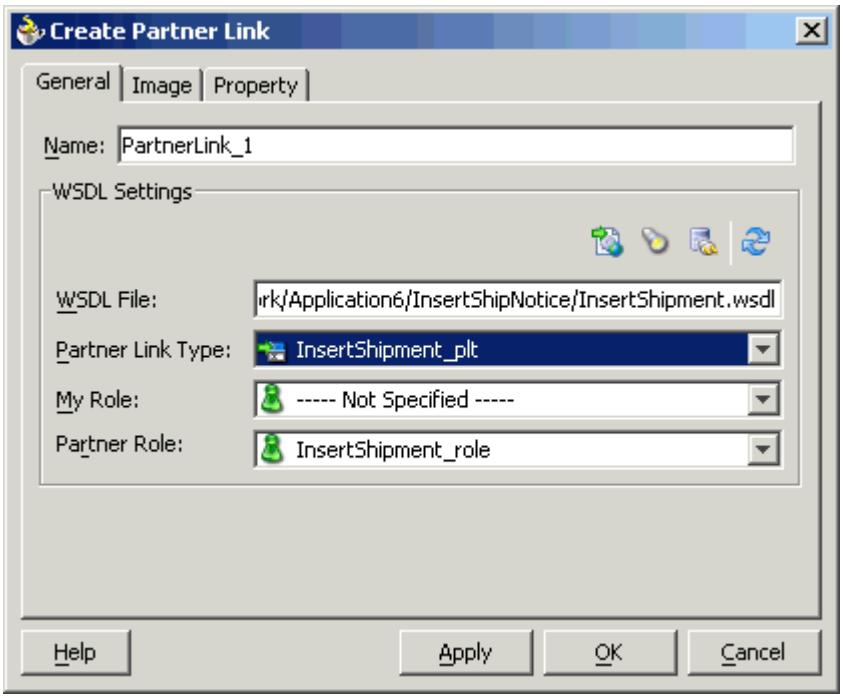
    > **Note:** You can also search for an EDI program by entering the name of the program in the **Object Name** field. Select the **EDI** check box and click **Search**.

22. The EDI program is added to Operation Objects. Click **Next** in the Operation dialog box.

23. Click **Finish**. When you click **Finish**, the wizard generates the WSDL file corresponding to the selected interface. This WSDL file is now available for the partner link.

    Note: When you click **Finish,** two SQL files may be added to the project if a wrapper does not exist for the function. A wrapper is generated the very first time you create the e-Commerce Gateway based service. Subsequent services reuse the same wrapper.

Completing the Partner Link Configuration



24. Click **OK**. The partner link is created with the required WSDL settings.

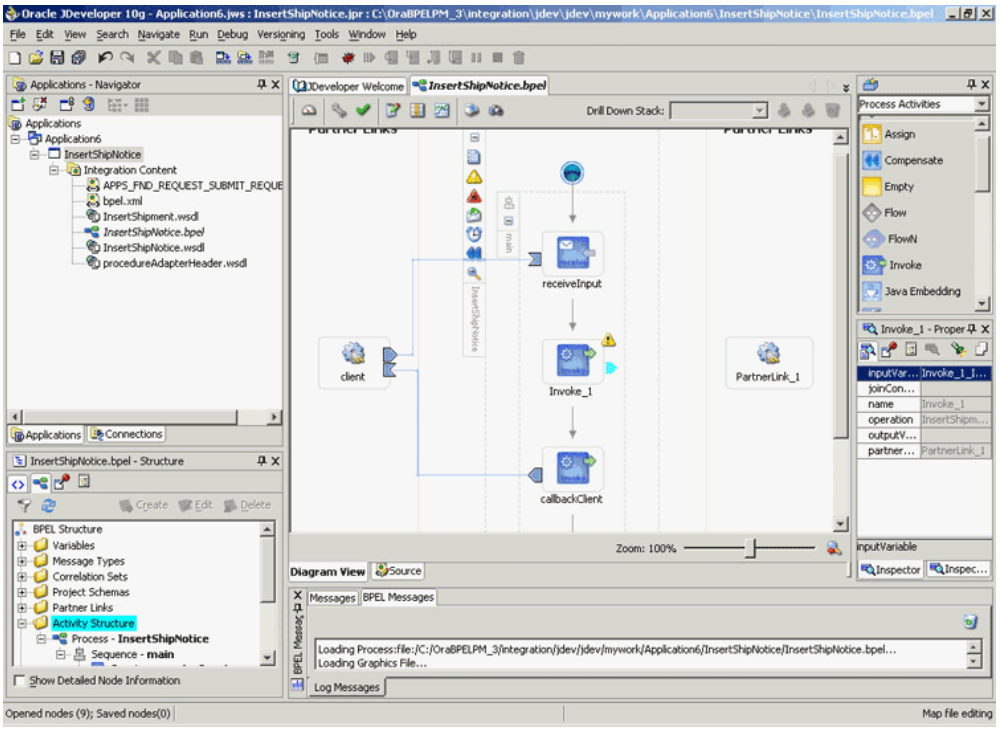
Configuring the Invoke Activity

After adding and configuring the partner link, the next task is to configure the BPEL process itself. You can start by configuring the **Invoke** process activity to invoke the EDI concurrent program.

To configure the Invoke activity:

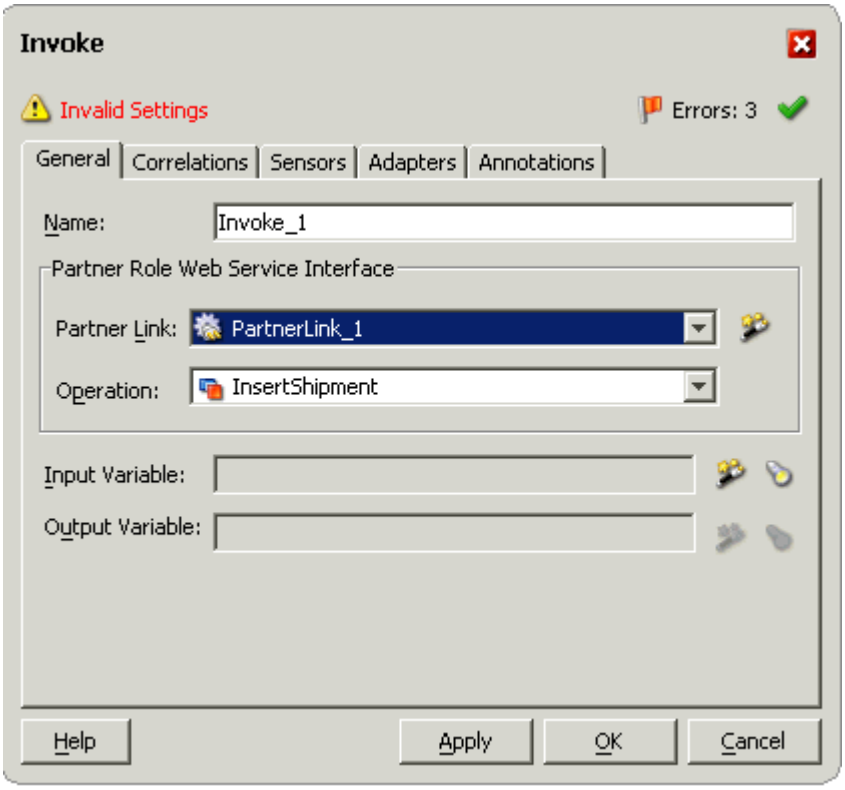
1. Drag and drop **Invoke** into the process map window. shows the **Invoke** activity after it has been added to the process map.

Adding the Invoke Activity



2. Double-click **Invoke** in the process map to open the Invoke dialog box. The **General** tab is selected by default.

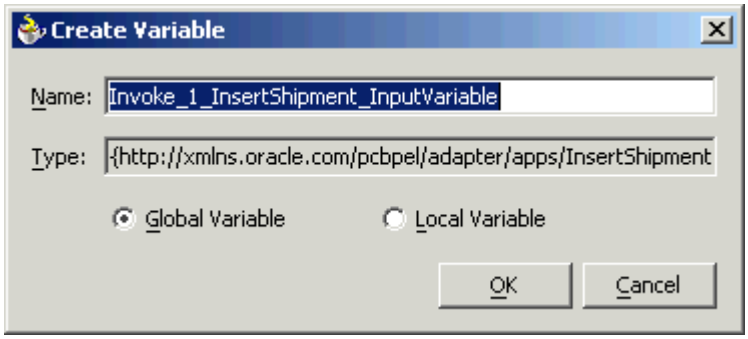
Configuring the Invoke Activity



3. In the **Partner Link** box, select the partner link to invoke. This is the partner link that you configured in the previous section. The **Operation** is automatically selected, depending on the EDI concurrent program that you chose when configuring the partner link.

4. Click the **Create** icon next to the **Input Variable** field. Enter a descriptive name for the variable in the Create Variable dialog box that appears. You can also accept the default name. Click **OK**.

Creating the Input Variable



5. Click the **Create** icon next to the **Output Variable** field. Enter a descriptive name for the variable in the Create Variable dialog box that appears. You can also accept the default name. Click **OK**.

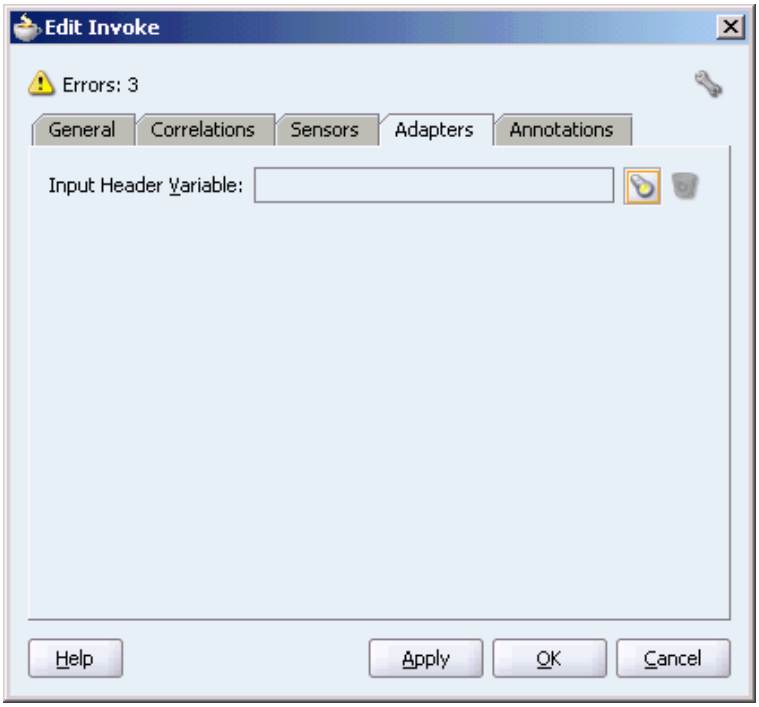
6. In the Invoke dialog box, click **Apply,** then click **OK.**

   Note: You can define an **Input Header Variable** on the **Adapters** tab of the Invoke dialog box. This variable can be used to provide context information for Oracle Applications in supporting multiple languages and multiple organization setups.

   To Create the Header Variable:

   1. Click on Adapters tab in the Edit Invoke dialog box and click **Browse Variable...** icon for the Input Header Variable field.

Specifying Input Header Variable

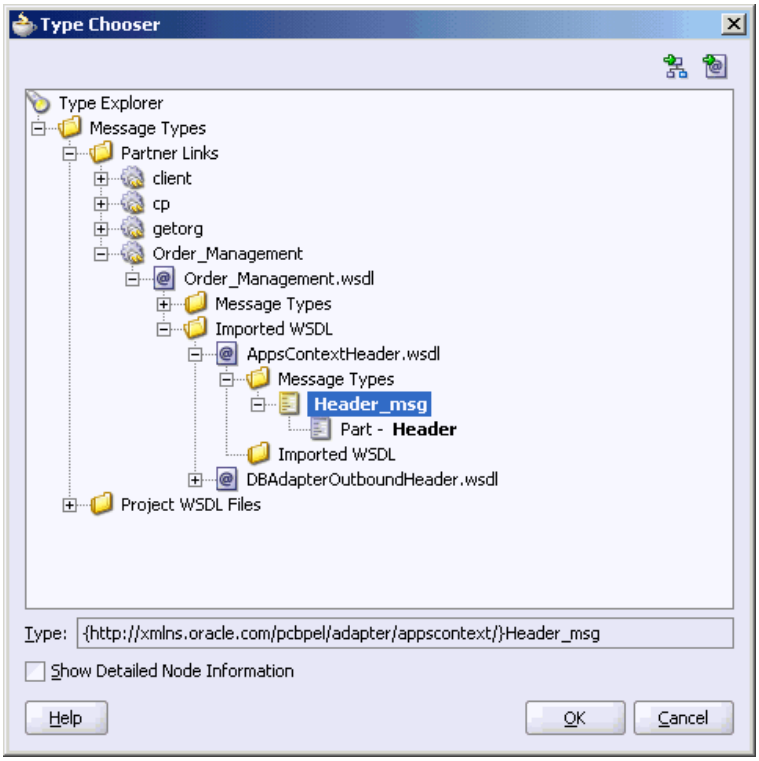


2. In the Variable Chooser dialog box, right-click on the Variables node and select Create Variable option from the menu.

3. Select **Message Type** and click **Browse Message Type...** icon to open Type Chooser dialog box.

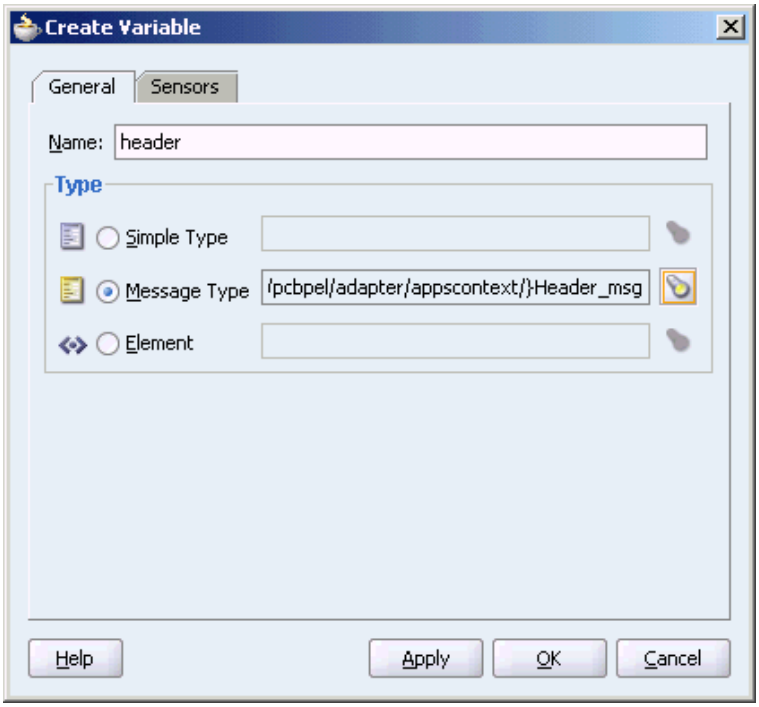
4. Expand the Partner Link node and locate the Header_msg node `{http://xmlns.oracle.com/pcbpel/adapter/appscontext/}Header_msg` for your partner link. The Header_msg node should be under the path, Partner Links node >*Your Partner Link WSDL* > Imported WSDL > AppsContextHeader.wsdl > Message Types > Header_msg.

Declaring Header Variable



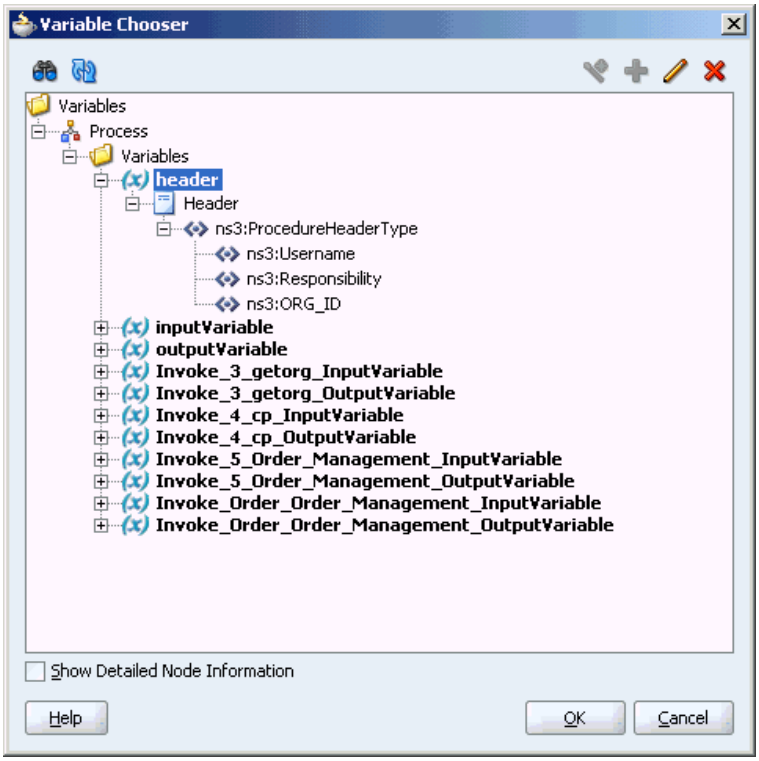
5. Click **OK** to return to the Create Variable dialog box with your selected message type populated.

Populating Selected Header Variable



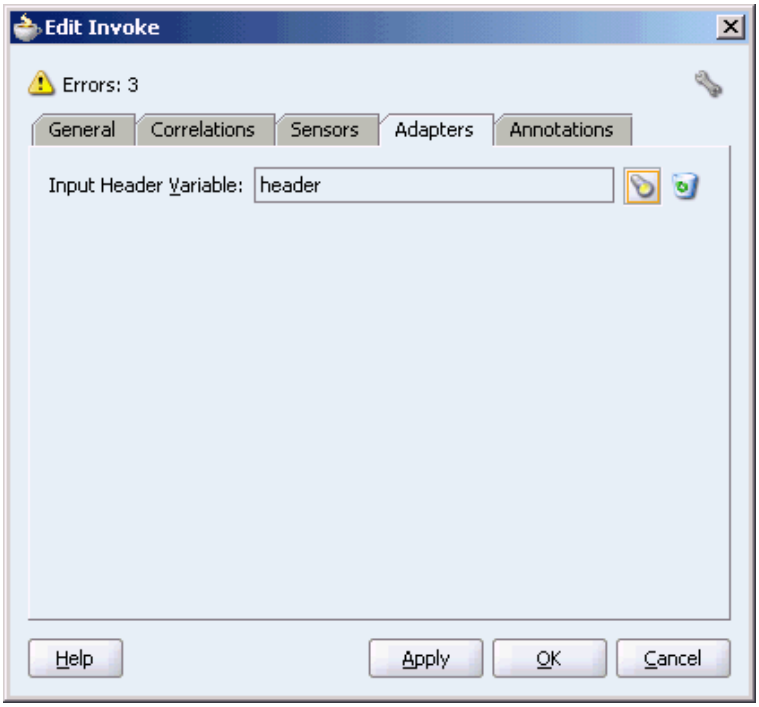
6. To view your header variable with Username, Responsibility, and ORG_ID, click **Browse Message Type...** icon to open Variable Chooser dialog box. Locate the header variable to view the variable hierarchical structure with these three parameters needed for applications context.

Viewing Header Variable Structure



7. Click **OK** to return to the Edit Invoke dialog with the selected header variable populated for the Input Header Variable field. Click **Apply** to complete the header creation.

Populating Input Header Variable



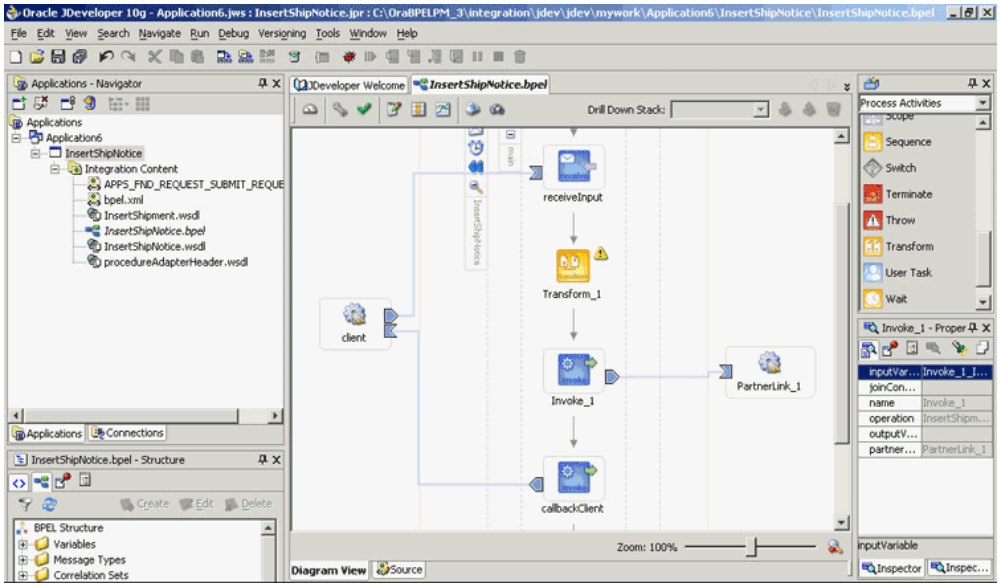
Configuring the Transform Activity

The Transform activity can be used to configure the parameters for the input and output variable. The Transform activity can also be used if variable values need to be transformed before updating them in Oracle Applications.

To configure the Transform Activity:

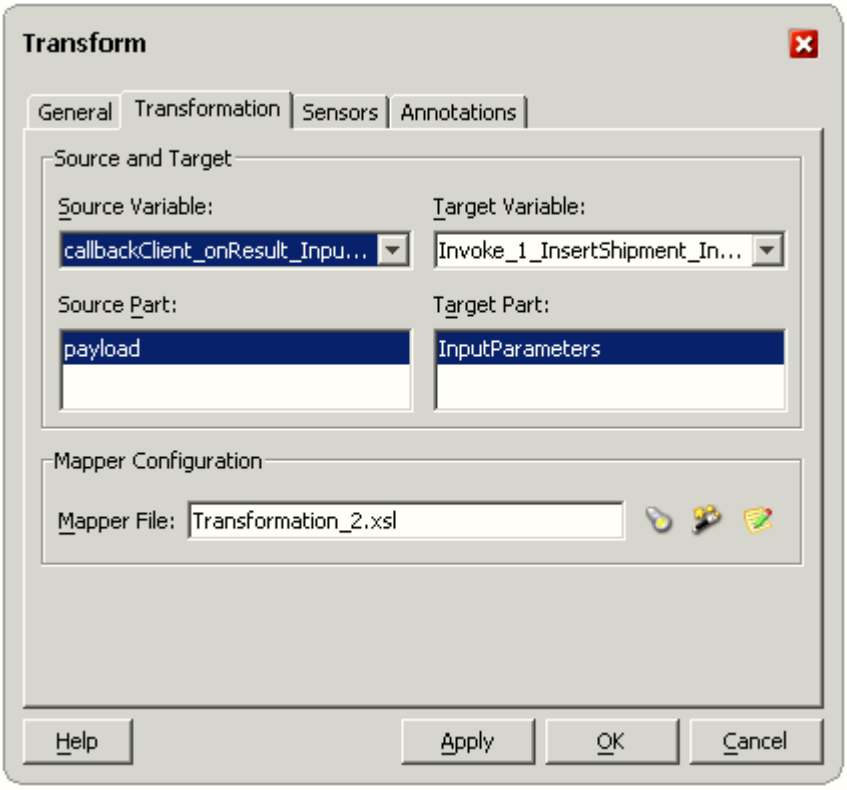
1. Drag and drop **Transform** into the process map window between the **Receive** and **Invoke** activities.

Adding the Transform Activity



2. Double-click **Transform** in the process map to open the Transform dialog box. The **Transformation** tab is selected by default.

Configuring the Transform Activity



3. Select the **Source Variable** and **Target Variable** from the respective boxes. Elements are mapped from the **Source Variable** to the **Target Variable**.

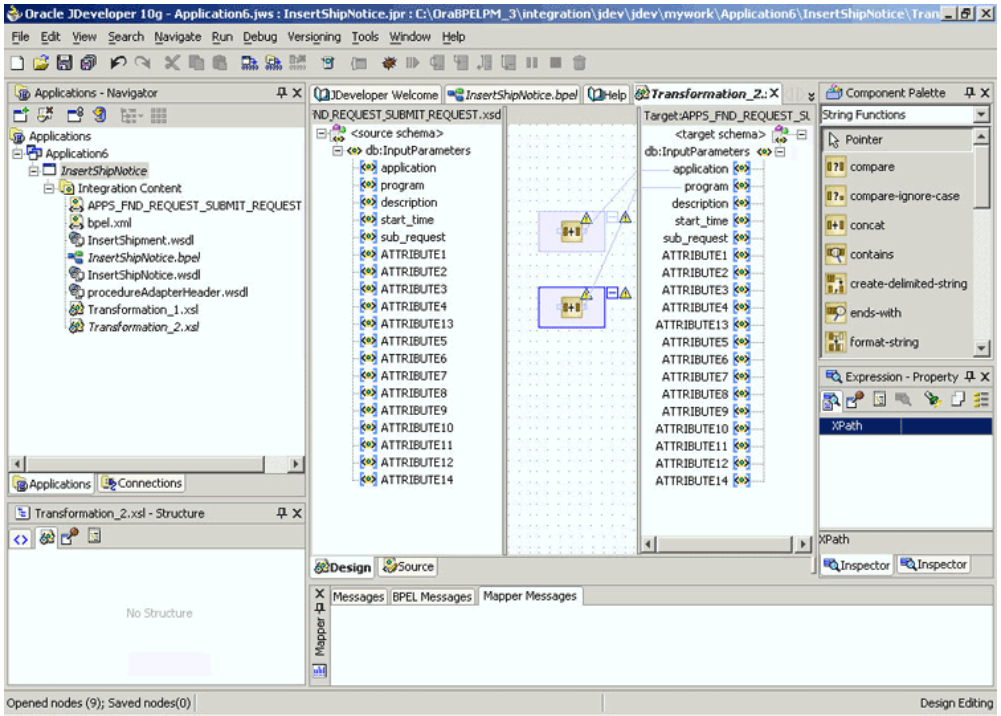
4. Select the **Source Part** of the variable from which to map elements. For example, the source part may be a payload schema consisting of a purchase order request.

5. Select the **Target Part** of the variable to which to map elements. For example, the target part may be a payload schema consisting of a purchase order acknowledgment.

6. **Mapper File** specifies the file in which you create the mappings using the XSLT Mapper Transformation tool. Click the **Create** icon next to the **Mapper File** field to create a new transformation mapping file.

7. The transformation mapping file appears. Its **Design** view is displayed by default.

Editing the Transformation Mapping File

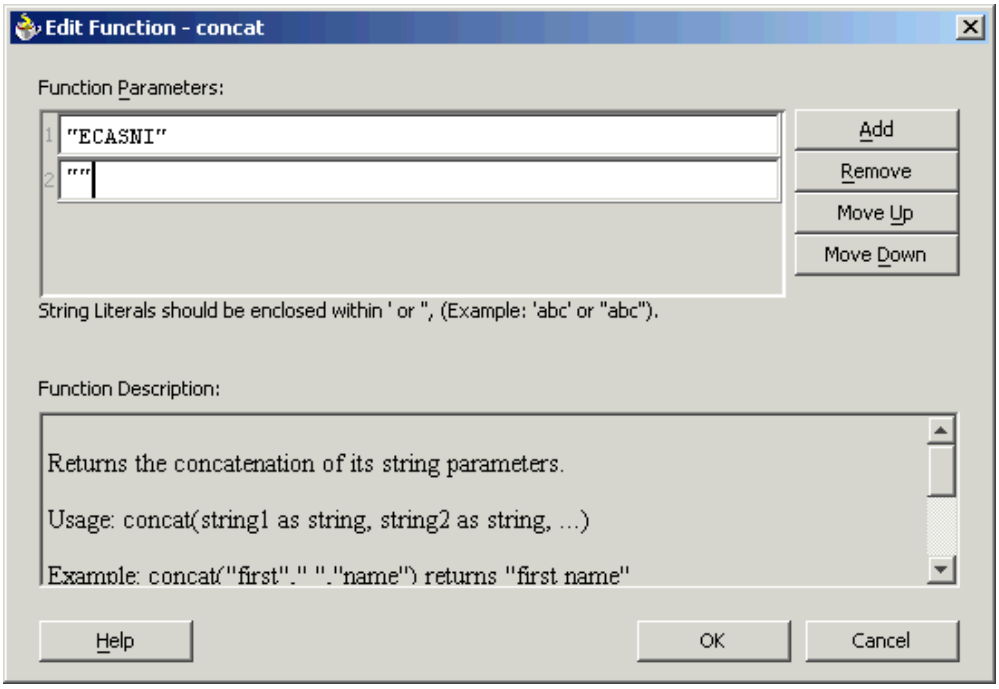


8. You can define the parameter values in the **Design** view. Drag a string function to the **Design** area. Connect the function to the appropriate parameter for which you want to define a value.

   Note: You can use an input parameter value from the source variable, transform it using a string function, and use it as the input parameter value for the target variable.

9. Double-click the icon for the function. The Edit Function dialog box appears.

Supplying the Function Parameters



10. Repeat steps 8 and 9 for all the parameters that you need to supply.

    Note: You can use both the Transform and Assign activities to configure the variables. If the number of mappings to be configured is less, say three or four, then you can use the Assign activity. The Transform activity is more suitable for situations where the number of mappings is large or where variable transformation is required.

Run-Time Tasks for e-Commerce Gateway

After designing the BPEL process, the next step is to deploy, run and monitor it.

1. Deploy the BPEL process., page 7-28
2. Test the BPEL process., page 7-30
3. Verify records in Oracle Applications., page 7-32

Deploying the BPEL Process

You need to deploy the BPEL process before you can run it. The BPEL process is first

compiled and then deployed to the BPEL server.

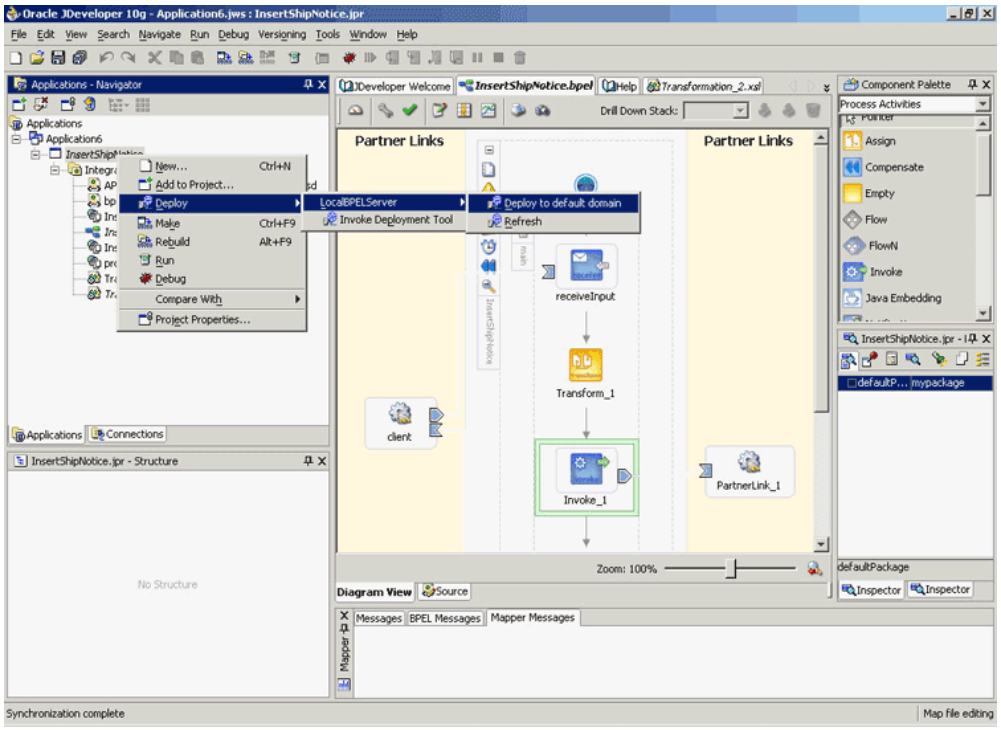
To deploy the BPEL process:

1. Select the BPEL project in the Applications window.

2. Right-click the project name, and then select **Deploy** from the menu that appears.

3. Select **Local BPEL Server** followed by **Deploy to Default Domain,** if you are deploying the process on the local BPEL server.

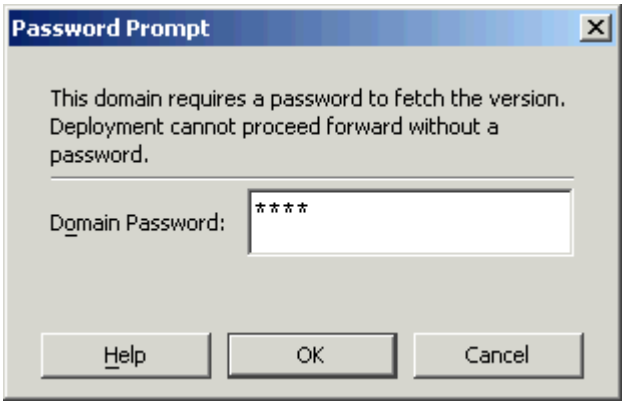
Deploying the BPEL Process



Note: You can select **Invoke Deployment Tool** if you want to deploy to a different BPEL server.

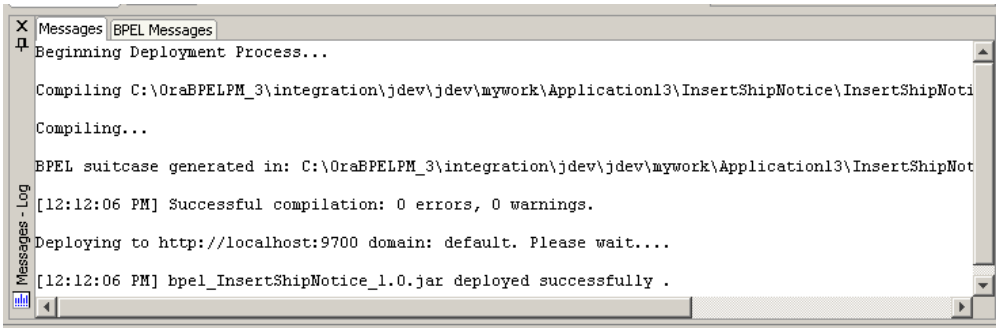
4. The Password Prompt dialog box appears. Enter the password for the default domain in the **Domain Password** field. Click **OK.**

Specifying the Domain Password



5. The BPEL process is compiled and deployed. You can check the progress in the Messages window.

Messages Window



Testing the BPEL Process

Once the BPEL process is deployed, it can be seen in the BPEL Console. You can manage and monitor the process from the BPEL Console. You can also test the process and the integration interface by manually initiating the process.

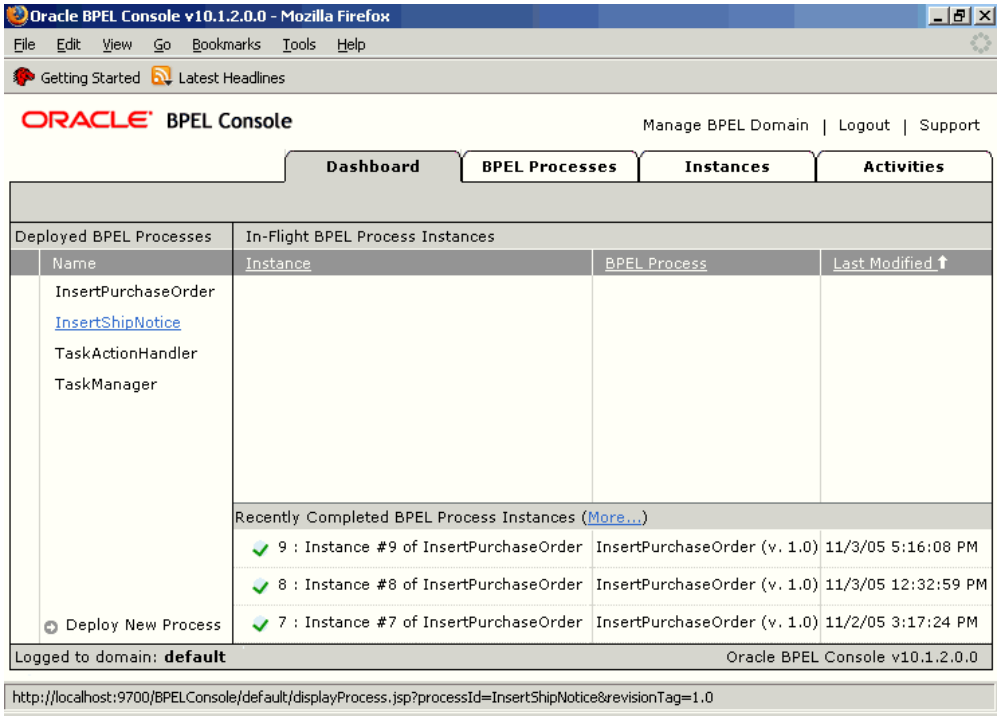
To test the BPEL process:

1. To open the BPEL Console, click **Start**, and choose **Programs**. In the Programs menu, select **Oracle -** *ORACLE_HOME*,**Oracle BPEL Process Manager 10.1.3**, and then select **BPEL Console**.

2. The BPEL Console login screen appears. Select **Default** in the **Domain** box. Enter the password for the default domain in the **Password** field. Click **Login.**

3. Oracle BPEL Console appears. The list of deployed processes is shown under Deployed BPEL Processes.

Deployed BPEL Processes

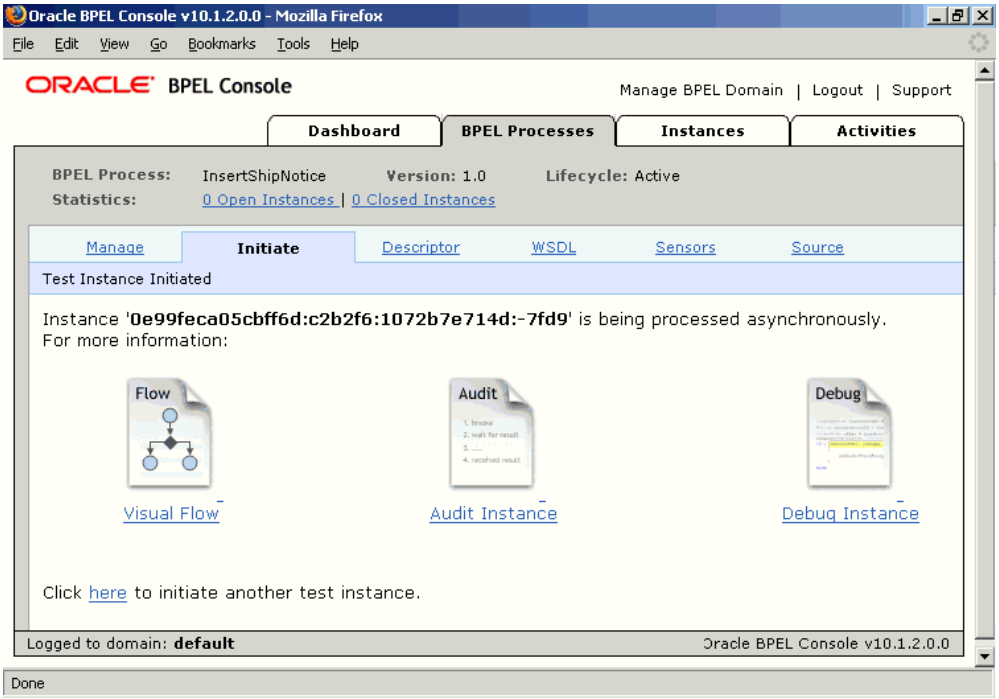


4. Click the BPEL process that you want to initiate. The Initiate page appears. Enter the input string required by the process.

5. Click **Post XML Message** to initiate the process.

6. The BPEL process is now initiated. You can check the process flow by clicking the **Visual Flow** icon.

BPEL Console Initiate Page



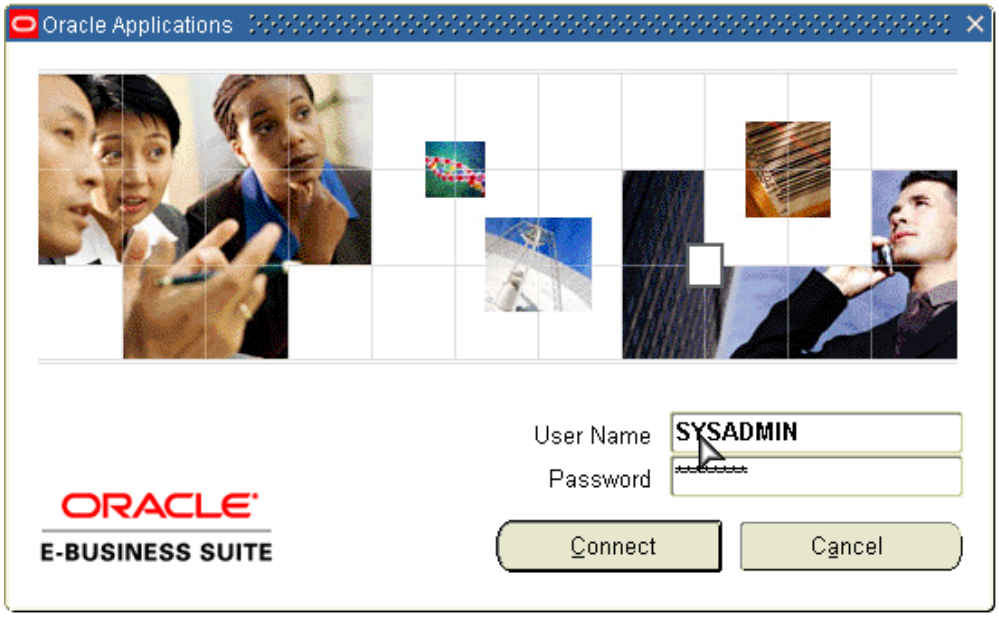
7. The audit trail provides information on the steps that have been executed. The audit trail also records the Request ID that is returned for the transaction. You can check the audit trail by clicking the **Audit Instance** icon.

Verifying Records in Oracle Applications

To verify records in Oracle Applications:

1. Log in to Oracle Applications as the System Administrator.

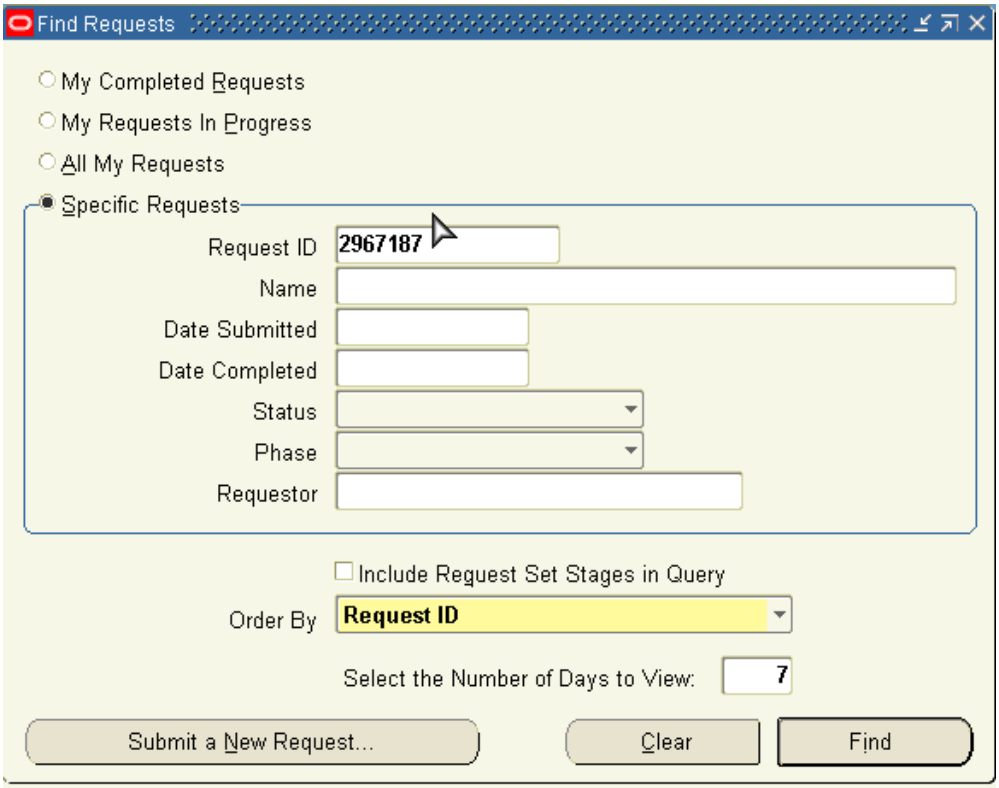
Oracle Applications Login Dialog Box



2. Select **Requests** from the **View** menu.

3. Search for the Request by entering the Request Id that you got from the audit trail, then click **Find.**

Find Requests Dialog Box



4. The Request details are displayed. You can check for details such as the Phase and Status of the request.

5. If the **Status** of the request is `Complete`, you can also query the appropriate table in Oracle Applications to search for the relevant records that have been inserted.

Querying Oracle Applications for a Record

```
SQL> select     distinct(shipment_num)  from    rcv_headers_interface   where  s
hipment_num     =       'S1722427';

SHIPMENT_NUM
------------------------------
S1722427

SQL>
```

A

Modifying Connection and Definition Files

This appendix covers the following topics:

- WSDL Definition File
- Configuring Connection Information
- Oracle E-Business Suite Connectivity Using J2EE Data Source

WSDL Definition File

Web Service Definition Language (WSDL) is generated by the JDeveloper BPEL Designer during design time. The WSDL file generated by the Adapter Wizard is the adapter service definition. In addition, it specifies various operations exposed by the service. The operations are based on user input to the Adapter Wizard. An operation either retrieves or inserts data to Oracle Applications and is represented by a JCA activation or interaction spec.

Example of a WSDL File

```
  <binding name="Ora_binding" type="tns:Ora_ptt">
  <jca:binding />
    <operation name="Ora">
    <jca:operation
      SchemaName="APPS"
      PackageName="XXBPEL_CUSTOMER"
      ProcedureName="CREATE_PERSON_PRIMITIVE"
      InteractionSpec="oracle.tip.adapter.apps.AppsStoredProcedureInteractionSpec" >
    </jca:operation>
        <input/>
    </operation>
  </binding>
  <service name="Ora">
    <port name="Ora_pt" binding="tns:Ora_binding">
<!--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
f eis/Apps/OracleApps is missing.
These mcf properties are safe to remove.-->
    <jca:address location="eis/Apps/OracleApps" UIConnectionName="OracleApps" UIOracleAppType="
DBOBJECT"
      ManagedConnectionFactory="oracle.tip.adapter.apps.AppsManagedConnectionFactory"
      mcf.ConnectionString="jdbc:oracle:thin:@myhost02.example.com:1521:sid02"
      mcf.UserName="apps"
      mcf.Password="53CB0F044A0D3DD2C063679F18F89870" />
    </port>
  </service>
 <plt:partnerLinkType name="Ora_plt" >
  <plt:role name="Ora_role" >
    <plt:portType name="tns:Ora_ptt" />
  </plt:role>
```

Configuring Connection Information

OracleAS Adapter for Oracle Applications is deployed as J2CA 1.5 resource adapters within the same OC4J container as BPEL Process Manager during installation.

Although OracleAS Adapter for Oracle Applications is physically deployed as J2CA 1.5 resource adapters, the logical deployment of the Adapter involves creating the connection entries for the J2CA 1.0 resource adapter. This connection information is for tying up the database connection information provided when you create a partner link or service and is used by iAS at run time in the background.

Use Enterprise Manager to modify the connection configuration so that the proper information is inserted into `oc4j-ra.xml` automatically.

> **Note:** The oc4j-ra.xml is the file that is administered through the Enterprise Manager console.

The following example of connection configuration for OracleAS Adapter for Oracle Applications can be used with the Oracle Applications adapter tutorials packaged with the product. For the logical deployment changes to take effect, the OC4J container process must be restarted.

To configure the connection information:

1. Log into Enterprise manager at the URL for your web server host machine:

   `http://servername:port/em`

   > **Note:** The default port number is `8888`. The default login user ID and password are `oc4jadmin` and `welcome1` respectively.

2. Create a connection pool and a data source:

   1. Click **Home**, then **Administration**.
   2. Click the **Go To Task** icon for Services/JDBC Resources.
   3. Under Connection Pools, click the **Create** button.
   4. Accept the defaults, and click **Continue**.
   5. Enter the following values (leave defaults for the rest):

Field	Value
Name	`appsSample_pool`
JDBC URL	The URL for your database. For example: `jdbc:oracle:thin:@host:1521:SID`
Username	`apps`
Password	`apps`

   6. Click **Finish**, then click the **Test Connection** icon for your new connection pool.
   7. On the page that appears, click **Test**.

      Back on the main page, a successful connection message is displayed. If you see an error message, check the URL and credentials to ensure you've entered the right information.
   8. Click **Finish**.

9. Under Data Sources, click **Create**.

10. Accept the defaults and click **Continue**.

11. Enter the following values (leave defaults for the rest):

Field	Value
Name	`appsDemoDS`
JNDI Location	`jdbc/appsSampleDataSource`
Connection Pool	`appsSample_pool`

12. Click **Finish**.

3. Create an Oracle Applications adapter connection:

   1. At the top of the page, click the **OC4J:home** breadcrumb link, then the **Applications** link.

   2. In the tree of applications, click the **default** link.

   3. Under Modules, click the **AppsAdapter** link, then the **Connection Factories** link.

   4. Under Connection Factories, click **Create**.

      Note: Use the **Create** button near the top of the screen, not the one in the Shared Connection Pools section.

   5. Accept all the defaults and click **Continue**.

   6. For **JNDI Location**, enter `eis/Apps/appsSample`.

   7. Under Configuration Properties, for **xADataSourceName**, enter `jdbc/appsSampleDataSource`. Keep the default entries for all of the other fields.

   8. Click **Finish**.

Oracle E-Business Suite Connectivity Using J2EE Data Source

OracleAS Adapter for Oracle Applications uses a new mechanism to authenticate users at run time and get the connection to Oracle Applications databases through the use of J2EE data sources. Since J2EE data sources are defined in OC4J container that runs the BPEL processes, this approach is native to Oracle E-Business Suite in defining the connection pool to access the application database.

With this new mechanism, account details information including application login user name and password that was required as part of the configuration for database connection is now added together with the dbc file location as input parameters during the J2EE data source creation.

To accomplish this process, the following steps are used to define J2EE data source connection to the Oracle E-Business Suite database:

1. Register your Service-Oriented Architecture (SOA) suite middle tier node on the E-Business Suite environment and generate the dbc file used by the data source implementation to instantiate the connections.

2. Copy the dbc file to the middle tier server where your SOA suite server runs, and place it on a location in the file system to which the SOA suite owner has access.

3. Create a connection pool where you need to enter the application login user name, password, and dbc file location as the connection factory properties.

4. Create an application data source. This is the step that you associate the application data source with the Java Naming and Directory Interface (JNDI) name for the application database connection for OracleAS Adapter for Oracle Applications.

> **Note:** To have this feature available, you must apply necessary patches to enable the connectivity between Oracle E-Business Suite and an external Oracle Application Server at run time for Oracle E-Business Suite Release 12 and Release 11*i*10. See "Oracle Application Sever Adapter for Oracle Applications Documentation Update, Release 10*g*" Oracle*MetaLink* Document 464164.1 for details.

Index

A

agent
  See business events concepts
APIs
  See PL/SQL APIs
applications context
  example, 1-12
  overview, 1-11
Applications Context
  Multiple Language Support, 1-20
  multiple organization setups, 1-17

B

B2B, 2-2
base tables, 5-2
business events, 3-1
  debugging, 3-53
  troubleshooting, 3-52
business events concepts, 3-3
  agent, 3-4
  deferred subscription processing, 3-3
  event, 3-3
  event activity, 3-3
  event data, 3-3
  event key, 3-3
  event message, 3-3
  event subscription, 3-3
business events outbound
  Assign activity, 3-33
  configuring OracleAS Adapter for Oracle Applications, 3-4
  creating a new BPEL project, 3-5
  creating a partner link, 3-7
  deploying the BPEL process, 3-36
  design-time prerequisites, 3-4
  design-time tasks, 3-4
  file adapter partner link, 3-25
  Invoke activity, 3-30
  Receive activity, 3-22
  run-time tasks, 3-36
  testing the BPEL process, 3-38

C

concurrent programs
  adding a partner link, 4-5
  configuring OracleAS Adapter for Oracle Applications, 4-1
  configuring the Invoke activity, 4-15
  configuring the Transform activity, 4-23
  creating a new BPEL project, 4-2
  deploying the BPEL process, 4-28
  design-time steps, 4-1
  overview, 4-1
  prerequisite to configure, 4-2
  run-time steps, 4-27
  testing the BPEL process, 4-29
  verifying records, 4-31
connection information, A-2

D

deferred subscription processing, 3-3
  See also business events concepts
design-time tasks

business events outbound, 3-4

E

EDI
adding a partner link, 7-6
configuring OracleAS Adapter for Oracle Applications, 7-2
configuring the Invoke activity, 7-16
configuring the Transform activity, 7-24
creating a new BPEL project, 7-3
deploying the BPEL process, 7-28
design-time steps, 7-2
overview, 7-1
prerequisites to configuring OracleAS Adapter for Oracle Applications, 7-2
run-time steps, 7-28
testing the BPEL process, 7-30
verifying records in Oracle Applications, 7-32
event
See business events concepts
event activity, 3-3
event data, 3-3
event key, 3-3
event message, 3-3
event subscription, 3-3
example WSDL file, A-1

I

integration architecture, 2-2
interface tables, 5-2
adding a partner link, 5-6
configuring OracleAS Adapter for Oracle Applications, 5-3
configuring the Assign activity, 5-22
configuring the Invoke activity, 5-19
creating a new BPEL project, 5-3
debugging, 5-53
deploying the BPEL process, 5-23
design-time steps, 5-3
overview, 5-2
prerequisites to configure, 5-3
run-time steps, 5-23
testing the BPEL process, 5-25
troubleshooting, 5-53

M

message queues, 2-3
inbound queues, 2-3
outbound queues, 2-3
Module Browser, 1-31

N

new features, 1-10

O

oc4j-ra.xml file
configuring, A-2
Oracle Applications Module Browser, 1-31
OracleAS Adapter for Oracle Applications
architecture, 1-3
features, 1-2
Function security support, 1-24
installing, 1-30
interfaces, 1-4
issues and workarounds, 1-32
new features, 1-10
Oracle E-Business Suite Support, 1-6
Oracle Integration Repository, 1-6
overview, 1-1

P

PL/SQL APIs
adding a new partner link, 6-5
configuring OracleAS Adapter for Oracle Applications, 6-2
configuring the Invoke activity, 6-23
configuring the Transform activity, 6-30
creating a new BPEL project, 6-2
debugging, 6-38
deploying the BPEL process, 6-33
design-time steps, 6-2
overview, 6-1
prerequisite to configure, 6-2
run-time steps, 6-33
testing the BPEL process, 6-35
troubleshooting, 6-37
prerequisites
business events outbound, 3-4

R

run-time tasks
business events outbound, 3-36

S

standards-based messaging, 2-2

V

views, 5-2
adding a partner link, 5-31
configuring Oracles Adapter for Oracle Applications, 5-27
configuring the Assign activity, 5-48
configuring the Invoke activity, 5-45
creating a new BPEL project, 5-28
deploying the BPEL process, 5-49
design-time steps, 5-27
overview, 5-2
prerequisites to configure, 5-27
run-time steps, 5-49
testing the BPEL process, 5-51

W

wrapper APIs, 6-17
DEFAULT clause handling in wrapper procedures, 6-22
describing parameters with DEFAULT clause, 6-20
WSDL, A-1

X

XML Gateway
debugging, 2-49
overview, 2-1
troubleshooting, 2-48
XML gateway envelope, 2-4
ATTRIBUTE3, 2-5
DOCUMENT_NUMBER, 2-4
MESSAGE_STANDARD, 2-4
MESSAGE_TYPE, 2-4
parameters defined by the application, 2-5
parameters not used, 2-5
PARTY_SITE_ID, 2-5
PASSWORD, 2-5
PAYLOAD, 2-5
PROTOCOL_ADDRESS, 2-4
PROTOCOL_TYPE, 2-4
TRANSACTION_SUBTYPE, 2-4
TRANSACTION_TYPE, 2-4
USERNAME, 2-5
XML gateway inbound
adding a partner link for the file adapter, 2-27
configuring OracleAS Adapter for Oracle Applications, 2-6
configuring the Assign activity, 2-36
configuring the Invoke activity, 2-21
configuring the Receive activity, 2-33
creating a new BPEL project, 2-7
creating a partner link, 2-10
deploying the BPEL process, 2-39
design-time steps, 2-6
prerequisites to configure, 2-6
run-time steps, 2-39
testing the BPEL process, 2-41
using transaction monitor, 2-44
verifying records, 2-44
XML gateway outbound
configuring OracleAS Adapter for Oracle Applications, 2-46
design-time steps, 2-45
prerequisites to configure, 2-45
run-time steps, 2-47
testing the BPEL process, 2-47