

**Oracle® E-Business Suite**

Integrated SOA Gateway Developer's Guide

Release 12.1

**Part No. E12065-06**

June 2010

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 Oracle E-Business Suite Integrated SOA Gateway Overview**

Oracle E-Business Suite Integrated SOA Gateway Overview.....	1-1
Major Components Features and Definitions.....	1-3

## **2 Discovering and Viewing Integration Interfaces**

Overview.....	2-1
Searching and Viewing Integration Interfaces.....	2-1
Reviewing Interface Details.....	2-6
Reviewing WSDL Element Details.....	2-10
Understanding SOAP Messages.....	2-19

## **3 Using PL/SQL APIs as Web Services**

Overview.....	3-1
Using PL/SQL WSDLs at Design Time.....	3-1
Creating a New BPEL Project.....	3-6
Creating a Partner Link for the Web Service.....	3-8
Adding a Partner Link for File Adapter.....	3-12
Adding Invoke Activities.....	3-21
Adding Assign Activities.....	3-25
Deploying and Testing the BPEL Process.....	3-37
Deploying the BPEL Process.....	3-38
Testing the BPEL Process.....	3-39

## 4 Using XML Gateway Inbound and Outbound Interfaces

Overview.....	4-1
Using XML Gateway Inbound Services.....	4-2
Using XML Gateway Inbound Services at Design Time.....	4-2
Creating a New BPEL Project.....	4-7
Creating a Partner Link.....	4-9
Adding Partner Links for File Adapter.....	4-12
Adding Invoke Activities.....	4-18
Adding Assign Activities.....	4-21
Deploying and Testing the BPEL Process at Run Time.....	4-26
Deploying the BPEL Process.....	4-27
Testing the BPEL Process.....	4-28
Using XML Gateway Outbound Through Subscription Model.....	4-30
Using XML Gateway Outbound Services at Design Time.....	4-30
Creating a New BPEL Project.....	4-32
Creating a Partner Link for AQ Adapter.....	4-33
Adding a Receive Activity.....	4-39
Adding a Partner Link for File Adapter.....	4-41
Adding an Invoke Activity.....	4-47
Adding an Assign Activity.....	4-49
Deploying and Testing the BPEL Process at Run Time.....	4-51
Deploying the BPEL Process.....	4-52
Testing the BPEL Process.....	4-53

## 5 Using Business Events Through Subscription Model

Overview.....	5-1
Using a Business Event in Creating a BPEL Process at Design Time.....	5-1
Creating a New BPEL Project.....	5-5
Creating a Partner Link for AQ Adapter.....	5-6
Adding a Receive Activity.....	5-13
Adding a Partner Link for File Adapter.....	5-15
Adding an Invoke Activity.....	5-21
Adding an Assign Activity.....	5-23
Deploying and Testing the BPEL Process at Run Time.....	5-25
Deploying the BPEL Process.....	5-26
Testing the BPEL Process.....	5-27

## 6 Using Concurrent Programs

Overview.....	6-1
---------------	-----



<b>Using Concurrent Program WSDLs at Design Time.....</b>	<b>6-1</b>
Creating a New BPEL Project.....	6-5
Creating a Partner Link for the Web Service.....	6-7
Adding a Partner Link for File Adapter.....	6-10
Adding Invoke Activities.....	6-15
Adding Assign Activities.....	6-19
<b>Deploying and Testing the BPEL Process at Run Time.....</b>	<b>6-23</b>
Deploying the BPEL Process.....	6-24
Testing the BPEL Process.....	6-25

## **7 Using Business Service Objects**

<b>Overview.....</b>	<b>7-1</b>
<b>Using Business Service Object WSDLs at Design Time.....</b>	<b>7-1</b>
Creating a New BPEL Project.....	7-5
Creating a Partner Link.....	7-7
Adding a Partner Link for File Adapter.....	7-9
Adding an Invoke activity.....	7-15
Adding an Assign activity.....	7-18
<b>Deploying and Testing the BPEL Process at Run Time.....</b>	<b>7-24</b>
Deploying the BPEL Process.....	7-24
Testing the BPEL Process.....	7-25

## **8 Using Java APIs for Forms Services**

<b>Overview.....</b>	<b>8-1</b>
<b>Using Java APIs for Forms Services at Design Time.....</b>	<b>8-1</b>
Creating a New BPEL Project.....	8-6
Creating a Partner Link.....	8-7
Adding a Partner Link for File Adapter.....	8-11
Adding Invoke Activities.....	8-15
Adding Assign Activities.....	8-19
<b>Deploying and Testing the BPEL Process.....</b>	<b>8-27</b>
Deploying the BPEL Process.....	8-27
Testing the BPEL Process.....	8-28

## **9 Using Composite Services - BPEL**

<b>Overview.....</b>	<b>9-1</b>
<b>Viewing Composite Services.....</b>	<b>9-2</b>
<b>Downloading Composite Services.....</b>	<b>9-2</b>
<b>Modifying and Deploying BPEL Processes.....</b>	<b>9-4</b>

## **10 Creating and Using Custom Integration Interfaces**

<b>Overview</b> .....	10-1
<b>Creating Custom Integration Interfaces</b> .....	10-2
Creating Custom Integration Interfaces of Interface Types.....	10-2
Creating Custom Integration Interfaces of Composite Services.....	10-8
Creating Custom Business Events Using Workflow XML Loader.....	10-14
<b>Using Custom Integration Interfaces as Web Services</b> .....	10-22
Using Custom Interface WSDL in Creating a BPEL Process at Design Time.....	10-23
Creating a New BPEL Project.....	10-26
Creating a Partner Link for the Web Service.....	10-28
Adding a Partner Link for File Adapter.....	10-31
Adding Invoke Activities.....	10-36
Adding Assign Activities.....	10-39
Deploying and Testing the BPEL Process at Run Time.....	10-46
Deploying the BPEL Process.....	10-47
Testing the BPEL Process.....	10-49

## **11 Working With Oracle Workflow Business Event System to Invoke Web Services**

<b>Oracle Workflow and Service Invocation Framework Overview</b> .....	11-1
<b>Web Service Invocation Using Service Invocation Framework</b> .....	11-2
Understanding Message Patterns in WSDL.....	11-3
Defining Web Service Invocation Metadata.....	11-5
Step 1: Creating a Web Service Invoker Business Event.....	11-6
Step 2: Creating Local and Error Event Subscriptions to the Invoker Event.....	11-8
Step 3: Creating a Receive Event and Subscription (Optional).....	11-17
Understanding Web Service Input Message Parts.....	11-21
Supporting WS-Security.....	11-26
<b>Calling Back to Oracle E-Business Suite With Web Service Response</b> .....	11-28
<b>Invoking Web Services</b> .....	11-30
<b>Managing Errors</b> .....	11-36
<b>Testing Web Service Invocation</b> .....	11-37
<b>Troubleshooting Web Service Invocation Failure</b> .....	11-43
<b>Extending Seeded Java Rule Function</b> .....	11-48
<b>Other Invocation Usage Considerations</b> .....	11-54

## **A Integration Repository Annotation Standards**

<b>General Guidelines</b> .....	A-1
<b>Java Annotations</b> .....	A-4

PL/SQL Annotations.....	A-11
Concurrent Program Annotations.....	A-17
XML Gateway Annotations.....	A-19
Business Event Annotations.....	A-31
Business Entity Annotation Guidelines.....	A-37
Composite Service - BPEL Annotation Guidelines.....	A-105
Glossary of Annotations.....	A-112

## **B Configuring Server Connection**

Overview.....	B-1
---------------	-----

## **C Sample Payload**

Sample Payload for Creating Supplier Ship and Debit Request.....	C-1
Sample Payload for Inbound Process Purchase Order XML Transaction.....	C-3

## **D Understanding Basic BPEL Process Creation**

Overview.....	D-1
---------------	-----

## **Glossary**

## **Index**



---

# Send Us Your Comments

## **Oracle E-Business Suite Integrated SOA Gateway Developer's Guide, Release 12.1**

### **Part No. E12065-06**

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 Oracle E-Business Suite Release Online Documentation CD available on My Oracle Support and [www.oracle.com](http://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](mailto: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](http://www.oracle.com).



---

# Preface

## Intended Audience

Welcome to Release 12.1 of the *Oracle E-Business Suite Integrated SOA Gateway Developer's Guide*.

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

- The principles and customary practices of your business area.
- Computer desktop application usage and terminology.
- Oracle E-Business Suite integration interfaces.
- B2B, A2A and BP integrations.

This documentation assumes familiarity with Oracle E-Business Suite. It is written for the technical consultants, implementers and system integration consultants who oversee the functional requirements of these applications and deploy the functionality to their users.

If you have never used Oracle E-Business Suite, we suggest you attend one or more of the Oracle E-Business Suite training classes available through Oracle University.

See Related Information Sources on page xiii for more Oracle E-Business Suite product information.

## Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at <http://www.fcc.gov/cgb/consumerfacts/trs.html>, and a list of phone numbers is available at <http://www.fcc.gov/cgb/dro/trsphonebk.html>.

## 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 Oracle E-Business Suite Integrated SOA Gateway Overview**
- 2 Discovering and Viewing Integration Interfaces**
- 3 Using PL/SQL APIs as Web Services**
- 4 Using XML Gateway Inbound and Outbound Interfaces**
- 5 Using Business Events Through Subscription Model**
- 6 Using Concurrent Programs**
- 7 Using Business Service Objects**
- 8 Using Java APIs for Forms Services**
- 9 Using Composite Services - BPEL**
- 10 Creating and Using Custom Integration Interfaces**
- 11 Working With Oracle Workflow Business Event System to Invoke Web Services**
- A Integration Repository Annotation Standards**
- B Configuring Server Connection**
- C Sample Payload**
- D Understanding Basic BPEL Process Creation**
- Glossary**



## Related Information Sources

This book is included on the Oracle E-Business Suite Documentation Library, which is supplied in the Release 12.1 Media Pack. You can download soft-copy documentation as PDF files from the Oracle Technology Network at <http://www.oracle.com/technology/documentation/>. The Oracle E-Business Suite Release 12.1 Documentation Library contains the latest information, including any documents that have changed significantly between releases. If substantial changes to this book are necessary, a revised version will be made available on the "virtual" documentation library on My Oracle Support (formerly Oracle*MetaLink*).

If this guide refers you to other Oracle E-Business Suite documentation, use only the latest Release 12.1 versions of those guides.

### Online Documentation

All Oracle E-Business Suite documentation is available online (HTML or PDF).

- **Online Help** - Online help patches (HTML) are available on My Oracle Support.
- **PDF Documentation** - See the Oracle E-Business Suite Documentation Library for current PDF documentation for your product with each release. The Oracle E-Business Suite Documentation Library is also available on My Oracle Support and is updated frequently.
- **Release Notes** - For information about changes in this release, including new features, known issues, and other details, see the release notes for the relevant product, available on My Oracle Support.
- **Oracle Electronic Technical Reference Manual** - The Oracle Electronic Technical Reference Manual (eTRM) contains database diagrams and a detailed description of database tables, forms, reports, and programs for each Oracle E-Business Suite product. This information helps you convert data from your existing applications and integrate Oracle E-Business Suite data with non-Oracle applications, and write custom reports for Oracle E-Business Suite products. The Oracle eTRM is available on My Oracle Support.

### Related Guides

You should have the following related books on hand. Depending on the requirements of your particular installation, you may also need additional manuals or guides.

#### Oracle Alert User's Guide

This guide explains how to define periodic and event alerts to monitor the status of your Oracle E-Business Suite data.

#### Oracle E-Business Suite Concepts

This book is intended for all those planning to deploy Oracle E-Business Suite Release 12, or contemplating significant changes to a configuration. After describing the Oracle

E-Business Suite architecture and technology stack, it focuses on strategic topics, giving a broad outline of the actions needed to achieve a particular goal, plus the installation and configuration choices that may be available.

### **Oracle E-Business Suite CRM System Administrator's Guide**

This manual describes how to implement the CRM Technology Foundation (JTT) and use its System Administrator Console.

### **Oracle E-Business Suite Desktop Integration Framework Developer's Guide**

Oracle E-Business Suite Desktop Integration Framework is a development tool that lets you define custom integrators for use with Oracle Web Applications Desktop Integrator. This guide describes how to define and manage integrators and all associated supporting objects, as well as how to download and upload integrator definitions.

### **Oracle E-Business Suite Developer's Guide**

This guide contains the coding standards followed by the Oracle E-Business Suite development staff. It describes the Oracle Application Object Library components needed to implement the Oracle E-Business Suite user interface described in the *Oracle E-Business Suite User Interface Standards for Forms-Based Products*. It provides information to help you build your custom Oracle Forms Developer forms so that they integrate with Oracle E-Business Suite. In addition, this guide has information for customizations in features such as concurrent programs, flexfields, messages, and logging.

### **Oracle E-Business Suite Flexfields Guide**

This guide provides flexfields planning, setup, and reference information for the Oracle E-Business Suite implementation team, as well as for users responsible for the ongoing maintenance of Oracle E-Business Suite product data. This guide also provides information on creating custom reports on flexfields data.

### **Oracle Application Framework Developer's Guide**

This guide contains the coding standards followed by the Oracle E-Business Suite development staff to produce applications built with Oracle Application Framework. This guide is available in PDF format on My Oracle Support and as online documentation in JDeveloper 10g with Oracle Application Extension.

### **Oracle Application Framework Personalization Guide**

This guide covers the design-time and run-time aspects of personalizing applications built with Oracle Application Framework.

### **Oracle E-Business Suite Installation Guide: Using Rapid Install**

This book is intended for use by anyone who is responsible for installing or upgrading Oracle E-Business Suite. It provides instructions for running Rapid Install either to carry out a fresh installation of Oracle E-Business Suite Release 12, or as part of an upgrade from Release 11i to Release 12. The book also describes the steps needed to install the technology stack components only, for the special situations where this is applicable.

### **Oracle Application Server Adapter for Oracle Applications User's Guide (Oracle**

## **Fusion Middleware Adapter for Oracle Applications User's Guide)**

This guide covers the use of Adapter for Oracle Applications in developing integrations between Oracle E-Business Suite and trading partners.

Please note that the user's guide can be found in the following documentation libraries:

- As part of the Oracle Application Server in 10g, *Oracle Application Server Adapter for Oracle Applications User's Guide* is available in the Oracle Application Server 10g Documentation Library.
- As part of the Oracle Fusion Middleware and SOA Suite in 11g, *Oracle Fusion Middleware Adapter for Oracle Applications User's Guide* is available in the Oracle Fusion Middleware 11g Documentation Library.

## **Oracle E-Business Suite System Administrator's Guide Documentation Set**

This documentation set provides planning and reference information for the Oracle E-Business Suite System Administrator. *Oracle E-Business Suite System Administrator's Guide - Configuration* contains information on system configuration steps, including defining concurrent programs and managers, enabling Oracle Applications Manager features, and setting up printers and online help. *Oracle E-Business Suite System Administrator's Guide - Maintenance* provides information for frequent tasks such as monitoring your system with Oracle Applications Manager, administering Oracle E-Business Suite Secure Enterprise Search, managing concurrent managers and reports, using diagnostic utilities including logging, managing profile options, and using alerts. *Oracle E-Business Suite System Administrator's Guide - Security* describes User Management, data security, function security, auditing, and security configurations.

## **Oracle E-Business Suite User's Guide**

This guide explains how to navigate, enter data, query, and run reports using the user interface (UI) of Oracle E-Business Suite. This guide also includes information on setting user profiles, as well as running and reviewing concurrent requests.

## **Oracle E-Business Suite User Interface Standards for Forms-Based Products**

This guide contains the user interface (UI) standards followed by the Oracle E-Business Suite development staff. It describes the UI for the Oracle E-Business Suite products and how to apply this UI to the design of an application built by using Oracle Forms.

## **Oracle Diagnostics Framework User's Guide**

This manual contains information on implementing and administering diagnostics tests for Oracle E-Business Suite using the Oracle Diagnostics Framework.

## **Oracle E-Business Suite Integrated SOA Gateway User's Guide**

This guide describes the high level service enablement process, explaining how users can browse and view the integration interface definitions and services residing in Oracle Integration Repository.

## **Oracle E-Business Suite Integrated SOA Gateway Implementation Guide**

This guide explains how integration repository administrators can manage and administer the Web service activities for integration interfaces including native packaged integration interfaces, composite services (BPEL type), and custom integration interfaces. It also describes how to invoke Web services from Oracle E-Business Suite by employing the Oracle Workflow Business Event System, and how to manage Web service security, configure logs, and monitor SOAP messages.

### **Oracle e-Commerce Gateway User's Guide**

This guide describes the functionality of Oracle e-Commerce Gateway and the necessary setup steps in order for Oracle E-Business Suite to conduct business with trading partners through Electronic Data Interchange (EDI). It also contains how to run extract programs for outbound transactions, import programs for inbound transactions, and the relevant reports.

### **Oracle e-Commerce Gateway Implementation Manual**

This guide describes implementation details, highlighting additional setup steps needed for trading partners, code conversion, and Oracle E-Business Suite. It also provides architecture guidelines for transaction interface files, troubleshooting information, and a description of how to customize EDI transactions.

### **Oracle Report Manager User's Guide**

Oracle Report Manager is an online report distribution system that provides a secure and centralized location to produce and manage point-in-time reports. Oracle Report Manager users can be either report producers or report consumers. Use this guide for information on setting up and using Oracle Report Manager.

### **Oracle iSetup Developer's Guide**

This manual describes how to build, test, and deploy Oracle iSetup Framework interfaces.

### **Oracle iSetup User's Guide**

This guide describes how to use Oracle iSetup to migrate data between different instances of the Oracle E-Business Suite and generate reports. It also includes configuration information, instance mapping, and seeded templates used for data migration.

### **Oracle Web Applications Desktop Integrator Implementation and Administration Guide**

Oracle Web Applications Desktop Integrator brings Oracle E-Business Suite functionality to a spreadsheet, where familiar data entry and modeling techniques can be used to complete Oracle E-Business Suite tasks. You can create formatted spreadsheets on your desktop that allow you to download, view, edit, and create Oracle E-Business Suite data, which you can then upload. This guide describes how to implement Oracle Web Applications Desktop Integrator and how to define mappings, layouts, style sheets, and other setup options.

### **Oracle Workflow Administrator's Guide**

This guide explains how to complete the setup steps necessary for any product that includes workflow-enabled processes. It also describes how to manage workflow processes and business events using Oracle Applications Manager, how to monitor the progress of runtime workflow processes, and how to administer notifications sent to workflow users.

### **Oracle Workflow Developer's Guide**

This guide explains how to define new workflow business processes and customize existing Oracle E-Business Suite-embedded workflow processes. It also describes how to define and customize business events and event subscriptions.

### **Oracle Workflow User's Guide**

This guide describes how users can view and respond to workflow notifications and monitor the progress of their workflow processes.

### **Oracle Workflow API Reference**

This guide describes the APIs provided for developers and administrators to access Oracle Workflow.

### **Oracle Workflow Client Installation Guide**

This guide describes how to install the Oracle Workflow Builder and Oracle XML Gateway Message Designer client components for Oracle E-Business Suite.

### **Oracle XML Gateway User's Guide**

This guide describes Oracle XML Gateway functionality and each component of the Oracle XML Gateway architecture, including Message Designer, Oracle XML Gateway Setup, Execution Engine, Message Queues, and Oracle Transport Agent. It also explains how to use Collaboration History that records all business transactions and messages exchanged with trading partners.

The integrations with Oracle Workflow Business Event System, and the Business-to-Business transactions are also addressed in this guide.

### **Oracle XML Publisher Report Designer's Guide**

Oracle XML Publisher is a template-based reporting solution that merges XML data with templates in RTF or PDF format to produce a variety of outputs to meet a variety of business needs. Using Microsoft Word or Adobe Acrobat as the design tool, you can create pixel-perfect reports from the Oracle E-Business Suite. Use this guide to design your report layouts.

This guide is available through the Oracle E-Business Suite online help.

### **Oracle XML Publisher Administration and Developer's Guide**

Oracle XML Publisher is a template-based reporting solution that merges XML data with templates in RTF or PDF format to produce a variety of outputs to meet a variety of business needs. Outputs include: PDF, HTML, Excel, RTF, and eText (for EDI and EFT transactions). Oracle XML Publisher can be used to generate reports based on existing Oracle E-Business Suite report data, or you can use Oracle XML Publisher's

data extraction engine to build your own queries. Oracle XML Publisher also provides a robust set of APIs to manage delivery of your reports via e-mail, fax, secure FTP, printer, WebDav, and more. This guide describes how to set up and administer Oracle XML Publisher as well as how to use the Application Programming Interface to build custom solutions.

This guide is available through the Oracle E-Business Suite online help.

## **Integration Repository**

The Oracle Integration Repository is a compilation of information about the service endpoints exposed by the Oracle E-Business Suite of applications. It provides a complete catalog of Oracle E-Business Suite's business service interfaces. The tool lets users easily discover and deploy the appropriate business service interface for integration with any system, application, or business partner.

The Oracle Integration Repository is shipped as part of the E-Business Suite. As your instance is patched, the repository is automatically updated with content appropriate for the precise revisions of interfaces in your environment.

## **Do Not Use Database Tools to Modify Oracle E-Business Suite Data**

Oracle **STRONGLY RECOMMENDS** that you never use SQL\*Plus, Oracle Data Browser, database triggers, or any other tool to modify Oracle E-Business Suite 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 E-Business Suite data, you risk destroying the integrity of your data and you lose the ability to audit changes to your data.

Because Oracle E-Business Suite tables are interrelated, any change you make using an Oracle E-Business Suite form can update many tables at once. But when you modify Oracle E-Business Suite data using anything other than Oracle E-Business Suite, 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 E-Business Suite.

When you use Oracle E-Business Suite to modify your data, Oracle E-Business Suite automatically checks that your changes are valid. Oracle E-Business Suite 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.

---

# Oracle E-Business Suite Integrated SOA Gateway Overview

## Oracle E-Business Suite Integrated SOA Gateway Overview

Building on top of Oracle Fusion Middleware and service-oriented architecture (SOA) technology, Oracle E-Business Suite Integrated SOA Gateway (ISG) provides a customer-focused robust communication and integration infrastructure between independently managed components and loosely coupled applications. This infrastructure not only allows greater and effective business integration between heterogeneous applications, but also facilitates the development and execution of complex business processes into highly flexible and reusable Web services. With this standardized and interoperable Web service platform, Oracle E-Business Suite Integrated SOA Gateway provides a powerful framework that accelerates dynamic business processes and service integration between applications over the Web.

Oracle E-Business Suite Integrated SOA Gateway is a complete set of service infrastructure. It supports almost all integration interface types and services invoked within Oracle E-Business Suites no matter if they are native packaged interfaces or the services that are orchestrated using native services. With this pre-built, reusable business services and service-oriented components, Oracle E-Business Suite Integrated SOA Gateway provides a capability of allowing various users to perform different tasks and to monitor and manage service integration throughout the entire service deployment life cycle.

For example, system integration developers can perform end-to-end service integration activities including creating and annotating custom integration interfaces, orchestrating discrete Web services into meaningful end-to-end business processes, defining Web service invocation metadata, and testing the Web service invocation.

Application users or system integration analysts can then browse through and search on available integration interfaces and services, regardless of custom or Oracle packaged ones, as well as view each interface details through the centralized repository.

Integration repository administrators can take further actions on transforming native

interfaces into Web services, and then deploying the services for public use and access. The administrators are also responsible for enforcing service related securities, monitoring and managing the entire integrated service deployment life cycle to ensure smooth service integration between applications.

With pre-built, reusable business services and an essential service-oriented framework allowing service generation, deployment, invocation, and management, Oracle E-Business Suite Integrated SOA Gateway is the intrinsic part of Oracle E-Business Suite for service enablement. It not only enables services within and beyond Oracle E-Business Suite, but also facilitates dynamic business execution through a seamless service integration and consumption over the internet.

For more information about each integration interface and service, see *Oracle E-Business Suite Integrated SOA Gateway User's Guide*; for more information on implementing and administering Oracle E-Business Suite Integrated SOA Gateway, see *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

## Major Features

Oracle E-Business Suite Integrated SOA Gateway contains the following features:

- Provide robust, consistent integration framework with extensive infrastructure based on SOA principles
- Integrate loosely coupled and heterogeneous applications
- Contain pre-built and reusable business services
- Provide native service enablement capability within the Oracle E-Business Suite
- Use native services as building blocks to create composite services
- Support annotated custom integration interfaces from Oracle Integration Repository
- Enforce function security and role-based access control security to allow only authorized users to execute administrative functions
- Support multiple authentication types for inbound service requests in securing Web service content and authenticating Web service operations
- Provide centralized, user-friendly user interface for logging configuration
- Enable Web service invocation from Oracle E-Business Suite
- Audit and monitor Oracle E-Business Suite service operations from native SOA Monitor



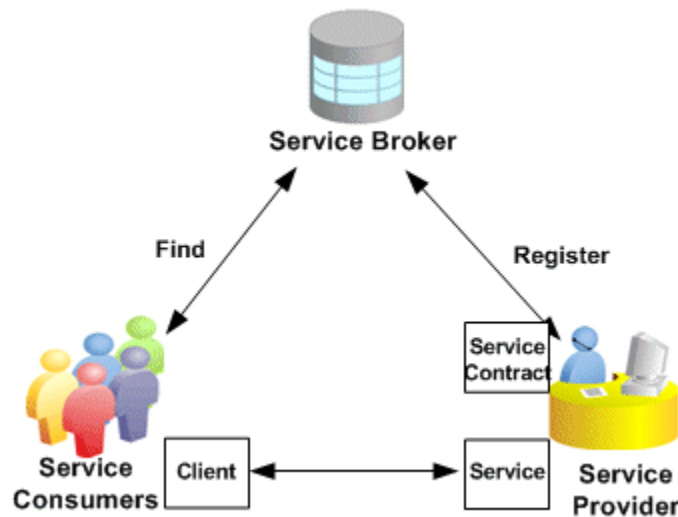
## Major Components Features and Definitions

The better understand Oracle E-Business Suite Integrated SOA Gateway and its key components, this section describes some key features and the definition of each component.

### Native Service Enablement

Service enablement is the key feature within Oracle E-Business Suite Integrated SOA Gateway. It provides a mechanism that allows native packaged integration interface definitions residing in Oracle Integration Repository to be further transformed into Web services that comply with Web standards. Additionally, these services can be deployed from the Integration Repository to the application server allowing more consumptions over the Web.

To understand the basic concept of Web services and how the service works, the following diagram illustrates the essential components of enabling services:



A Service Provider is the primary engine underlying the Web services. It facilitates the service enablement for various types of interfaces.

A Service Consumer (Web service client) is the party that uses or consumes the services provided by the Service Provider.

A Service Broker (Service Registry) describes the service's location and contract to ensure service information is available to any potential service consumer.

### Composite Services

Composite services use the native service as building blocks to construct the sequence of business flows. Basically, this interface type orchestrates the invocation sequence of discrete Web services into a meaningful end-to-end business process through a Web

service composition language BPEL (business process execution language).

For example, use Oracle BPEL Process Manager (BPEL PM) to integrate the Order-to-Receipt business process that contains sales order entry, item availability check, pack and ship, and invoice to Accounts Receivable sub processes handled by various applications. This approach effectively tightens up the control of each individual process and makes the entire business flow more efficiently.

### Oracle Integration Repository and Service Enablement

Oracle Integration Repository, an integral part of Oracle E-Business Suite, is the centralized repository that contains numerous interface endpoints exposed by applications within the Oracle E-Business Suite.

To effectively manage all integration interfaces and services incurred within the Oracle E-Business Suite, Oracle E-Business Suite Integrated SOA Gateway now supports complex business processes or composite services, Web service generation and deployment, as well as business event subscriptions through the centralized Integration Repository.

You can browse these interface definitions and services through the Oracle Integration Repository user interfaces. Users with administrator privileges can further perform administrative tasks through the same interfaces.

Oracle Integration Repository supports the following interface types:

- PL/SQL
- XML Gateway
- Concurrent Programs
- Business Events
- Open Interface Tables/Views
- EDI
- Business Service Object (Service Beans)
- Java APIs for Forms

**Note:** Java APIs for Forms are XML document-based integration points wrapped in Java classes for executing business logic in Oracle Forms. These specialized Java classes are categorized as a subtype of Java interface.

- Composite Services

## **Service Invocation Framework**

To invoke all integration services from Oracle E-Business Suite, Oracle E-Business Suite Integrated SOA Gateway uses the Service Invocation Framework (SIF) that leverages Oracle Workflow Java Business Event System (JBES) and a seeded Java rule function to allow any WSDL-described service to be invoked.

By using this service invocation framework, developers or implementors can interact with Web services through WSDL descriptions instead of working directly with SOAP APIs, the usual programming model. This approach lets you use WSDL as a normalized description of disparate software, and allows you to access this software in a manner that is independent of protocol or location.

Since this feature is the major development framework in invoking Web services within the entire Oracle E-Business Suite, detailed implementation information is described in a separate chapter in this book.

See *Web Service Invocation Using Service Invocation Framework*, page 11-2.

## **SOA Monitor**

SOA Monitor is a centralized, light-weight service execution monitoring and management tool. It not only monitors all the SOAP requests that SOA Provider and Web Service Provider process, but also provides auditing feature for the SOAP messages if the auditing feature is enabled.

With SOA Monitor, the Integration Repository Administrator can effectively manage and identify errors incurred during the service deployment life cycle and take necessary actions to expedite the interaction between services.

## **Manage Security**

Security is the most critical feature that is designed to guard service content from unauthorized access.

To ensure secure access and the execution of integration interfaces and Web services, Oracle E-Business Suite integrated SOA Gateway enforces the security rules through security grants to authorize interface methods access or feature access (such as the downloading composite services feature) to appropriate users. Multiple organization access control security rule is also implemented for authorizing interface execution related to multiple organizations.

Additionally, Web service security rule is enforced for Web service authentication, requiring an username and password to be passed as part of the security header in the SOAP request sent to the Web service.



---

# Discovering and Viewing Integration Interfaces

## Overview

Similar to regular users or system integration analysts, system integration developers can view integration interfaces and their details from Oracle Integration Repository, as well as review generated or deployed Web service WSDL files in the appropriate Web Service region. The developers cannot perform administrative tasks, such as generating or deploying Web services, which are done by the integration repository administrators.

However, the developers have more privileges than the analysts in viewing all types of integration interfaces including public, private, and internal interface types from Oracle Integration Repository. These privileges allow developers to have sufficient integration interface information which could be useful to better understand each integration interface from different perspectives.

**Note:** System integration analysts can view *Public* integration interfaces only, and they do not have the access privileges to view *Private to Application* and *Internal to Oracle* interfaces from the Oracle Integration Repository.

This section covers the following topics:

- Searching and Viewing Integration Interfaces, page 2-1
- Reviewing Interface Details, page 2-6
- Reviewing WSDL Element Details, page 2-10

## Searching and Viewing Integration Interfaces

To better understand each integration interface and the integration between different

applications, Oracle E-Business Suite Integrated SOA Gateway allows system integration developers and integration repository administrators to have more interface access privileges in viewing all integration interface types regardless of public, private, or internal interface types.

### **Browsing the Integration Interfaces**

When viewing integration interfaces, you can browse by product family, by interface type, or by standard based on your selection in the View By drop-down list. Expand the navigation tree in one of these views to see a list of the available interfaces.

For more information on how to browse the interfaces, see *Browsing the Integration Interfaces, Oracle E-Business Suite Integrated SOA Gateway User's Guide*.

### **Searching the Integration Interfaces**

To search for an integration interface, click **Search** to access the main Search page. After clicking the **Show More Search Options** link in the Search page, you can find *Private to Application* and *Internal to Oracle* interface types along with *Public* and *All* displayed from the Scope drop-down menu. If 'All' is selected from the Scope field, then all integration interfaces including public, private to application, and internal to Oracle interfaces can be listed in the results region.

**Note:** System integration analysts can view *Public* integration interfaces only, and they do not have the access privileges to view *Private to Application* and *Internal to Oracle* interfaces from the Oracle Integration Repository.

In addition, they can only find 'All' (default) and 'Public' list of values available from the Scope drop-down list. And only Public integration interfaces will be retrieved and listed in the search result even if they do not change the default value 'All' in the Scope field.

For detailed information on Public, Private to Application, and Internal to Oracle, see *Scope, Oracle E-Business Suite Integrated SOA Gateway User's Guide*.

By using the search feature, you can easily locate a deployed Web service for a particular product or product family if you want to use the deployed service for a partner link creation while orchestrating the BPEL process.

For example, to locate all deployed Web services for concurrent program, first select 'Concurrent Program' from the Interface drop-down list and then click **Show More Search Options** to select 'Deployed' for the Web Service Status field. After executing the search, you should find all deployed Web services for the concurrent program interface type.

## Searching for Deployed Web Services

**ORACLE** Integration Repository

Home Logout Preferences Help Diagnostics

Integration Repository

Search

Browse

Interface Name

Product Family

Product

Internal Name

Interface Type

Business Entity

Hide More Search Options

TIP Select Category before a Category Value

Category

Category Value

Interface Source

Status

Web Service Status

Standard

Standard Specification

Scope

Go

Clear All

Export

Name	Internal Name	Product	Type	Source	Status	Description
<a href="#">Departure Shipment Notice Outbound</a>	WSHDSNO	Shipping Execution Common	Concurrent Program	Oracle	Active	This concurrent program generates a Departure Ship Notice Outbound(DSNO).
<a href="#">Transaction Layout Definition</a>	ECRDTLD	e-Commerce Gateway	Concurrent Program	Oracle	Active	Reports the layout of a specified transaction data file.

Browse

Integration Repository SOA Monitor Home Logout Preferences Help Diagnostics

About this Page Privacy Statement

Copyright (c) 2006, Oracle. All rights reserved.

## Searching for Java APIs for Forms Interfaces

Java APIs for Forms interfaces are XML document-based integration points wrapped in Java classes for executing business logic in Oracle Forms. These specialized Java classes are categorized with *subtype* 'Java APIs for Forms' and displayed in the Integration Repository under the Java interface type.

To locate a Java APIs for Forms interface, you must perform a search by clicking **Show More Search Options** to display more search fields. Enter the following key search values along with any product family or scope if needed as the search criteria:

- Category: Interface Subtype
- Category Value: Java APIs for Forms

## Searching for Java APIs for Forms Interfaces

**ORACLE** Integration Repository

Home Logout Preferences Help Personalize Page Diagnostics

Integration Repository

Search

Interface Name

Product Family 

Order Management Suite

Product 

All

Internal Name

Interface Type 

All

Business Entity

☐ Hide More Search Options

☒ TIP Select Category before a Category Value

Category 

Interface Subtype

Category Value 

Java APIs for Forms

Interface Source 

All

Status 

All

Web Service Status 

All

Standard 

All

Standard Specification

Scope 

All

Go

Clear All

Export

Name	Internal Name	Product	Type	Source	Status	Description
<a href="#">CreateAndMaintainSalesOrders</a>	oracle.apps.ont.services.oexoeord.OEXOEORDServices_DocStyle	Order Management	Java	Oracle	Active	The service end-point provides capabilities to Create, Delete, Query and Update a Sales Order.

Browse

To view an interface details, click the interface name link that you would like to view from the search result region to view the information details. See *Reviewing Interface Details*, page 2-6.

## Searching for Custom Integration Interfaces

Annotated custom interface definitions, once they are uploaded successfully, are merged into the interface types they belong to and displayed together with Oracle interfaces from the Integration Repository browser window. To easily distinguish annotated custom interface definitions from Oracle interfaces, the Interface Source "Custom" is used to categorize those custom integration interfaces in contrast to Interface Source "Oracle" for Oracle interfaces.

Therefore, you can search for custom integration interfaces by clicking **Show More Search Options** to display more search fields.



## Searching for Custom Integration Interfaces

**ORACLE** Integration Repository

Home Logout Preferences Help Diagnostics

Integration Repository

Search Browse

Interface Name  Internal Name

Product Family  All Interface Type  PL/SQL

Product  All Business Entity

[Hide More Search Options](#)

**TIP** Select Category before a Category Value

Category  All Web Service Status  All

Category Value  Standard  All

Interface Source  Custom Standard Specification

Status  All Scope  All

Custom  Clear All

Oracle

Name	Internal Name	Product	Type	Source	Status	Description
<a href="#">Custom_SDR</a>	CUSTOM_SDR	Trade Management	PL/SQL	Custom	Active	This custom PL/SQL package can be used for SDR
<a href="#">Custom_SDR1</a>	CUSTOM_SDR1	Trade Management	PL/SQL	Custom	Active	This custom PL/SQL package can be used for SDR
<a href="#">Custom_Supplier_Ship_and_Debit_Request</a>	CUSTOM_SD_REQUEST	Trade Management	PL/SQL	Custom	Active	This custom PL/SQL package This package can be used to Create Ship & Debit Request

Browse

Integration Repository SOA Monitor Home Logout Preferences Help Diagnostics

About this Page Privacy Statement Copyright (c) 2006, Oracle. All rights reserved.

Enter the following information along with any interface type, product family, or scope if needed as the search criteria:

- Interface Source: Custom

For information on how to view custom integration interfaces, see *Viewing Custom Integration Interfaces, Oracle E-Business Suite Integrated SOA Gateway User's Guide*.

For more information on each search field in the Search page, see *Searching for an Integration Interface, Oracle E-Business Suite Integrated SOA Gateway User's Guide*.

### To search for all integration interface types:

1. Log on Oracle Integration Repository with the username granted with the system integration developer role. Select the Integrated SOA Gateway responsibility from the navigation menu. Select the Integration Repository link to open the repository browser.
2. Click **Search** to open the main Search page.
3. Enter appropriate search information such as product family, product, interface type, or business entity.

4. Click **Show More Search Options** to open more search options.
  - To search custom integration interfaces, select 'Custom' in the Interface Source field.
  - To search Java APIs for Forms interfaces, select 'Interface Subtype' in the Category field and 'Java API for Forms' in the Category Value field.
5. To view deployed integration interfaces, select 'Deployed' from the Web Service Status field drop-down list.
6. To view all integration interfaces, select All from the Scope field. This allows all integration interfaces including Public, Internal to Oracle, and Private to Application displayed in the results region.
7. To view integration interfaces of Public, Internal to Oracle, or Private to Application type, select 'Public', 'Internal to Oracle', or 'Private to Application' from the Scope drop-down list respectively.
8. Click **Go** to execute the search. All interfaces that match your search criteria are displayed.
9. Select an interface type from the search result to view the interface details.

## Reviewing Interface Details

After searching for an integration interface, integration developers can review a selected interface details by clicking on an interface name from the search result page. This opens the interface details page where you can view the interface general information, a description region, a source region, and an interface methods or procedure and functions region.

If the selected interface has a Web service generated successfully, then the Web Service - SOA Provider region is displayed in the interface details page.

## Viewing Interface Details Page

The screenshot shows the Oracle Integration Repository interface. At the top, the Oracle logo and 'Integration Repository' are displayed. Navigation links include Home, Logout, Preferences, Help, and Diagnostics. The breadcrumb trail shows 'Integration Repository > PL/SQL Interface : Message Dictionary'. On the right, there are buttons for 'Browse', 'Search', and 'Printable Page'.

Metadata for the interface is listed on the left:

- Internal Name: **FND\_MESSAGE**
- Type: **PL/SQL**
- Product: **Application Object Library**
- Status: **Active**
- Business Entity: [Applications Message Dictionary](#)
- Online Help: [See the related online help](#)

On the right, the 'Scope' is **Public** and the 'Interface Source' is **Oracle**.

The 'Full Description' section states: 'APIs to Set, Retrieve, Clear the messages in Message Stack.'

The 'Web Service - SOA Provider' section shows:

- Web Service Status: **Deployed**
- WSDL: [View WSDL](#)
- \* Authentication Type: ☒ Username Token, ☒ SAML Token (Sender Vouches)

The 'Source Information' section shows:

- Source File: **patch/115/sql/AFRLMSG6.pls**
- Source Version: **120.2.12000000.1**
- Source Product: **FND**

The 'Procedures and Functions' section contains a table:

Name	Internal Name	Status	Description
<a href="#">Clear Message Buffer</a>	CLEAR	Active	Clears the message stack of all messages
<a href="#">Get Message Text</a>	GET	Active	Retrieves a translated and token-substituted message from the message stack and then clears that message from the message stack.
<a href="#">Get Message Text and Number</a>	GET_TEXT_NUMBER	Active	Retrieves the message text and message number from fnd_new_messages table for a given message name.
<a href="#">Set Message Name</a>	SET_NAME	Active	In Database Server, this Sets a message name in the global area without actually retrieving the message from Message Dictionary.
<a href="#">Set Token value</a>	SET_TOKEN	Active	In Database Server, SET_TOKEN adds a token/value pair to the global area without actually doing the substitution.

At the bottom right, there are buttons for 'Browse', 'Search', and 'Printable Page'.

**Note:** For Business Service Object interface type, since it is service enabled by Web Service Provider, you will find the Web Service - Web Service Provider region instead in the interface details if the services are available.

If it is for XML Gateway Map interface type, you may also find the Web Service - Web Service Provider region available. This is because XML Gateway Map interfaces can be service enabled by Web Service Provider in Oracle E-Business Suite Release 12.0. Hence, if your system is upgraded from the Release 12.0, and your system has already have service enabled through Web Service Provider in the Release 12.0, you can find the Web Service - Web Service Provider region displayed along with the Web Service - SOA Provider region if you also have service available in this release.

In the Web Service - SOA Provider region (or Web Service - Web Service Provider region), you can notice the following fields:

- **Web Service Status:** This field indicates whether the service is deployed or not. It can have either one of the following possible values:

- **Generated:** It indicates that the selected interface has WSDL description available.
- **Deployed:** It indicates that the selected interface has been successfully deployed to the application server.
- **View WSDL link:** Click this link allowing you to view the generated or deployed WSDL code.

For more information on how to view WSDL file, see *Reviewing WSDL Element Details*, page 2-10.

- **Authentication Type:** This field contains the following read-only check boxes:
  - **Username Token**  
This authentication type provides username and password information in the security header for a Web service provider to use in authenticating the SOAP request. It is the concept of Oracle E-Business Suite username/password (or the username/password created through the Users window in defining an application user).
  - **SAML Token (Sender Vouches)**  
This authentication type is used for Web services relying on sending a username only through SAML Assertion.

These authentication types are used by SOA Provider to secure Web service content and authenticate Web service operation. Prior to deploying or redeploying a Web service generated by SOA Provider, an integration repository administrator must first select at least one authentication type. Once the service has been successfully deployed, the Web Service Status field is changed from 'Generated' to 'Deployed' along with the selected authentication type(s).

For more information on how to deploy a service, see *Deploying and Undeploying Web Services*, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

**Note:** Please note that not all integration interface definitions can be service enabled. Oracle Integration Repository supports service enablement only for the following interface types:

- PL/SQL
- XML Gateway Map (inbound)
- Concurrent Program
- Business Service Object (Service Beans)

- Java APIs for Forms

Java APIs for Forms are XML document-based integration points wrapped in Java classes for executing business logic in Oracle Forms. These specialized Java classes are categorized as a subtype of Java interface.

The Business Event and XML Gateway Map (outbound) interface types are supported through subscription model. Non-service enabled public interfaces are Open Interface Tables, Open Interface Views, and EDI interface. For the Composite Services - BPEL interface type, since it uses native integration services as building blocks to orchestrate a business process with service endpoints through BPEL language, this type of interface itself is already service enabled.

Based on the interface type support model described above, for those interface types that can be service enabled, an integration repository administrator can perform additional tasks to generate, deploy, or redeploy a Web services for a selected interface type. Additionally, the administrator can perform other administrative tasks including subscribing to a selected business event, creating security grants, and viewing available log messages written during service generation and deployment. For detailed information on these administrative tasks, see:

- Generating Web Services, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*
- Deploying and Undeploying Web Services, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*
- Subscribing to Business Events, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*
- Creating Grants, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*
- Viewing Generate and Deploy Time Logs, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*

Once the Web services representing in WSDL are generated, integration developers can then use them in creating a composite service - BPEL process to insert or update Oracle E-Business Suite.

How to use WSDL definitions in creating composite service - BPEL processes, see each individual chapter described in this book for details.

## Reviewing WSDL Element Details

If an interface can be exposed as a Web service, the corresponding WSDL file is created and can be accessed through the interface details page.

By clicking the View WSDL link, a new window containing the WSDL document appears. This XML-based document describes a selected Web service as a set of endpoints operating on messages containing document-oriented information.

### Locating the Interface Exposed as a Web Service

The screenshot displays the Oracle Integration Repository web application. The top navigation bar includes the Oracle logo, 'Integration Repository', and links for Home, Logout, Preferences, Help, and Diagnostics. The main content area shows the 'Integration Repository >' breadcrumb and the selected interface 'PL/SQL Interface : Invoice Creation'. On the right, there are 'Browse', 'Search', and 'Printable Page' buttons.

Interface Details:

- Internal Name: **AR\_INVOICE\_API\_PUB**
- Type: **PL/SQL**
- Product: **Receivables**
- Status: **Active**
- Business Entity: [Receivables Invoice](#)
- Meta Link: [See OracleMetaLink note 236938.1](#)
- Scope: **Public**
- Interface Source: **Oracle**

**Full Description**

Invoice Creation API allows users to create an invoice using simple calls to PL/SQL functions. Invoices can be created either in batch mode with multiple invoices or a single invoice. The Invoice Creation API is not intended to replace the existing Transaction workbench, AutoInvoice, or the Transaction API program.

**Web Service - SOA Provider**

Web Service Status: **Deployed**  
WSDL: [View WSDL](#)

\* Authentication Type: ☐ Username Token ☒ SAML Token (Sender Vouches)

**Source Information**

Source File: **patch/115/sql/ARXPINVS.pls**  
Source Version: **120.25.12010000.4**  
Source Product: **AR**

**Procedures and Functions**

Name	Internal Name	Status	Description
<a href="#">Create Invoice in a Batch</a>	CREATE_INVOICE	Active	Use this procedure to create multiple invoices in a batch.
<a href="#">Create Single Invoice</a>	CREATE_SINGLE_INVOICE	Active	Use this procedure to create a single invoice and return a customer transaction ID.

At the bottom, there are 'Browse', 'Search', and 'Printable Page' buttons, and a footer with 'About this Page', 'Privacy Statement', 'Integration Repository Home Logout Preferences Help Diagnostics', and 'Copyright (c) 2005, Oracle. All rights reserved.'

For example, click the deployed View WSDL link for the PL/SQL: Invoice Creation from the interface details page, the WSDL document appears.

```

<?xml version="1.0" encoding="UTF-8" ?>
- <definitions name="AR_INVOICE_API_PUB" targetNamespace="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/"
  xmlns:tns="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/" xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns1="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/create_invoice/"
  xmlns:tns2="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/create_single_invoice/"
- <types>
- <schema xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/create_invoice/"
  <include
    schemaLocation="http://rws60132rems.us.oracle.com:8009/webservices/SOAPProvider/plsql/ar_invoice_api_pub/APPS_ISG_CREATE_INVOICE_AR_INVO
    -24CREATE_I61.xsd" />
  </schema>
- <schema xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/create_single_invoice/"
  <include
    schemaLocation="http://rws60132rems.us.oracle.com:8009/webservices/SOAPProvider/plsql/ar_invoice_api_pub/APPS_ISG_CREATE_SINGLE_INVOICE
    -24CREATE_S89.xsd" />
  </schema>
- <schema xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  targetNamespace="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_invoice_api_pub/"
- <element name="SOAHeader">
- <complexType>
- <sequence>
  <element name="Responsibility" minOccurs="0" type="string" />
  <element name="RespApplication" minOccurs="0" type="string" />
  <element name="SecurityGroup" minOccurs="0" type="string" />
  <element name="NLSLanguage" minOccurs="0" type="string" />
  <element name="Org_Id" minOccurs="0" type="string" />
  </sequence>
</complexType>
</element>
</schema>
</types>
- <message name="CREATE_INVOICE_Input_Msg">
  <part name="header" element="tns:SOAHeader" />
  <part name="body" element="tns1:InputParameters" />
</message>
- <message name="CREATE_INVOICE_Output_Msg">
  <part name="body" element="tns1:OutputParameters" />
</message>
- <message name="CREATE_SINGLE_INVOICE_Input_Msg">
  <part name="header" element="tns:SOAHeader" />
  <part name="body" element="tns2:InputParameters" />
</message>
- <message name="CREATE_SINGLE_INVOICE_Output_Msg">
  <part name="body" element="tns2:OutputParameters" />
</message>
- <portType name="AR_INVOICE_API_PUB_PortType">
- <operation name="CREATE_INVOICE">
  <input message="tns:CREATE_INVOICE_Input_Msg" />

```

**Note:** The http:// address in the new window has the exact WSDL URL information that appeared in the interface details page. This address can be copied and used directly in any of the Web service clients for invoking Web services.

For example, it can be used while creating a partner link for the invocation of the interface that is exposed as Web service in a BPEL process.

## WSDL Document Structure

A WSDL document is simply a set of definitions. There is a **definitions** element at the root, and definitions inside. The *definitions* element defines the set of services that the Web service offers.

It often contains an optional TargetNamespace property, a convention of XML schema that enables the WSDL document to refer to itself.

The structure of this definitions element can be like:

```

<definitions name="nmtoken"
  <targetNamespace="uri">
    <import namespace="uri" location="uri"/> *
</definitions>

```

For example, a corresponding WSDL document for the Invoice Creation API

(AR\_INVOICE\_API\_PUB) that is exposed as a Web service appears in a new window.

```
<definitions name="AR_INVOICE_API_PUB"
targetNamespace="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_in
voice_api_pub/"
xmlns:tns="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_invoice_
api_pub/
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns1="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_invoice_
api_pub/create_invoice/"
xmlns:tns2="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_invoice_
api_pub/create_single_invoice/>
```

For example, the *definitions* element specifies that this WSDL document is the called 'AR\_INVOICE\_API\_PUB'. It also specifies numerous namespaces that will be used throughout the remainder of the document. It also specifies a default namespace: `xmlns=http://schemas.xmlsoap.org/wsdl/`.

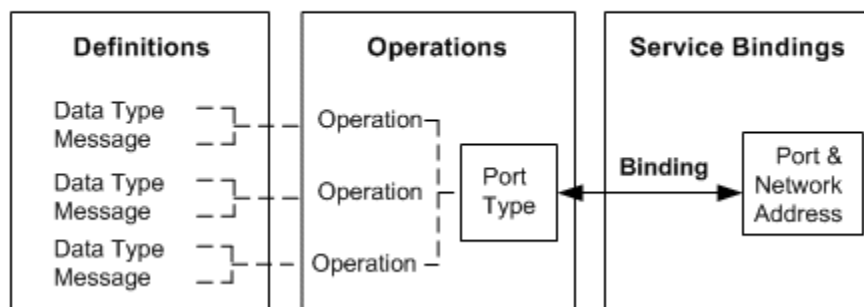
In addition to the *definitions* element, Web services are defined using the following six major elements:

- **Types:** It provides data type definitions used to describe the messages exchanged.
- **Message:** It represents an abstract definition of the data being transmitted.  
A message consists of logical parts, each of which is associated with a definition within some type system.
- **PortType:** It is a set of abstract operations. Each operation refers to an input message and output messages.
- **Binding:** It specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType.
- **Port:** It specifies an address for a binding, thus defining a single communication endpoint.
- **Service:** It is used to aggregate a set of related ports.

The following diagram shows the relationship of the basic parts of WSDL:



## WSDL Basic Parts Relationship Diagram



## Types

The **types** element contains all data types used in all method calls described in the WSDL. It can be used to specify the XML Schema (xsd:schema) that is used to describe the structure of a WSDL Part.

The structure of this Types element can be like:

```
<definitions...>
  <types>
    <xsd:schema.../>*
  </types>
</definitions>
```

For example, the Invoice Creation Web service contains the following two functions:

- CREATE\_INVOICE
- CREATE\_SINGLE\_INVOICE

Each function is described in the data type definition. WSDL prefers the use of XSD as the type of system mechanism to define the types in a message schema. As a result, the message schema location of the CREATE\_INVOICE function is defined in the APPS\_XX\_BPEL\_CREATE\_INVOICE\_AR\_INVOICE\_API\_PUB-24CREATE\_INV.xsd. The message schema location of the CREATE\_SINGLE\_INVOICE function is defined in the APPS\_XX\_BPEL\_CREATE\_SINGLE\_INVOICE\_AR\_INVOICE\_API\_PUB-24CREATE\_SINGLE\_IN.xsd.

```

<types>
  <schema xmlns="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"

    targetNamespace="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_in
voice_api_pub/create_invoice/">
    <include

    schemaLocation="https://host.us.oracle.com:1234/webservices/SOAPProvider/
plsql/ar_invoice_api_pub/APPS_XX_BPEL_CREATE_INVOICE_AR_INVOICE_API_PUB-
24CREATE_INV.xsd"/>
    </schema>
    <schema xmlns="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified"

      targetNamespace="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_in
voice_api_pub/create_single_invoice/">
      <include

      schemaLocation="https://host.us.oracle.com:1234/webservices/SOAPProvider/
plsql/ar_invoice_api_pub/APPS_XX_BPEL_CREATE_SINGLE_INVOICE_AR_INVOICE_A
PI_PUB-24CREATE_SIN.xsd"/>
      </schema>
    ...

```

In addition to message schema locations and schema elements that help to define Web messages, the *Types* element can also take a complex data type as input.

For example, the Responsibility, Responsibility Application, Security Group, NLS Language, and Organization ID complex types listed under the "SOAHeader" as shown below are used in passing values that would be used to set applications context during service execution.

```

...
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"

  targetNamespace="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_in
voice_api_pub/">
  <element name="SOAHeader">
    <complexType>
      <sequence>
        <element name="Responsibility" minOccurs="0" type="string"/>
        <element name="RespApplication" minOccurs="0" type="string"/>
        <element name="SecurityGroup" minOccurs="0" type="string"/>
        <element name="NLSLanguage" minOccurs="0" type="string"/>
        <element name="Org_Id" minOccurs="0" type="string" />
      </sequence>
    </complexType>
  </element>
</schema>
</types>

```

## Message

The *Message* element defines the name of the message. It consists of one or more Part elements, which describe the content of a message using *Element* or *Type* attributes.

Parts are a flexible mechanism for describing the logical abstract content of a message.

A binding may reference the name of a part in order to specify binding-specific information about the part.

The structure of this element can be like:

```
<definitions...>
  <message name="nmtoken"> *
    <part name="nmtoken" element="qname"? type="qname"? />
  </message>
</definitions>
```

A typical document-style Web service could have a *header* and *body* part in the input message and output message as well. For example, the *Message* element for the Invoice Creation Web service appears:

```
<message name="CREATE_INVOICE_Input_Msg">
  <part name="header" element="tns:SOAHeader"/>
  <part name="body" element="tns1:InputParameters"/>
</message>
<message name="CREATE_INVOICE_Output_Msg">
  <part name="body" element="tns1:OutputParameters"/>
</message>
<message name="CREATE_SINGLE_INVOICE_Input_Msg">
  <part name="header" element="tns:SOAHeader"/>
  <part name="body" element="tns2:InputParameters"/>
</message>
<message name="CREATE_SINGLE_INVOICE_Output_Msg">
  <part name="body" element="tns2:InputParameters"/>
</message>
```

Each message defined by the associated schema includes input message and output message parts. For example, the Invoice Creation Web service has two functions:

- **CREATE\_INVOICE**

The input message of this function which has all its parameter is defined by **CREATE\_INVOICE\_Input\_Msg**.

The output message of this function which gives its result is defined by **CREATE\_INVOICE\_Output\_Msg**.

The schema of input and output messages is defined in the APPS\_XX\_BPEL\_CREATE\_INVOICE\_AR\_INVOICE\_API\_PUB-24CREATE\_INV.xsd.

- **CREATE\_SINGLE\_INVOICE**

The input message of this function which has all its parameter is defined by **CREATE\_SINGLE\_INVOICE\_Input\_Msg**.

The output message of this function which gives its result is defined by **CREATE\_SINGLE\_INVOICE\_Output\_Msg**.

The schema of input and output messages is defined in the APPS\_XX\_BPEL\_CREATE\_SINGLE\_INVOICE\_AR\_INVOICE\_API\_PUB-24CREATE\_INV.xsd

The value of body part of each message will be set as SOAP body; the value of header

part will be set in the SOAP header which is required for Web service authorization.  
For more information, see Understanding Web Service Input Message Parts, page 11-21  
.

## PortType

The *portType* element combines multiple message elements to form a complete one-way or round-trip operation supported by a Web service.

For example, a *portType* can combine one request (input message element) and one response (output message element) message into a single request/ response operation for the synchronous request - response operation, most commonly used in SOAP services.

If it is for one-way operation, then the operation would contain an Input element only.

The structure of this element can be like:

```
<wsdl:definitions...>
  <wsdl:portType name="nmtoken">*
    <operation name="nmtoken"/>
      <wsdl:input name="nmtoken"? message="qname">?
    </wsdl:input>
      <wsdl:output name="nmtoken"? message="qname">?
    </wsdl:output>
      <wsdl:fault name="nmtoken"? message="qname">?
    </wsdl:fault>
  </wsdl:operation>
</wsdl:porttype>
</wsdl:definitions>
```

**Note:** An optional Fault element can be used for error handling in both request-response and solicit response Operation models. This feature is not supported in this release.

In this Invoice Creation Web service example, corresponding to above two functions, AR\_INVOICE\_API\_PUB\_PortType has the following two operations:

- CREATE\_INVOICE
  - Input: CREATE\_INVOICE\_Input\_Msg
  - Output: CREATE\_INVOICE\_Output\_Msg
- CREATE\_SINGLE\_INVOICE
  - Input: CREATE\_SINGLE\_INVOICE\_Input\_Msg
  - Output: CREATE\_SINGLE\_INVOICE\_Output\_Msg

```

<portType name="AR_INVOICE_API_PUB_PortType">
  <operation name="CREATE_INVOICE">
    <input name="tns:CREATE_INVOICE_Input_Msg" />
    <output name="tns:CREATE_INVOICE_Output_Msg" />
  </operation>
  <operation name="CREATE_SINGLE_INVOICE">
    <input name="tns:CREATE_SINGLE_INVOICE_Input_Msg" />
    <output name="tns:CREATE_SINGLE_INVOICE_Output_Msg" />
  </operation>
</portType>

```

## Binding

A *binding* defines message format and protocol details for operations and messages defined by a particular *portType*. It provides specific details on how a *portType* operation will actually be transmitted over the Web. Bindings can be made available through multiple transports, including HTTP GET, HTTP POST, or SOAP.

A *port* defines an individual endpoint by specifying a single address for a binding.

The structure of this element can be like:

```

<wsdl:definitions...>
  <wsdl:binding name="nmtoken" type="qname">*
    <wsdl:operation name="nmtoken"/>
    <wsdl:input> ?
    </wsdl:input>
    <wsdl:output>?
    </wsdl:output>
    <wsdl:fault name="nmtoken"? message="qname">?
    </wsdl:fault>
  </wsdl:operation>
</wsdl:binding>
</wsdl:definitions>

```

In the same example, the binding element as shown below describes the SOAP binding for PortType AR\_INVOICE\_API\_PUB\_PortType.

```

<binding name="AR_INVOICE_API_PUB_Binding"
type="tns:AR_INVOICE_API_PUB_PortType">
  <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="CREATE_INVOICE">
      <soap:operation

soapAction="https://host.us.oracle.com:1234/webservices/SOAPProvider/plsq
l/ar_invoice_api_pub/">
      <input>
        <soap:header message="tns:CREATE_INVOICE_Input_Msg" part="header"
use="literal" />
        <soap:body parts="body" use="literal" />
      </input>
      <output>
        <soap:body use="literal" />
      </output>
    </operation>
    <operation name="CREATE_SINGLE_INVOICE">
      <soap:operation

soapAction="https://host.us.oracle.com:1234/webservices/SOAPProvider/plsq
l/ar_invoice_api_pub/">
      <input>
        <soap:header message="tns:CREATE_SINGLE_INVOICE_Input_Msg"
part="header" use="literal" />
        <soap:body parts="body" use="literal" />
      </input>
      <output>
        <soap:body use="literal" />
      </output>
    </operation>
  </binding>

```

The binding used is always document style, SOAP over http binding. It also defines the content of SOAP header and SOAP body.

**Note:** Because it is a document-style service (`style="document"`), the request and response messages will consist of simply XML documents, instead of using the wrapper elements required for the remote procedure call (RPC-style) Web service. The `transport` attribute indicates the transport of the SOAP messages is through SOAP HTTP.

Within each operation, the `soap:operation` element indicates the binding of a specific operation (such as `CREATE_INVOICE`) to a specific SOAP implementation. The `soapAction` attribute specifies that the SOAPAction HTTP header be used for identifying the service.

The `soap:header` element allows header to be defined that is transmitted inside the Header element of the SOAP Envelope. The `SOAHeader` comprises of `Responsibility`, `RespApplication`, `SecurityGroup`, `NLSLanguage`, and `Org_Id` complex types within the `Types` element.

The `soap:body` element enables you to specify the details of the input and output messages for a specific operation.

## Service

The *service* element defines the Web service, and typically consists of one or more *Port* elements. A port defines an individual endpoint by specifying a single address for a binding.

The service binding is commonly created using SOAP.

The structure of this element can be like:

```
<wsdl:definitions...>
  <wsdl:service name="nmtoken">*
    <wsdl:port name="nmtoken" binding="qname"> *
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

In this example, the *Service* element `AR_INVOICE_API_PUB_Service` defines physical location of service endpoint where the service is hosted for the portType

`AR_INVOICE_API_PUB_PortType`.

```
<service name="AR_INVOICE_API_PUB_Service">
  <port name="AR_INVOICE_API_PUB_Port"
binding="tns:AR_INVOICE_API_PUB_Binding">
    <soap:address
location="https://host.us.oracle.com:1234/webservices/SOAPProvider/plsql/
ar_invoice_api_pub/"/>
  </port>
</service>
```

## Understanding SOAP Messages

SOAP (Simple Object Access Protocol) is a lightweight, XML-based protocol specification for exchanging structured information in the implementation of Web services in computer networks. For example, Web service provider receives SOAP requests from Web service clients to invoke Web services and also sends the corresponding SOAP responses out to the clients.

To support all integration interface types and services in Oracle E-Business Suite Integrated SOA Gateway, all SOAP messages are authenticated, authorized, and service enabled through SOA Provider except for Business Service Object services and generic XML Gateway messages that are enabled through Web Service Provider.

### SOAP Message Structure

SOAP is an XML-based protocol and acts as a building block for Web service communication. SOAP messages are contained in one of the SOAP components called *Envelope*. The SOAP envelop defines an overall framework for describing what is in a message; who should deal with it, and whether it is optional or mandatory. It consists of the following elements:

- Header (Optional)

An envelope element can optionally have a Header element. If an envelope contains a Header element, it must contain no more than one, and it must appear as the first child of the envelope. The first level child elements of the Header element are called Header Blocks.

Header blocks can be used in the following mechanisms:

- It provides a mechanism for attaching security related information targeted at a specific recipient.

For more information, see SOAP Security Header, page 2-21.

- It can be used to set applications context values required for services.

For more information, see SOAP Header for Applications Context, page 2-25.

- It can be used to populate mandatory header variables for XML Gateway inbound transactions to be completed successfully.

For more information, see SOAP Header for XML Gateway Messages, page 2-28

- Body

Every envelope element must contain exactly one Body element that holds the message. Immediate child elements of the Body element are called Body Blocks or Parts.

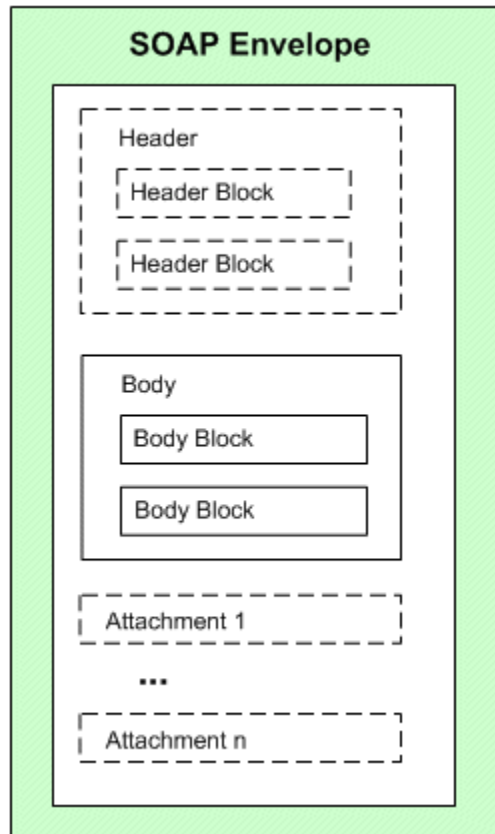
- Attachment (Optional)

A SOAP message can carry multiple attachments and these attachments can be of any type including text, binary, image, and so on.

The following diagram depicts the structure of a SOAP message.



## SOAP Message Structure



A skeleton of a SOAP message can be like:

```
<xml version="1.0">
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
    ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

## SOAP Security Header

When a SOAP request message is received through SOA Provider, the SOAP message is

passed on to OC4J Web Service Framework for authentication. The framework authenticates the SOAP message based on the specified authentication type(s) during the service deployment. The identified authentication information is embedded in the `wsse:security` Web Security headers.

### UsernameToken-based SOAP Security Header

A UsernameToken-based SOAP header should include the following `wsse:security` section:

```
<soapenv:Header>
<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
soapenv:mustUnderstand="1">
  <wsse:UsernameToken>
    <wsse:Username>Username</wsse:Username>
    <wsse:Password>Password</wsse:Password>
  </wsse:UsernameToken>
</wsse:Security>

</soapenv:Header>
```

**Note:** When a `<wsse:security>` header includes a `mustUnderstand="1"` attribute, then the receiver must generate a fault if it is unable to interpret or process security tokens contained the `<wsse:security>` header block according to the corresponding WS SOAP message security token profiles.

See A Sample Fault SOAP Response for Business Service Object, page 2-37.

A typical WS-Security header in a SOAP Request can be like:

```
<soapenv:Header>
<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
soapenv:mustUnderstand="1">
  <wsse:UsernameToken>
    <wsse:Username>myUser</wsse:Username>
    <wsse:Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">myPasswd</wsse:Password>
  </wsse:UsernameToken>
</wsse:Security>

</soapenv:Header>
```

The UsernameToken based security mechanism includes UsernameToken profile which provides username and password information in the Web service security header. Username is a clear text; password is the most sensitive part of the UsernameToken profile. In this security model, the supported password type is plain text password (or PasswordText).

The username/password in SOAP Header of a SOAP message will be passed for Web

service authentication. The username/password discussed here in `wsse:security` is the Oracle E-Business Suite username/password (or the username/password created through the Users window in defining an application user).

Passing security header elements along with the SOAP request is essential to the success of invoking Oracle E-Business Suite Web services through SOA Provider or Web Service Provider.

If these security header values are not passed, the Web service will not be authenticated and the execution of the service will be failed.

Detailed instructions on how to pass the security header along with the SOAP request when invoking an Oracle E-Business Suite Web service from a BPEL process, see *Passing Values to Security Headers*, page 3-10.

### **SAML Token-based SOAP Security Header**

Security Assertion Markup Language (SAML) is an XML-based standard for exchanging authentication and authorization data between security domains, that is, between an identity provider and a service provider.

When a Web application invokes a service that uses SAML as its authentication mechanism, this SOAP request message containing or referencing SAML assertions is received through SOA Provider and passed on to OC4J Web Service Framework for authentication. The framework authenticates the SOAP message based on the `wsse:security` Web Security headers. As part of the validation and processing of the assertions, the receiver or authentication framework must establish the relationship between the subject, claims of the referenced SAML assertions, and the entity providing the evidence to satisfy the confirmation method defined for the statements.

A trusted entity uses the sender-vouches confirmation method to ensure that it is acting on behalf of the subject of SAML statements attributed with a sender-vouches `SubjectConfirmation` element.

The following SOAP example describes a trusted entity uses the sender-vouches subject confirmation method with an associated `<ds:Signature>` element to establish its identity and to assert that it has sent the message body on behalf of the subject(s):

```

<soapenv:Envelope
xmlns:fnd="http://xmlns.oracle.com/apps/fnd/soapprovider/plsql/fnd_user_p
kg/" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  <soapenv:Header>
    <wsse:Security
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-secext-1.0.xsd">
      <ds:Signature Id="Signature-26598842"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
          <ds:Reference URI="#id-31755621">
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          </ds:Transforms>
          <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
          <ds:DigestValue>hbb/y+b3whhaFakWGO+bnkNm5/Q=</ds:DigestValue>
        </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>

jiXB+bsTfqd0uYxnaPAJcooCGb9UrKfzqSlGu/1E0nbL+sPkQQzmaB+ZKMFxUAc5pJStyeBu
3DIg
6bEXSknB3JeJaHy6UFeGKZz3ROf4WKqRvDLXsa10Ei6Id66go3goqYzYtoUA4J43MjLJbKUw
5KG/
LGBImRKABFPRP4qlAlQ=
        </ds:SignatureValue>
        <ds:KeyInfo Id="KeyId-1042529">
          <wsse:SecurityTokenReference wsu:Id="STRId-6382436"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssec
urity-utility-1.0.xsd"><wsse:KeyIdentifier

EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-so
ap-message-security-1.0#Base64Binary"

ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-
token-profile-1.0#X509SubjectKeyIdentifier">ADoNKKuduSTKTwi7jqEzCxd7JU=
</wsse:KeyIdentifier></wsse:SecurityTokenReference>
        </ds:KeyInfo></ds:Signature>
        <Assertion AssertionID="be7d9814c36381c27fefa89d8f27e126"
IssueInstant="2010-02-27T17:26:21.241Z" Issuer="www.oracle.com"
MajorVersion="1" MinorVersion="1"
xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><Conditions
NotBefore="2010-02-27T17:26:21.241Z"
NotOnOrAfter="2011-02-27T17:26:21.241Z" />
          <AuthenticationStatement
AuthenticationInstant="2010-02-27T17:26:21.241Z"
AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
            <Subject>
              <NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
NameQualifier="notRelevant">SYSADMIN</NameQualifier>
            <SubjectConfirmation>

```

```

<ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:sender-vouches</Confir
mationMethod>
  </SubjectConfirmation>
</Subject>
</AuthenticationStatement>
</Assertion>
</wsse:Security>

  <fnd:SOAHeader>
    <!--Optional:-->
    <fnd:Responsibility>UMX</fnd:Responsibility>
    <!--Optional:-->
    <fnd:RespApplication>FND</fnd:RespApplication>

  </fnd:SOAHeader>
</soapenv:Header>

  <soapenv:Body wsu:Id="id-31755621"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssec
urity-utility-1.0.xsd">
  <tes:InputParameters
xmlns:tes="http://xmlns.oracle.com/apps/fnd/soapprovider/plsql/fnd_user_p
kg/testusername/">
    <!--Optional:-->
    <tes:X_USER_NAME>AMILLER</tes:X_USER_NAME>
  </tes:InputParameters>
</soapenv:Body>
</soapenv:Envelope>

```

**Note:** SAML Token based security can be used to authenticate users in both Single Sign-On (SSO) and non-SSO enabled environments. The format of the NameIdentifier in the SAML assertion indicates if the user has been authenticated against LDAP (SSO user) or Oracle E-Business Suite FND\_USER table (for non-SSO user).

The SAML assertion in the above SOAP message is for non-SSO enabled environment. If the username in the NameIdentifier tag is of the form of LDAP DN as shown below, then the username is verified in the registered OID for SSO user.

```

<NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecif
ied"

NameQualifier="notRelevant">orclApplicationCommonName=PROD
1,cn=EBusiness,cn=Products,cn=OracleContext,dc=us,dc=orac
le,dc=com</NameIdentifier>

```

For more information about SAML Token sender-vouches based security, see SAML Sender-Vouches Token Based Security, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

## SOAP Header for Applications Context

Applications context contains many crucial elements that are used in passing values

that may be required in proper functioning of Oracle E-Business Suite Web services. For example, the context header information is required for an API transaction or a concurrent program in order for an Oracle E-Business Suite user that has sufficient privileges to run the program.

### **Applications Context in SOAHeader Part of a SOAP Request**

These context header elements defined in `SOAHeader` part of SOAP request for PL/SQL, Concurrent Program, and Java APIs for Forms services are:

- **Responsibility**  
It is the Oracle E-Business Suite application responsibility information. It accepts `responsibility_key` (such as `SYSTEM_ADMINISTRATOR`) as its value.
- **RespApplication**  
It is the responsibility application short name information. It accepts `Application Short Name` (such as `FND`) as its value.
- **SecurityGroup**  
It accepts `Security Group Key` (such as `STANDARD`) as its value.
- **NLSLanguage (optional)**  
It is an optional parameter to be passed in `SOAHeader` part of a SOAP request for PL/SQL and Concurrent Program services.  
  
If the NLS Language element is specified, SOAP requests can be consumed in the language passed. All corresponding SOAP responses and error messages can also be returned in the same language. If no language is identified, then the default language of the user will be used.
- **Org\_Id (optional for PL/SQL and Concurrent Program services)**
  - It is an optional parameter to be passed in `SOAHeader` part of a SOAP request for PL/SQL and Concurrent Program services. If a service execution is dependent on any particular organization, then you must pass the `Org_Id` element of that SOAP request.
  - `Org_Id` is a mandatory value that must be passed for the Java APIs for Forms services.

The following SOAP message shows the `SOAHeader` part highlighted in bold text:

```

<soapenv:Header>
<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
soapenv:mustUnderstand="1">
  <wsse:UsernameToken>
    <wsse:Username>myUser</wsse:Username>
    <wsse:Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">myPasswd</wsse:Password>
  </wsse:UsernameToken>
</wsse:Security><ozf:SOAHeader>
  <ozf:Responsibility>OZF_USER</ozf:Responsibility>
  <ozf:RespApplication>OZF</ozf:RespApplication>
  <ozf:SecurityGroup>STANDARD</ozf:SecurityGroup>
  <ozf:NLSLanguage>AMERICAN</ozf:NLSLanguage>
  <ozf:Org_Id>204</ozf:Org_Id>
</ozf:SOAHeader>
</soapenv:Header>

```

### Applications Context in **ServiceBean\_Header** Part of a SOAP Request

These context header elements defined in **ServiceBean\_Header** part of SOAP requests for Business Service Object services are:

- **RESPONSIBILITY\_NAME**  
It is the Oracle E-Business Suite application responsibility information. It can accept both the name (**Responsibility\_Name**, such as **System Administrator**) and the key (in the format of {key}responsibility\_key, such as {key}SYSTEM\_ADMINISTRATOR) as its values.
- **RESPONSIBILITY\_APPL\_NAME**  
It is the responsibility application short name information. It accepts Application Short Name (such as **FND**) as its value.
- **SECURITY\_GROUP\_NAME**  
It accepts Security Group Key (such as **STANDARD**) as its value.
- **NLSLanguage (optional)**  
It is an optional parameter to be passed in **ServiceBean\_Header** part of a SOAP request for Business Service Object service.  
If the NLS Language element is specified (such as **AMERICAN**), SOAP requests can be consumed in the language passed. All corresponding SOAP responses and error messages can also be returned in the same language. If no language is identified, then the default language of the user will be used.
- **Org\_Id (optional)**  
It is an optional parameter to be passed in **ServiceBean\_Header** part of a SOAP request for Business Service Object service.

If a service execution is dependent on any particular organization, then you must pass the `Org_Id` element of that SOAP request.

The following SOAP request example includes the `ServiceBean_Header` part highlighted in bold text for business service object:

```
<soapenv:Envelope
xmlns:ser="http://xmlns.oracle.com/apps/fnd/ServiceBean"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://xmlns.oracle.com/apps/fnd/rep/ws">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-secext-1.0.xsd">
      <wsse:UsernameToken wsu:Id="UsernameToken-22948433"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssec
urity-utility-1.0.xsd">
        <wsse:Username>sysadmin</wsse:Username>
        <wsse:Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-t
oken-profile-1.0#PasswordText">sysadmin</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security><ser:ServiceBean_Header>
      <ser:RESPONSIBILITY_NAME>System
Administrator</ser:RESPONSIBILITY_NAME>

      <ser:RESPONSIBILITY_APPL_NAME>sysadmin</ser:RESPONSIBILITY_APPL_NAME>
      <ser:SECURITY_GROUP_NAME>standard</ser:SECURITY_GROUP_NAME>
      <ser:NLS_LANGUAGE>american</ser:NLS_LANGUAGE>
      <ser:ORG_ID>202</ser:ORG_ID>
    </ser:ServiceBean_Header>
  </soapenv:Header>
  <soapenv:Body>
    <ws:IntegrationRepositoryService_GetInterfaceByType>
      <interfaceType>XMLGATEWAY</interfaceType>
    </ws:IntegrationRepositoryService_GetInterfaceByType>
  </soapenv:Body>
</soapenv:Envelope>
```

## SOAP Header for XML Gateway Messages

The SOAP header part can also be used to populate header variables for XML Gateway inbound transactions to be completed successfully. These XML Gateway header parameters defined in the `SOAHeader` (through SOA Provider) or `XMLGateway_Header` (through Web Service Provider) part of a SOAP Request are described in the following table:

The following code snippet shows the `SOAHeader` part of a SOAP request for an XML Gateway inbound message through SOA Provider:



```

<soapenv: Envelope
xmlns:ecx="http://xmlns.oracle.com/apps/ecx/soapprovider/xmlgateway/ecx__
cbodi/"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sys="http://xmlns.oracle.com/xdb/SYSTEM">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1"

xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-secext-1.0.xsd">
      <wsse:UsernameToken wsu:Id="UsernameToken-10586449"

xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssec
urity-utility-1.0.xsd">
        <wsse:Username>SYSADMIN</wsse:Username>
        <wsse:Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-t
oken-profile-1.0#PasswordText">SYSADMIN</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  <ecx:SOAHeader>
    <sys:ECXMSG>
      <MESSAGE_TYPE></MESSAGE_TYPE>
      <MESSAGE_STANDARD></MESSAGE_STANDARD>
      <TRANSACTION_TYPE></TRANSACTION_TYPE>
      <TRANSACTION_SUBTYPE></TRANSACTION_SUBTYPE>
      <DOCUMENT_NUMBER></DOCUMENT_NUMBER>
      <PARTYID></PARTYID>
      <PARTY_SITE_ID></PARTY_SITE_ID>
      <PARTY_TYPE></PARTY_TYPE>
      <PROTOCOL_TYPE></PROTOCOL_TYPE>
      <PROTOCOL_ADDRESS></PROTOCOL_ADDRESS>
      <USERNAME></USERNAME>
      <PASSWORD></PASSWORD>
      <ATTRIBUTE1></ATTRIBUTE1>
      <ATTRIBUTE2></ATTRIBUTE2>
      <ATTRIBUTE3></ATTRIBUTE3>
      <ATTRIBUTE4></ATTRIBUTE4>
      <ATTRIBUTE5></ATTRIBUTE5>
    </sys:ECXMSG>
  </ecx:SOAHeader>
</soapenv:Header>

```

The following table describes the XML Gateway header information in SOAHeader part of a SOAP request:

#### ***XML Gateway Header Information in SOAHeader Part of a SOAP Request***

Attribute	Description
MESSAGE_TYPE	Payload message format. This defaults to XML. Oracle XML Gateway currently supports only XML.

Attribute	Description
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 is defined in the Trading Partner table.
PROTOCOL_ADDRESS	Transmission address is defined in the Trading Partner table.
USERNAME	USERNAME is defined in the Trading Partner table.
PASSWORD	The password associated with the USERNAME is 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.
ATTRIBUTE1	This parameter may be defined by the base application.

Attribute	Description
ATTRIBUTE2	This parameter may be defined by the base application.
ATTRIBUTE3	<p>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.</p> <p><b>Note:</b> For more information, see <i>Destination Trading Partner Location Code</i> in the <i>Oracle XML Gateway User's Guide</i>.</p>
ATTRIBUTE4	This parameter may be defined by the base application.
ATTRIBUTE5	This parameter may be defined by the base application.

- The Username and Password in `SOAHeader` here is the username and password associated with trading partner setup.  
The Username and Password in `<wsse:Security>` discussed earlier is the Oracle E-Business Suite username/password (or the username/password created through the Users window in defining an application user).
- The `PARTYID` and `PARTY_TYPE` parameters are not used.

The following code snippet shows the `XMLGateway_Header` part of a SOAP request through Web Service Provider:

```

<soap:Envelope>
  <soap:Header>
    ...
    <ns1:XMLGateway_Header
      xmlns:ns1="http://xmlns.oracle.com/apps/fnd/XMLGateway
      soapenv:mustUnderstand="0">
      <ns1:MESSAGE_TYPE>XML</ns1:MESSAGE_TYPE>
      <ns1:MESSAGE_STANDARD>OAG</ns1:MESSAGE_STANDARD>
      <ns1:TRANSACTION_TYPE>PO</ns1:TRANSACTION_TYPE>
      <ns1:TRANSACTION_SUBTYPE>PROCESS</ns1:TRANSACTION_SUBTYPE>
      <ns1:DOCUMENT_NUMBER>123</ns1:DOCUMENT_NUMBER>
      <ns1:PARTY_SITE_ID>4444</ns1:PARTY_SITE_ID>
    </ns1:XMLGateway_Header>
  </soap:Header>
  ...
</soap:Envelope>

```

The following table describes the XML Gateway header information in XMLGateway\_Header part of a SOAP request:

#### ***XMLGateway\_Header Part of a SOAP Request***

Parameter Name	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.

Parameter Name	Description
DOCUMENT_NUMBER	The document identifier used to identify the transaction, such as a purchase order or invoice number. This parameter is not used by the XML Gateway, but it may be passed on inbound messages.
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.

## Examples of SOAP Messages Through SOA Provider

To better understand SOAP request and response messages received through SOA Provider, the following sample SOAP messages are described in this section:

- A Sample SOAP Request, page 2-33
- A Sample SOAP Response, page 2-35
- A Sample Fault SOAP Response, page 2-35

### A Sample SOAP Request

The following example shows a SOAP request for a PL/SQL service:

```

<soapenv:Envelope xmlns:ser="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-secext-1.0.xsd"
xmlns:ozf="http://xmlns.oracle.com/apps/ozf/soapprovider/plsql/ozf_sd_req
uest_pub/"
xmlns:cre="http://xmlns.oracle.com/apps/ozf/soapprovider/plsql/ozf_sd_req
uest_pub/create_sd_request/">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1">
      <wsse:UsernameToken>
        <wsse:Username>trademgr</wsse:Username>
        <wsse:Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-t
oken-profile-1.0#PasswordText">welcome</wsse:Password>
        </wsse:UsernameToken>
      </wsse:Security>
      <ozf:SOAHeader>
        <ozf:Responsibility>OZF_USER</ozf:Responsibility>
        <ozf:RespApplication>OZF</ozf:RespApplication>
        <ozf:SecurityGroupE>STANDARD</ozf:SecurityGroup>
        <ozf:NLSLanguage>AMERICAN</ozf:NLSLanguage>
        <ozf:Org_Id>204</ozf:Org_Id>
      </ozf:SOAHeader>
    </soapenv:Header>
    <soapenv:Body>
      <cre:InputParameters>
        <cre:P_API_VERSION_NUMBER>1.0</cre:P_API_VERSION_NUMBER>
        <cre:P_INIT_MSG_LIST>T</cre:P_INIT_MSG_LIST>
        <cre:P_COMMIT>F</cre:P_COMMIT>
        <cre:P_VALIDATION_LEVEL>100</cre:P_VALIDATION_LEVEL>
        <cre:P_SDR_HDR_REC>
          <cre:REQUEST_NUMBER>SDR-CREATE-A1</cre:REQUEST_NUMBER>

        <cre:REQUEST_START_DATE>2008-08-18T12:00:00</cre:REQUEST_START_DATE>
          <cre:REQUEST_END_DATE>2008-10-18T12:00:00</cre:REQUEST_END_DATE>
          <cre:USER_STATUS_ID>1701</cre:USER_STATUS_ID>
          <cre:REQUEST_OUTCOME>IN_PROGRESS</cre:REQUEST_OUTCOME>
          <cre:REQUEST_CURRENCY_CODE>USD</cre:REQUEST_CURRENCY_CODE>
          <cre:SUPPLIER_ID>601</cre:SUPPLIER_ID>
          <cre:SUPPLIER_SITE_ID>1415</cre:SUPPLIER_SITE_ID>
          <cre:REQUESTOR_ID>100001499</cre:REQUESTOR_ID>
          <cre:ASSIGNEE_RESOURCE_ID>100001499</cre:ASSIGNEE_RESOURCE_ID>
          <cre:ORG_ID>204</cre:ORG_ID>
          <cre:ACCRUAL_TYPE>SUPPLIER</cre:ACCRUAL_TYPE>
          <cre:REQUEST_DESCRIPTION>Create</cre:REQUEST_DESCRIPTION>

        <cre:SUPPLIER_CONTACT_EMAIL_ADDRESS>sdr.supplier@testing.com</cre:SUPPLI
ER_CONTACT_EMAIL_ADDRESS>

        <cre:SUPPLIER_CONTACT_PHONE_NUMBER>2255</cre:SUPPLIER_CONTACT_PHONE_NUMB
ER>
          <cre:REQUEST_TYPE_SETUP_ID>400</cre:REQUEST_TYPE_SETUP_ID>
          <cre:REQUEST_BASIS>Y</cre:REQUEST_BASIS>
          <cre:USER_ID>1002795</cre:USER_ID>
        </cre:P_SDR_HDR_REC>
        <cre:P_SDR_LINES_TBL>
          <cre:P_SDR_LINES_TBL_ITEM>
            <cre:PRODUCT_CONTEXT>PRODUCT</cre:PRODUCT_CONTEXT>
            ...
          </cre:P_SDR_LINES_TBL_ITEM>
        </cre:P_SDR_LINES_TBL>
      </cre:InputParameters>
    </soapenv:Body>
  </soapenv:Envelope>

```

```

</cre:P_SDR_LINES_TBL>
  <cre:P_SDR_CUST_TBL>
    ...
  </cre:P_SDR_CUST_TBL>
</cre:InputParameters>>
</soapenv:Body>
</soapenv:Envelope>

```

### A Sample SOAP Response

The following example shows a SOAP response for a PL/SQL service:

```

<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header/>
  <env:Body>
    <OutputParameters
      xmlns="http://xmlns.oracle.com/apps/ozf/soapprovider/plsql/ozf_sd_request
      _pub/create_sd_request/">
      <X_RETURN_STATUS>S</X_RETURN_STATUS>
      <X_MSG_COUNT>23</X_MSG_COUNT>
      <X_MSG_DATA xsi:nil="true"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"/>
      <X_REQUEST_HEADER_ID>162</X_REQUEST_HEADER_ID>
    </OutputParameters>
  </env:Body>
</env:Envelope>

```

### A Sample Fault SOAP Response

The SOAP Fault element is used to carry error and status information within a SOAP message.

For example, the following fault response message indicates that the service is not deployed:

```

<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header/>
  <env:Body>
    <Fault xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
      <faultcode xmlns="">SOAP-ENV:Server</faultcode>
      <faultstring xmlns="">Service is not deployed.</faultstring>
    </env:Fault>
  </env:Body>
</env:Envelope>

```

## Examples of SOAP Messages Through Web Service Provider

To better understand SOAP request and response messages for business service object exposed to Web services through Web Service Provider, the following sample SOAP messages are described in this section:

- A Sample SOAP Request for Business Service Object, page 2-36
- A Sample SOAP Response for Business Service Object, page 2-36
- A Sample Fault SOAP Response for Business Service Object, page 2-37

### A Sample SOAP Request for Business Service Object

The following example shows a valid SOAP request for business service object:

```
<soapenv:Envelope
xmlns:ser="http://xmlns.oracle.com/apps/fnd/ServiceBean"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ws="http://xmlns.oracle.com/apps/fnd/rep/ws">
  <soapenv:Header>
    <wsse:Security soapenv:mustUnderstand="1"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wsse
curity-secext-1.0.xsd">
      <wsse:UsernameToken wsu:Id="UsernameToken-22948433"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssec
urity-utility-1.0.xsd">
        <wsse:Username>sysadmin</wsse:Username>
        <wsse:Password
Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-t
oken-profile-1.0#PasswordText">sysadmin</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <ser:ServiceBean_Header>
    <ser:RESPONSIBILITY_NAME>System
Administrator</ser:RESPONSIBILITY_NAME>

    <ser:RESPONSIBILITY_APPL_NAME>sysadmin</ser:RESPONSIBILITY_APPL_NAME>
    <ser:SECURITY_GROUP_NAME>standard</ser:SECURITY_GROUP_NAME>
    <ser:NLS_LANGUAGE>american</ser:NLS_LANGUAGE>
    <ser:ORG_ID>202</ser:ORG_ID>
  </ser:ServiceBean_Header>
</soapenv:Header>
  <soapenv:Body>
    <ws:IntegrationRepositoryService_GetInterfaceByType>
      <interfaceType>XMLGATEWAY</interfaceType>
    </ws:IntegrationRepositoryService_GetInterfaceByType>
  </soapenv:Body>
</soapenv:Envelope>
```

### A Sample SOAP Response for Business Service Object

The following example shows a valid SOAP response for business service object:



```

<soapenv:Envelope xmlns:env=http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header/>
  <env:Body>
    <oans:IntegrationRepositoryService_GetInterfaceByType_Response
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:oans="http://xmlns.oracle.com/apps/fnd/rep/ws">
      <interfaceClass>
        <ClassId>906</ClassId>
        <ClassName>XMLGATEWAY:CLN:SHIP_ORDER_CONFIRM_OUT</ClassName>
        <IrepName>CLN:SHIP_ORDER_CONFIRM_OUT</IrepName>
        <SecurityGroupId xsi:nil="true"/>
        <ClassType>XMLGATEWAY</ClassType>
        <ProductCode>cln</ProductCode>
        <ImplementationName xsi:nil="true"/>
        <DeployedFlag>N</DeployedFlag>
        <GeneratedFlag>N</GeneratedFlag>
        <CompatibilityFlag>N</CompatibilityFlag>
        <AssocClassId xsi:nil="true"/>
        <ScopeType>PUBLIC</ScopeType>
        <LifecycleMode>ACTIVE</LifecycleMode>
        <SourceFileProduct>CLN</SourceFileProduct>
        ...
      </interfaceClass>
      <InterfaceFunction>
        ...
      </InterfaceFunction>
    </oans:IntegrationRepositoryService_GetInterfaceByType_Response>
  </env:Body>
</env:Envelope>

```

### A Sample Fault SOAP Response for Business Service Object

The SOAP `Fault` element is used to carry error and status information within a SOAP message.

For example, if a SOAP request message contains invalid header information or the header is missing from the request, then `Fault` element appears as a body entry in the response message as shown below for business service object:

```

<env:Envelope xmlns:env=http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header/>
  <env:Body>
    <Fault xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
      <faultcode>SOAP-ENV:Client</faultcode>
      <faultstring>InvalidHeader: Invalid or missing header in
request.</faultstring>
    </env:Fault>
  </env:Body>
</env:Envelope>

```



---

## Using PL/SQL APIs as Web Services

### Overview

Oracle E-Business Suite Integrated SOA Gateway allows you to use PL/SQL application programming interfaces (APIs) to insert or update data in Oracle E-Business Suite. APIs are stored procedures that let you update or retrieve data from Oracle E-Business Suite.

Once a PL/SQL API interface definition is exposed as a Web service representing in WSDL URL, the generated Web service can be deployed from the Oracle Integration Repository to Oracle E-Business Suite application server. Services can then be exposed to customers through service provider and invoked through any of the Web service clients or orchestration tool including Oracle JDeveloper, Apache Axis, .NET Web Service Client, Oracle BPEL Process Manager, and Oracle Enterprise Service Bus (ESB).

For example, these deployed Web services can be orchestrated into a meaningful business process with service endpoints using a BPEL language. At run time, the BPEL process can be deployed to Oracle BPEL server or a third party BPEL server that can be consumed by customers.

To better understand how each individual Web service represented in WSDL URL can be used in inserting or updating application data, detailed design-time and run-time tasks in creating and deploying a BPEL process are discussed in this chapter. For the example described in the following sections, we use Oracle JDeveloper 10.1.3.3.0 as a design-time tool to create the BPEL process and use Oracle SOA Suite BPEL server 10.1.3.3.0 for the process deployment.

### Using PL/SQL WSDLs at Design Time

#### BPEL Process Scenario

Take PL/SQL Supplier Ship and Debit Request API `OZF_SD_REQUEST_PUB` as an example to explain the BPEL process creation.

When the creation of a ship and debit request is received, the creation information including input ship and debit payload will be read and passed to create a ship and

debit request. Once the request is created, the request number will then be returned to the requestor.

If the BPEL process is successfully executed after deployment, you should find a ship and debit request is created in the Oracle Order Management. The request number should be the same as the payload input value.

### **Prerequisites to Create a BPEL Process Using a PL/SQL Web Service**

Before performing the design-time tasks for PL/SQL Web services, you need to ensure the following tasks are in place:

**Note:** Before generating the Web service for a selected interface, you can also create a security grant to a specific user (such as "TRADEMGR") or user group if necessary to ensure the user has the access privilege to the interface.

- An integration repository administrator needs to successfully generate and deploy a Web service to the application server.
- An integration developer needs to locate and record the deployed WSDL URL for the PL/SQL exposed as a Web service.
- SOAHeader variables need to be populated for Web service authorization.

Please note that certain PL/SQL APIs exposed from Oracle E-Business Suite Integrated SOA Gateway take record types as input. Such APIs expect default values to be populated for parameters within these record types for successful execution.

The default values are `FND_API.G_MISS_CHAR` for characters, `FND_API.G_MISS_DATE` for dates, and `FND_API.G_MISS_NUM` for numbers. Oracle E-Business Suite Integrated SOA Gateway can default these values when the parameters within the record type are passed as nil values, for example, as shown below:

```
<PRICE_LIST_REC>
<ATTRIBUTE1 xsi:nil="true"/>
<ATTRIBUTE2 xsi:nil="true"/>
<ATTRIBUTE3 xsi:nil="true"/>
...
</PRICE_LIST_REC>
```

### ***Deploying PL/SQL WSDL URL***

An integration repository administrators must first create a Web service for a selected interface definition, and then deploy the service from Oracle Integration Repository to the application server.

For example, the administrator must perform the following steps before letting the integration developers use the deployed WSDL in creating a BPEL process:

1. To generate a Web service, locate the interface definition first (such as a PL/SQL

interface OZF\_SD\_REQUEST\_PUB) and click **Generate WSDL** in the interface details page.

Once the service is successfully generated, the Web Service - SOA Provider region appears in the interface details page. For detailed instruction on how to generate a Web service, see *Generating Web Services, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

2. To deploy a generated Web service, select at least one authentication type and click **Deploy** in the Web Service - SOA Provider region of the interface details page to deploy the service.

Once the service is successfully deployed, the selected authentication type(s) will be displayed along with 'Deployed' Web Service Status. For more information on securing Web services with authentication types, see *Managing Web Service Security, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

For detailed instruction on how to deploy a Web service, see *Deploying, Undeploying, and Redeploying Web Services, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

### ***Searching and Recording WSDL URL***

Apart from the required tasks performed by the administrators, an integration developer also needs to log on to the system to locate and record the deployed Web service WSDL URL for the interface that needs to be orchestrated into a meaningful business process in Oracle JDeveloper using BPEL language.

This can be done by clicking the **View WSDL** link in the interface details page to open a new window. Copy the WSDL URL from the new window. This URL will be used later in creating a partner link for the interface exposed as a Web service during the BPEL process creation at design time.

## Viewing and Recording a Deployed WSDL URL

The screenshot displays the Oracle Integration Repository web application. The header includes the Oracle logo and navigation links: Home, Logout, Preferences, Help, and Diagnostics. The main content area is titled "Integration Repository" and shows the path "Integration Repository > PLSQL Interface : OZF\_SD\_REQUEST Public API".

Metadata for the interface is listed:

- Internal Name: OZF\_SD\_REQUEST\_PUB
- Type: PL/SQL
- Product: Trade Management
- Status: Active
- Business Entity: [Supplier Ship and Debit Request](#)
- Scope: Public
- Interface Source: Oracle

The "Full Description" section states: "This package can be used to Create, Update and Copy Ship & Debit Request".

The "Web Service - SOA Provider" section shows the Web Service Status as [Deployed](#) and the WSDL as [View WSDL](#). The Authentication Type is set to ☒ SAML Token (Sender Vouches).

The "Source Information" section lists:

- Source File: [patch/115/sql/ozfspdrrs.pls](#)
- Source Version: 120.10.12010000.2
- Source Product: OZF

The "Procedures and Functions" section includes a "Select Object" dropdown and a table of procedures:

Select Details	Name	Internal Name	Status	Description
<input type="checkbox"/> <a href="#">Show Create SDR</a>	CREATE_SD_REQUEST	CREATE_SD_REQUEST	Active	This procedure creates a new Ship & Debit Request.
<input type="checkbox"/> <a href="#">Show Update SD Request</a>	UPDATE_SD_REQUEST	UPDATE_SD_REQUEST	Active	This Procedure updates existinf Ship & Debit Request
<input type="checkbox"/> <a href="#">Show copy sd request</a>	COPY_SD_REQUEST	COPY_SD_REQUEST	Active	This procedure Copies existing Ship & Debit Request.

How to search for an interface and review the interface details, see [Searching and Viewing Integration Interfaces](#), page 2-1.

### Setting Variables in SOAHeader for SOAP Request

You must populate certain variables in the BPEL process for SOAHeader elements to pass values that would be used to set application context during service execution. These SOAHeader elements for PL/SQL interface type are *Responsibility*, *RespApplication*, *SecurityGroup*, *NLSLanguage*, and *Org\_Id*.

**Note:** The user information is defined by the `wsseUsername` property passed within the security headers. Detailed instructions on how to pass the security headers along with the SOAP request, see [Passing Values to Security Headers](#), page 3-10.

The expected values for these elements are described in the following table:

### **Header Variables and Expected Values for PL/SQL Interface Type**

Element Name	Expected Value
Responsibility	responsibility_key (such as "SYSTEM_ADMINISTRATOR")
RespApplication	Application Short Name (such as "FND")
SecurityGroup	Security Group Key (such as "STANDARD")
NLSLanguage	NLS Language (such as "AMERICAN")
Org_Id	Org Id (such as "202")

**Note:** NLS Language and Org\_Id are optional values to be passed.

- If the NLS Language element is specified, SOAP requests can be consumed in the language passed. All corresponding SOAP responses and error messages can also be returned in the same language. If no language is identified, then the default language of the user will be used.
- If a service execution is dependent on any particular organization, then you must pass the Org\_Id element of that SOAP request.

The context information can be specified by configuring an Assign activity before the Invoke activity in the BPEL PM.

Detailed information on how to set SOAHeader for the SOAP request, see Assigning SOAHeader Parameters, page 3-25.

### **BPEL Process Creation Flow**

Based on the single invoice creation scenario, the following design-time tasks are discussed in this chapter:

1. Create a new BPEL project, page 3-6

Use this step to create a new BPEL project called `ShipDebitRequest.bpel` using an Synchronous BPEL Process template. This automatically creates two dummy activities - Receive and Reply - to receive input from a third party application and to reply output of the BPEL process back to the request application.

2. Create a Partner Link, page 3-8

Use this step to create a ship and debit request in Oracle Order Management by using the Supplier Ship and Debit Request API `OZF_SD_REQUEST_PUB` exposed as Web service.

3. Add a Partner Link for File Adapter, page 3-12

Use this step to synchronous read invoice header details passed from the first Assign activity.

4. Add Invoke activities, page 3-21

Use this step to configure two Invoke activities in order to:

- Point to the File Adapter to synchronous read invoice header details that is passed from the first Assign activity.
- Point to the `OZF_SD_REQUEST_PUB` partner link to initiate the request creation with payload and transaction details received from the Assign activities.

5. Add Assign activities, page 3-25

Use this step to configure Assign activities in order to pass request header details, payload information and request number to appropriate Invoke activities to facilitate the request creation. At the end, pass the request number to the request application through the dummy Reply activity.

For general information and basic concept of a BPEL process, see Understanding BPEL Business Processes, page D-1 and *Oracle BPEL Process Manager Developer's Guide* for details.

## Creating a New BPEL Project

Use this step to create a new BPEL project that will contain various BPEL process activities.

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



### Entering BPEL Project Information

The BPEL Project Creation Wizard allows you to create a project in which you can design a business process based on the BPEL (Business Process Execution Language) standard.

Please specify the process name and project settings below.

Name:

Namespace:

☒ **Use Default Project Settings**

Project Name:

Project Directory:

Template:

Help < Back Next > Finish Cancel

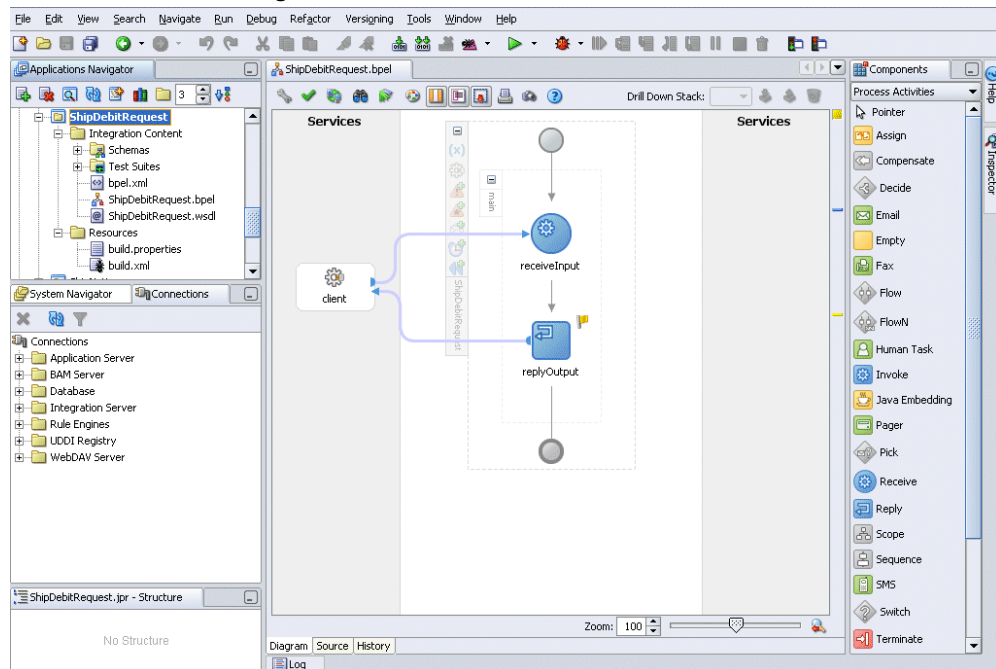
7. In the **Name** field, enter a descriptive name such as `ShipDebitRequest`.

**Note:** SOA Provider does not support service creation for PL/SQL stored procedures or packages which have '\$' character in parameter type names. The presence of \$ in the name would cause the XSD generation to fail.

8. From the Template list, select **Synchronous BPEL Process**. Select **Use Default Project Settings**.
9. Use the default input and output schema elements in the Input/Output Elements dialog box.
10. Click **Finish**.

A new synchronous BPEL process is created with the Receive and Reply activities. The required source files including `bpel.xml`, using the name you specified (for example `ShipDebitRequest.bpel`) are also generated.

## New BPEL Process Diagram

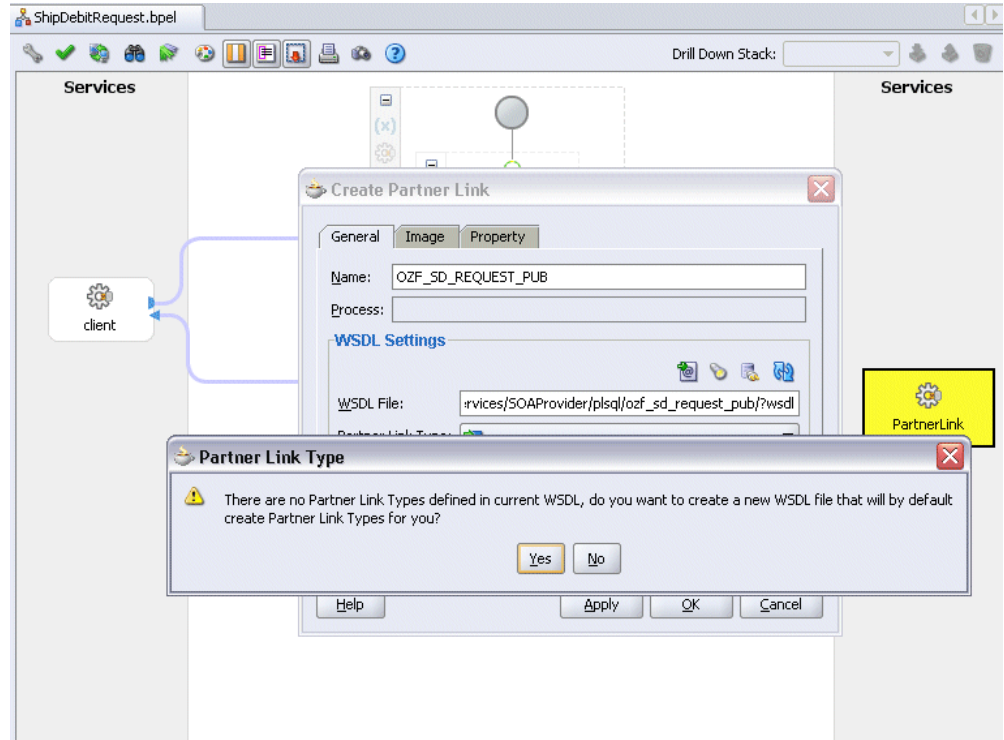


## Creating a Partner Link for the Web Service

Use this step to create a Partner Link called `OZF_SD_REQUEST_PUB`.

**To create a partner link for `OZF_SD_REQUEST_PUB` Web service:**

1. In JDeveloper BPEL Designer, drag and drop the **PartnerLink** service from the Component Palette into the Partner Link border area of the process diagram. The Service Name dialog box appears.
2. Copy the WSDL URL corresponding to the `OZF_SD_REQUEST_PUB` service that you recorded earlier from the Integration Repository, and paste it in the WSDL File field.
3. A Partner Link Type message dialog box appears asking whether you want the system to create a new WSDL file that will by default create partner link types for you.



Click **Yes** to have the Partner Name value populated automatically. The name is defaulted to OZF\_SD\_REQUEST\_PUB.

Select Partner Role and My Role fields from the drop-down lists.

### Create Partner Link

**Create Partner Link**

General Image Property

Name: OZF\_SD\_REQUEST\_PUB

Process:

**WSDL Settings**

WSDL File: nipDebitRequest/bpel/OZF\_SD\_REQUEST\_PUB1.wsdl

Partner Link Type: OZF\_SD\_REQUEST\_PUB\_PortType\_PL

Partner Role: OZF\_SD\_REQUEST\_PUB\_PortType\_Role

My Role: OZF\_SD\_REQUEST\_PUB\_PortType\_Role

Help Apply OK Cancel

Click **Apply**.

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.

#### 4. Passing Values to Security Headers

Select the Property tab and click the **Create Property** icon to select the following properties from the property name drop-down list in order to pass the security headers along with the SOAP request:

- wsseUsername

Specify the username to be passed in the Property Value box.

- wssePassword

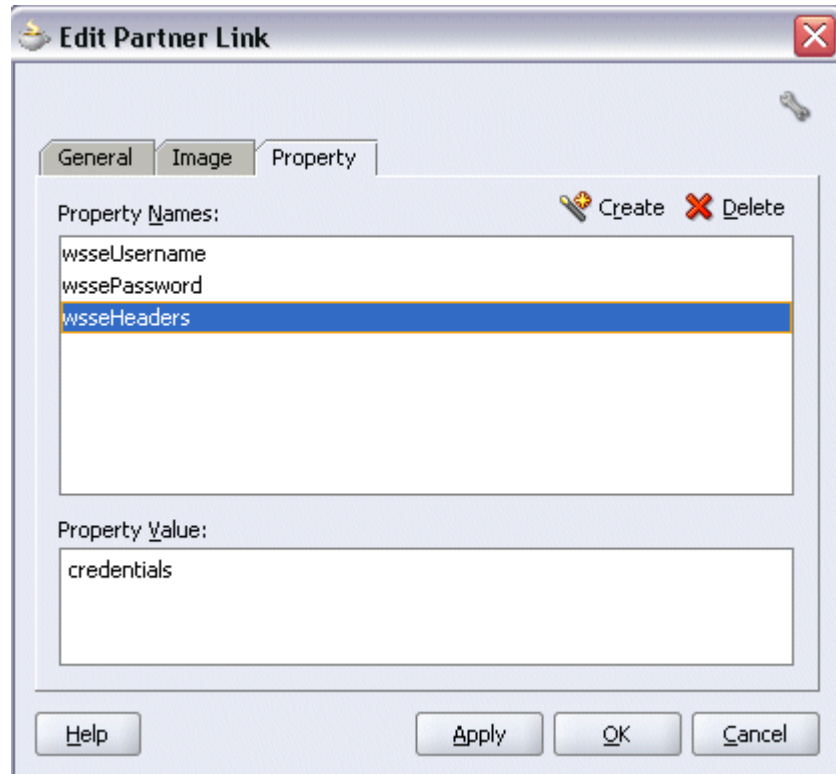
Specify the corresponding password for the username to be passed in the Property Value box.

- wsseHeaders

Enter `credentials` as the property value.

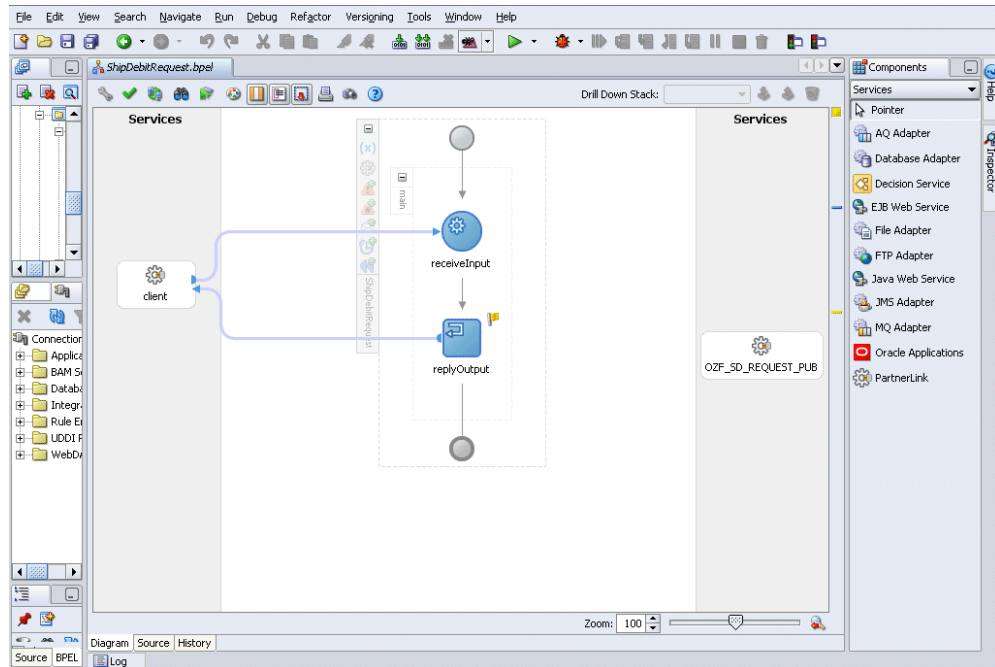
Click **Apply** to save the selected property values.

### Adding Properties



5. Click **OK** to complete the partner link configuration.

### Viewing a Partner Link from BPEL Diagram



Partner Link `OZF_SD_REQUEST_PUB` is added to the Services section in the BPEL process diagram.

### Adding a Partner Link for File Adapter

Use this step to configure a BPEL process by reading current contents of a file.

**To add a Partner Link for File Adapter to Read Payload:**

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 welcome page appears.
2. Click **Next**. The Service Name dialog box appears.
3. Enter a name for the file adapter service such as `ReadPayload`. You can add an optional description of the service.
4. Click **Next**. The Operation dialog box appears.



### Specifying the Operation

**Adapter Configuration Wizard - Step 2 of 5: Operation**

The File Adapter supports three operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, and a Synchronous Read File operation that reads the current contents of a file. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type:

☐ Read File

☐ Write File

☒ Synchronous Read File

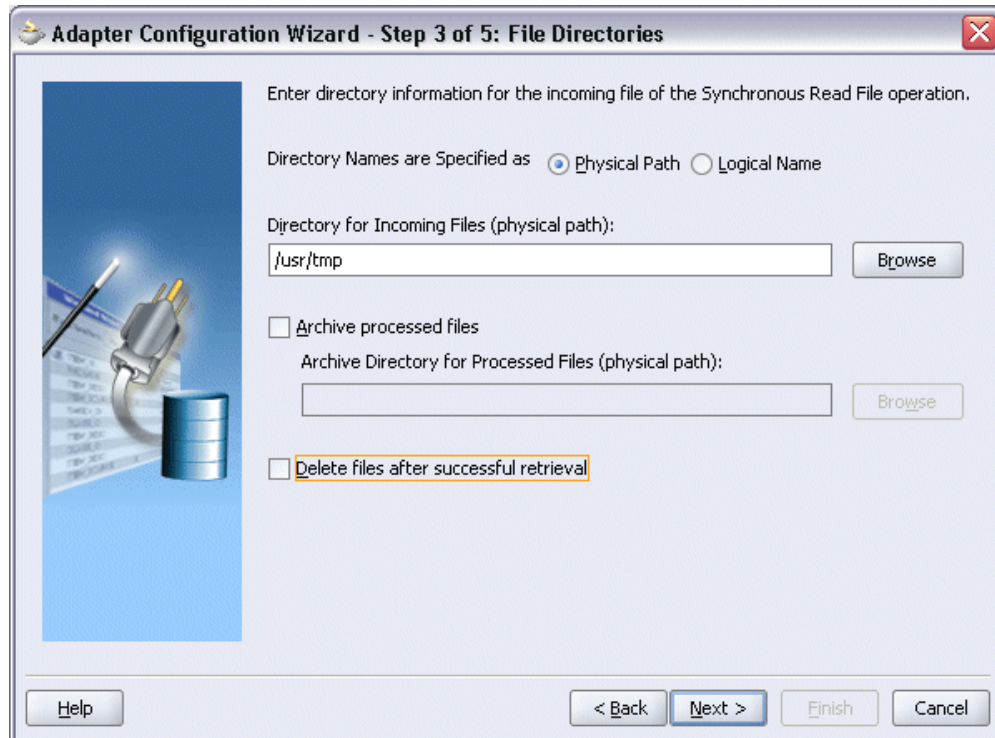
Operation Name:

Help < Back Next > Finish Cancel

5. Specify the operation type, for example **Synchronous Read File**. This automatically populates the **Operation Name** field.

Click **Next** to access the File Directories dialog box.

### Specifying the Input File Directory



Adapter Configuration Wizard - Step 3 of 5: File Directories

Enter directory information for the incoming file of the Synchronous Read File operation.

Directory Names are Specified as ☒ Physical Path ☐ Logical Name

Directory for Incoming Files (physical path):

☐ Archive processed files  
Archive Directory for Processed Files (physical path):

☐ Delete files after successful retrieval

Help < Back Next > Finish Cancel

6. Select **Physical Path** radio button and enter the input payload file directory information. For example, enter `/usr/tmp/` as the directory name.

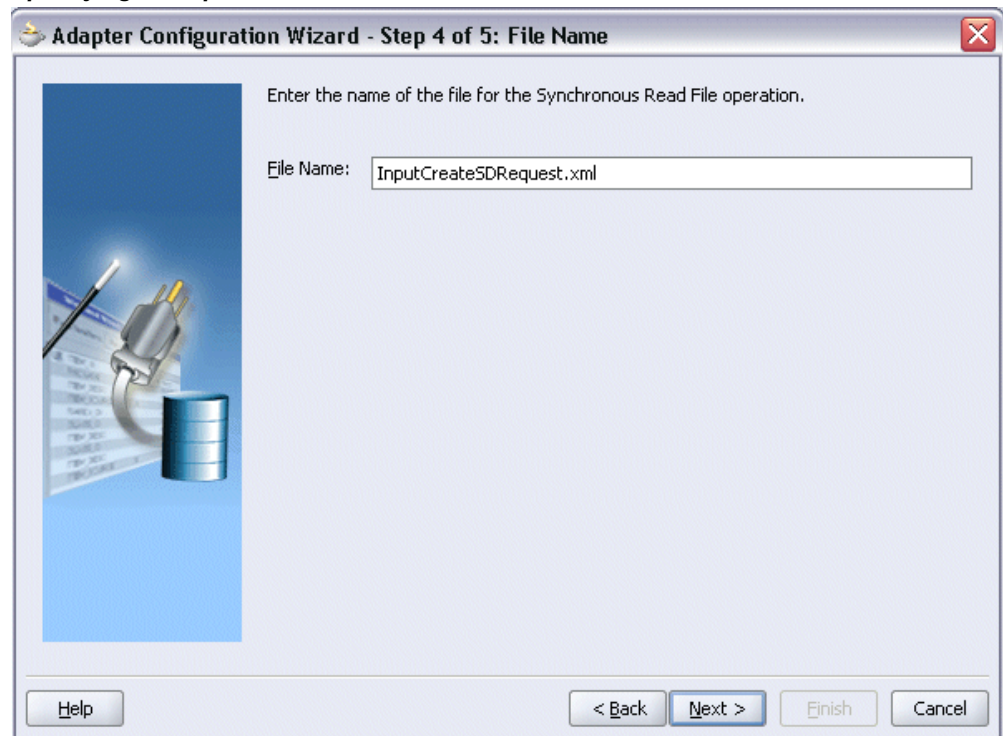
**Note:** You must ensure the input payload file `InputCreateSDRequest.xml` is available in the directory `'/usr/tmp/'` folder of SOA Suite server (or `D:\HOL` in case of SOA Server in Windows machine).

Uncheck the **Delete Files after successful retrieval** check box. Click **Next** to open the File Name dialog box.

7. Enter the name of the file for the synchronous read file operation. For example, enter `InputCreateSDRequest.xml`.



### Specifying the Input File Name



Click **Next**. The Messages dialog box appears.

8. Select **Browse** for schema file in Schema Location.

The Type Chooser window is displayed.

### Specifying Message Schema

Adapter Configuration Wizard - Step 5 of 5: Messages

Define the message for the Synchronous Read File operation. Specify the Schema File Location and select the Schema Element that defines the messages in the incoming files. Use the Browse button to find an existing schema definition. If you check 'Schema is Opaque', then you do not need to specify a Schema.

**Message Schema**

☐ Native format translation is not required (Schema is Opaque)

Define Schema for Native Format

Schema Location:  Browse

Schema Element:

Help < Back Next > Finish Cancel

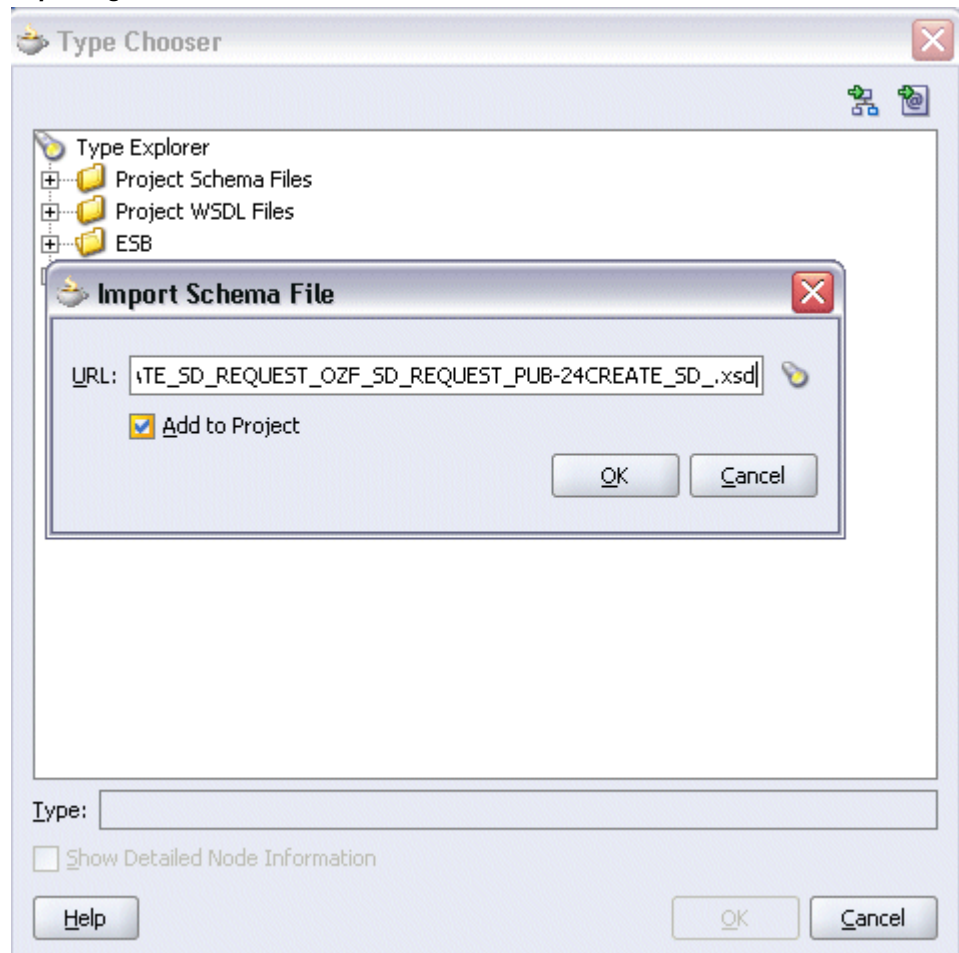
Click **Import Schema Files** button on the top right corner of the Type Chooser window.

Enter the schema location for the service. Such as  
`http://<myhost>:<port>/webservices/SOAPProvider/plsql/ozf_sd_request_pub/APPS_ISG_CREATE_SD_REQUEST_OZF_SD_REQUEST_PUB-24CR  
EATE_SD_.xsd.`

Schema location for your service can be found from the service WSDL URL (for example,  
`http://<myhost>:<port>/webservices/SOAPProvider/plsql/ozf_sd_request_pub/?wsdl).`

Select the **Add to Project** check box and click **OK**.

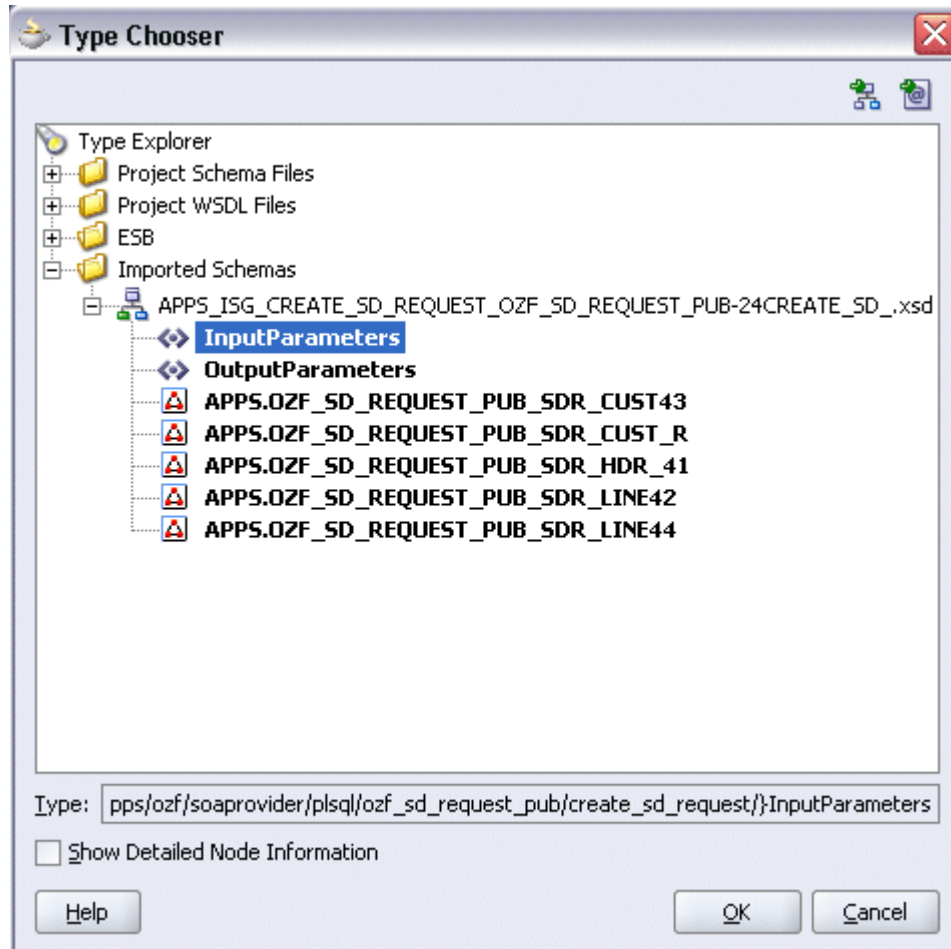
### Importing Schema File



Click **OK** for Import schema prompt.

The Imported Schemas folder is automatically added to the Type Chooser window.

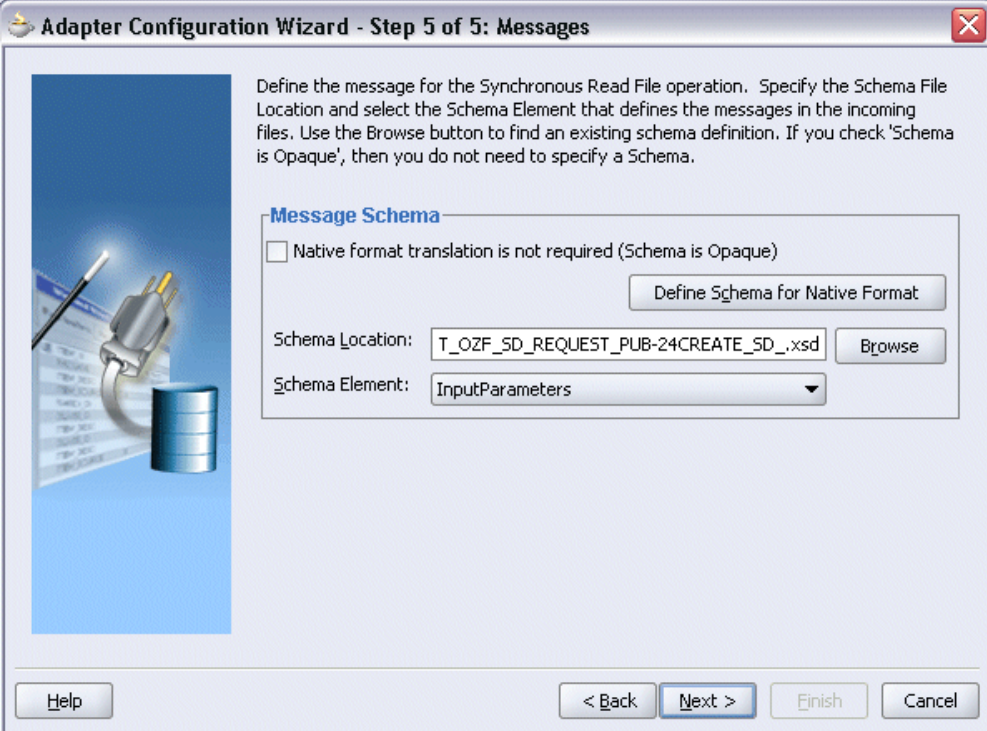
### Identifying Message Schema



Expand the Imported Schemas folder and select InputParameters Message in the APPS\_ISG\_CREATE\_SD\_REQUEST\_OZF\_SD\_REQUEST\_PUB-24CREATE\_SD\_.xsd. Click **OK**.

The selected xsd is displayed as Schema Location, and the InputParameters is selected as Schema Element.

### Viewing Selected Message Schema and Element



**Adapter Configuration Wizard - Step 5 of 5: Messages**

Define the message for the Synchronous Read File operation. Specify the Schema File Location and select the Schema Element that defines the messages in the incoming files. Use the Browse button to find an existing schema definition. If you check 'Schema is Opaque', then you do not need to specify a Schema.

**Message Schema**

☐ Native format translation is not required (Schema is Opaque)

Define Schema for Native Format

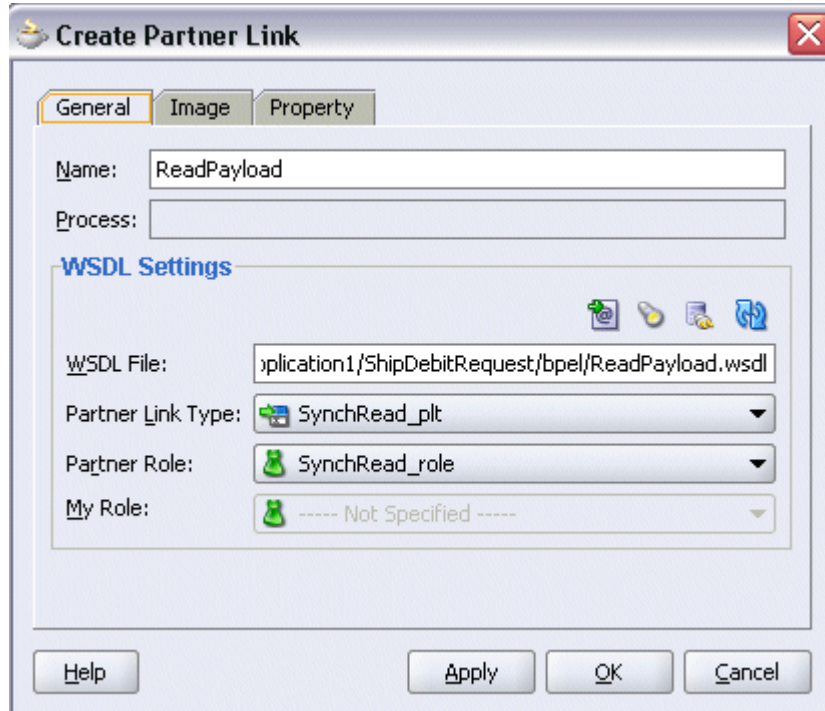
Schema Location: T\_OZF\_SD\_REQUEST\_PUB-24CREATE\_SD\_.xsd Browse

Schema Element: InputParameters

Help < Back Next > Finish Cancel

9. Click **Next** and 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 `ReadPayload.wsdl`.

### Completing the Partner Link Configuration



The image shows a 'Create Partner Link' dialog box with three tabs: 'General', 'Image', and 'Property'. The 'General' tab is active. It contains the following fields and controls:

- Name:** ReadPayload
- Process:** (empty text box)
- WSDL Settings:**
  - WSDL File:** Application1/ShipDebitRequest/bpel/ReadPayload.wsdl
  - Partner Link Type:** SynchRead\_plt
  - Partner Role:** SynchRead\_role
  - My Role:** ----- Not Specified -----

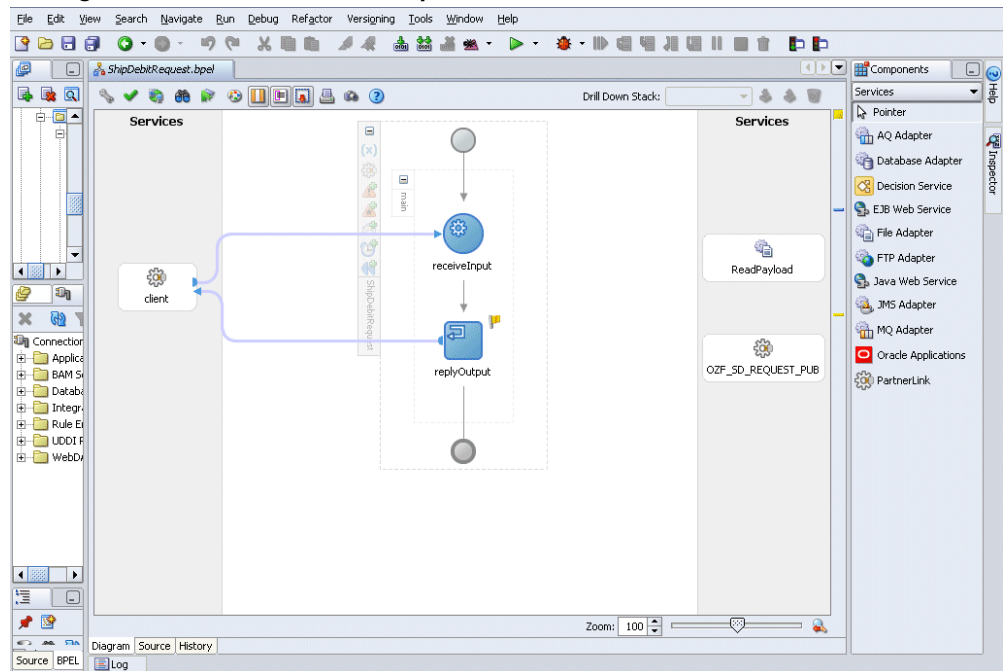
At the bottom of the dialog are four buttons: 'Help', 'Apply', 'OK', and 'Cancel'.

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

The ReadPayload Partner Link appears in the BPEL process diagram.



## Adding the Partner Link for File Adapter



## Adding Invoke Activities

This step is to configure two Invoke activities:

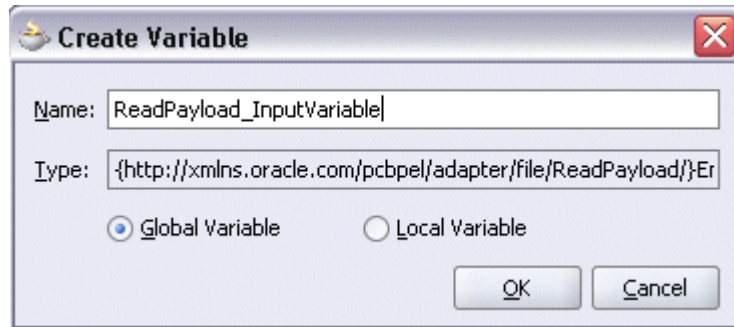
- Read request creation details that is passed from the first Assign activity using ReadPayload partner link for File Adapter.
- Send the payload and request details received from the Assign activities to create a ship and debit request by using the OZF\_SD\_REQUEST\_PUB partner link.

### To add an Invoke activity for ReadPayload Partner Link:

1. In JDeveloper BPEL Designer, drag and drop the **Invoke** activity from the Component Palette into the Activity box of the process diagram, between the **Receive** and **Reply** activities.
2. Link the Invoke activity to the ReadPayload service. The Edit Invoke dialog box appears.
3. Enter a name for the Invoke activity, and then click the **Create** icon next to the **Input Variable** field to create a new variable. The Create Variable dialog box appears.

Enter 'ReadPayload\_InputVariable' as the input variable name. You can also accept the default name.

### Creating a Variable



The 'Create Variable' dialog box is shown. It has a title bar with a close button. The 'Name' field contains 'ReadPayload\_InputVariable'. The 'Type' field contains '{http://xmlns.oracle.com/pcbpel/adapter/file/ReadPayload/}Er'. There are two radio buttons: 'Global Variable' (selected) and 'Local Variable'. At the bottom are 'OK' and 'Cancel' buttons.

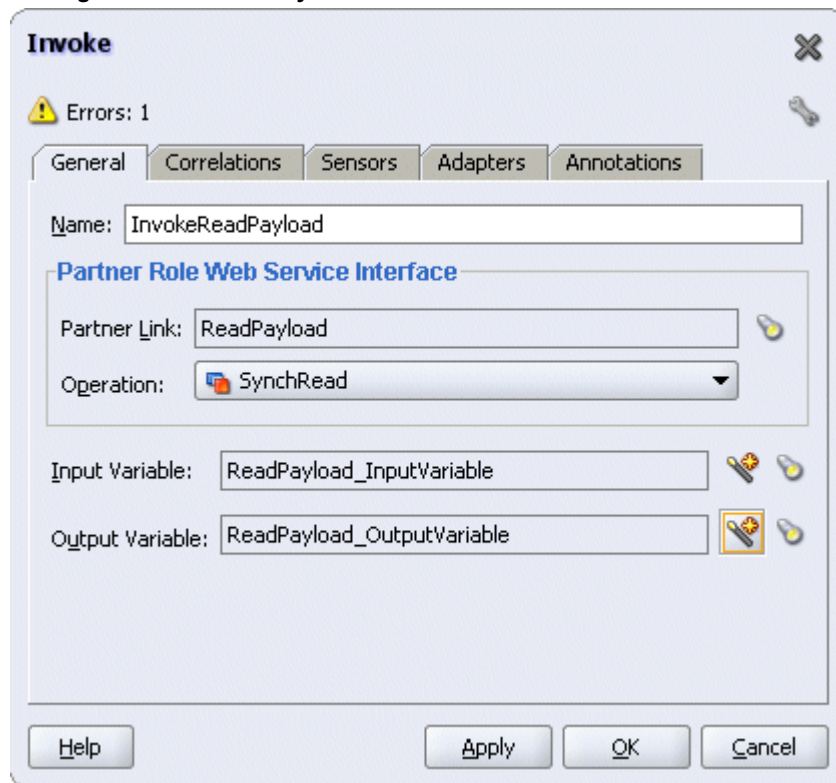
Select **Global Variable**, and then enter a name for the variable. Click **OK**.

4. Click the **Create** icon next to the **Output Variable** field to create a new variable. The Create Variable dialog box appears.

Enter 'ReadPayload\_OutputVariable' as the output variable name. You can also accept the default name.

Select **Global Variable**, and then enter a name for the variable. Click **OK**.

### Editing the Invoke Activity



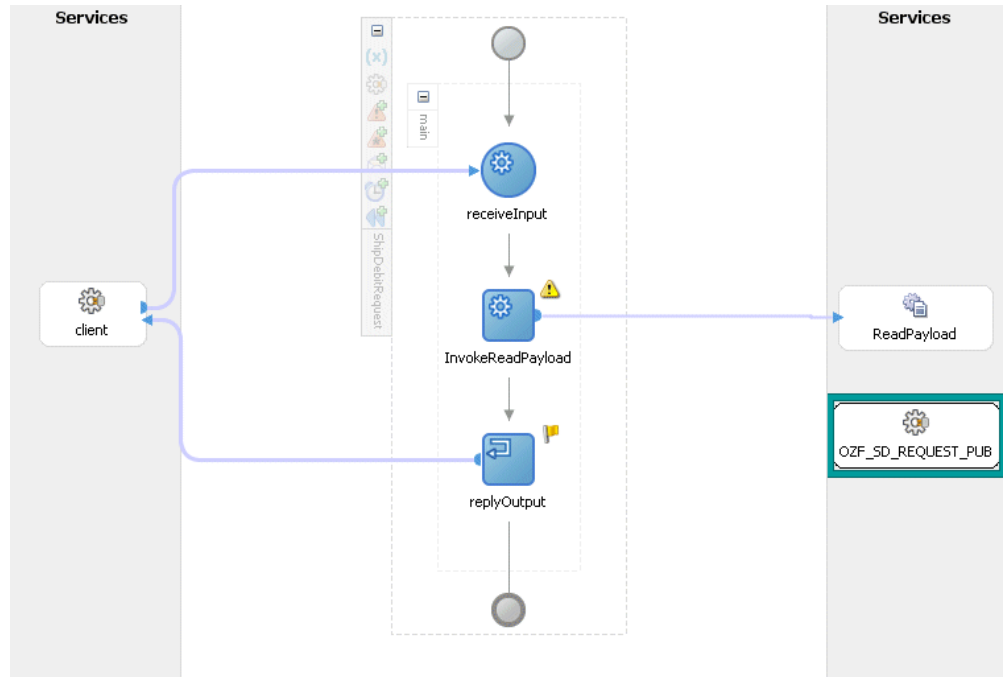
The 'Invoke' dialog box is shown. It has a title bar with a close button. Below the title bar is a warning icon and the text 'Errors: 1'. There are five tabs: 'General' (selected), 'Correlations', 'Sensors', 'Adapters', and 'Annotations'. The 'Name' field contains 'InvokeReadPayload'. Below this is a section titled 'Partner Role Web Service Interface'. It contains two fields: 'Partner Link' with the value 'ReadPayload' and a lightbulb icon, and 'Operation' with a dropdown menu showing 'SynchRead'. Below these are two more fields: 'Input Variable' with the value 'ReadPayload\_InputVariable' and a key icon, and 'Output Variable' with the value 'ReadPayload\_OutputVariable' and a key icon. At the bottom are 'Help', 'Apply', 'OK', and 'Cancel' buttons.



5. Click **Apply** and then **OK** in the Edit Invoke dialog box to finish configuring the Invoke activity.

The Invoke activity appears in the process diagram.

#### **Adding an Invoke Activity**



#### **To add an Invoke activity for OZF\_SD\_REQUEST\_PUB Partner Link:**

1. In JDeveloper BPEL Designer, drag and drop the **Invoke** activity from the Component Palette into the Activity box of the process diagram, after the first **Invoke** activity and the **Reply** activity.
2. Link the Invoke activity to the OZF\_SD\_REQUEST\_PUB service. The Edit Invoke dialog box appears.
3. Enter a name for the Invoke activity such as 'Invoke\_EBS\_SDR\_Service'.  
In the Operation field, select CREATE\_SD\_REQUEST from the drop-down list.
4. Create global Input and Output variables as CREATE\_SD\_REQUEST\_InputVariable and CREATE\_SD\_REQUEST\_OutputVariable.  
Click **OK** in Edit Invoke.

### Editing the Invoke Activity

**Invoke**

Errors: 1

General Correlations Sensors Adapters Annotations

Name: Invoke\_EBS\_SDR\_Service

**Partner Role Web Service Interface**

Partner Link: OZF\_SD\_REQUEST\_PUB

Operation: CREATE\_SD\_REQUEST

Input Variable: CREATE\_SD\_REQUEST\_InputVariable

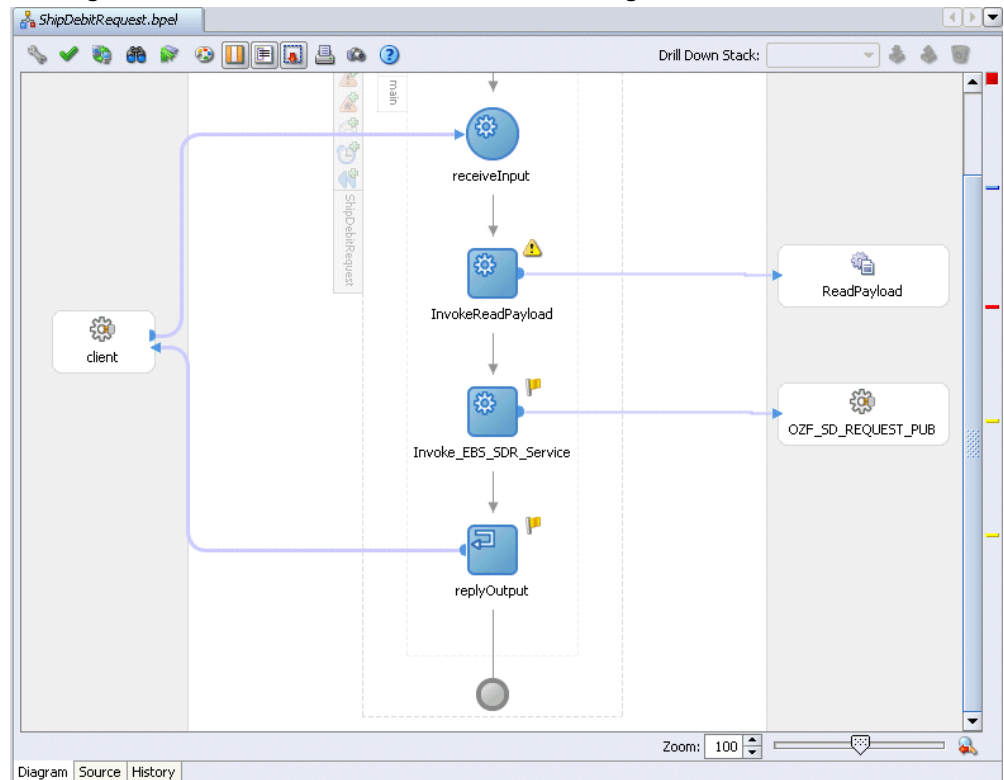
Output Variable: CREATE\_SD\_REQUEST\_OutputVariable

Help Apply OK Cancel

Click **Apply** and then **OK** in the Edit Invoke dialog box to finish configuring the Invoke activity.

The Invoke activity appears in the process diagram.

### Viewing the Invoke Activities in the BPEL Process Diagram



## Adding Assign Activities

This step is to configure four Assign activities:

1. To set the SOAHeader details for ship and debit SOAP request.

**Note:** You also need to populate certain variables in the BPEL process for SOAHeader elements to pass values that may be required to set application context during service execution. These SOAHeader elements are *Responsibility*, *RespApplication*, *SecurityGroup*, *NLSLanguage*, and *Org\_Id*.

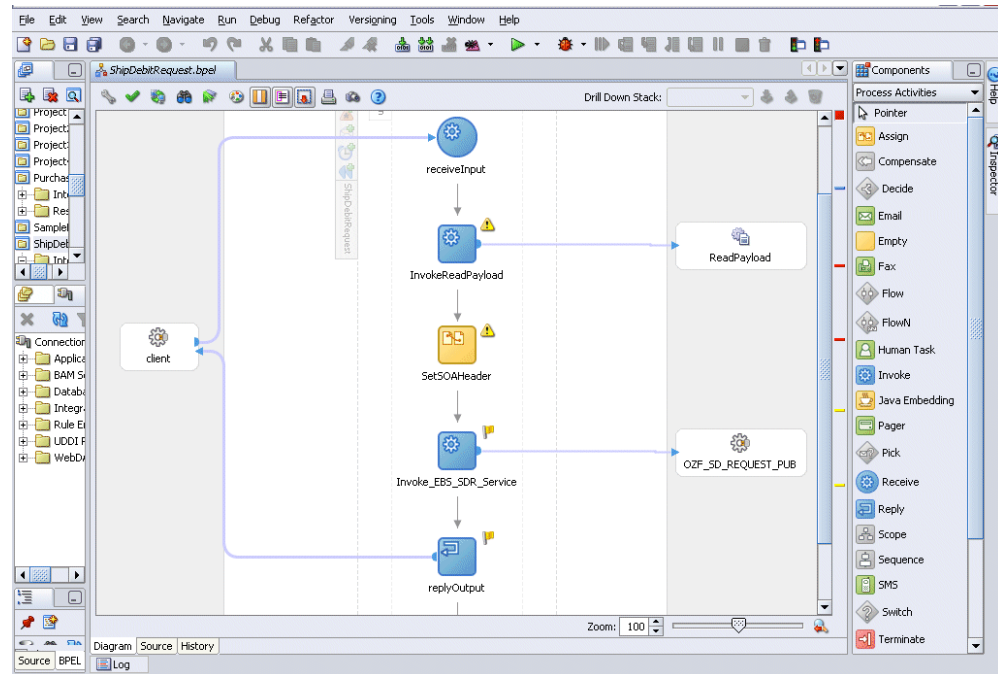
2. To set input payload for SOAP request.
3. To set input for SOAP request.
4. To set the SOAP response to output.

**To add the first Assign activity to set SOAHeader details:**

### Assigning SOAHeader Parameters:

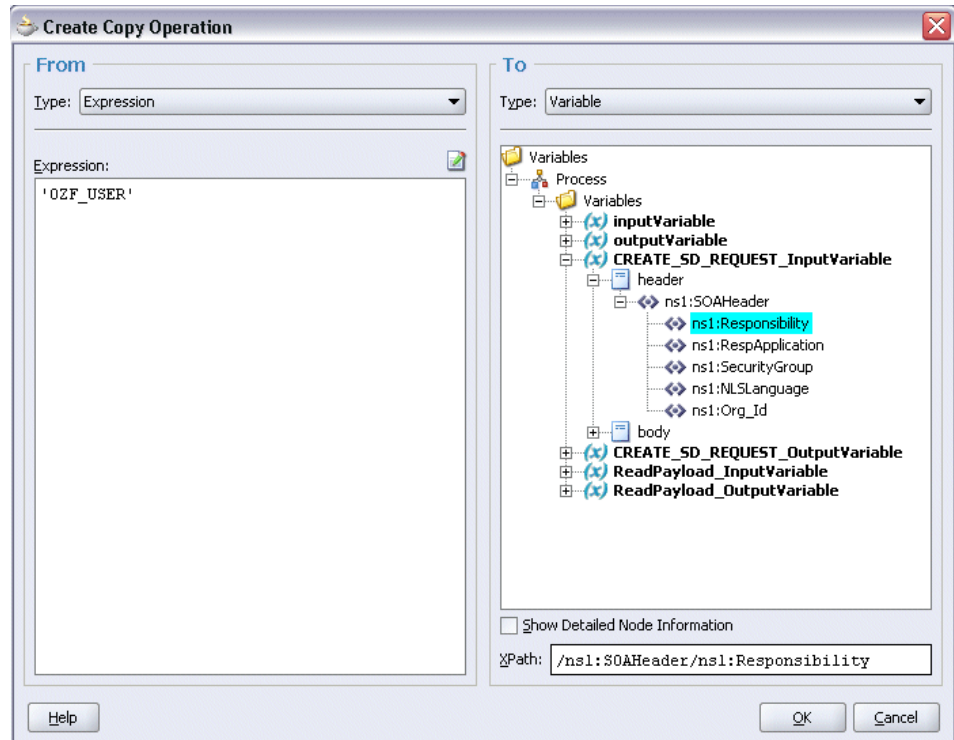
1. In JDeveloper BPEL Designer, drag and drop the **Assign** activity from the Component Palette into the Activity box of the process diagram, between two **Invoke** activities.

#### Adding an Assign Activity



2. Double-click the **Assign** activity to access the Edit Assign dialog box.
3. Click the General tab to enter the name for the Assign activity, such as 'SetSOAHeader'.
4. On the Copy Operation tab, click **Create** and then select **Copy Operation** from the menu. The Create Copy Operation window appears.
5. Enter the first pair of parameters:
  - In the From navigation tree, select type Expression and then enter 'OZF\_USER' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variables > Process > Variables > CREATE\_SD\_REQUEST\_InputVariable > header > ns1:SOAHeader** and select **ns1:Responsibility**. The XPath field should contain your selected entry.

### Assign Responsibility Parameter

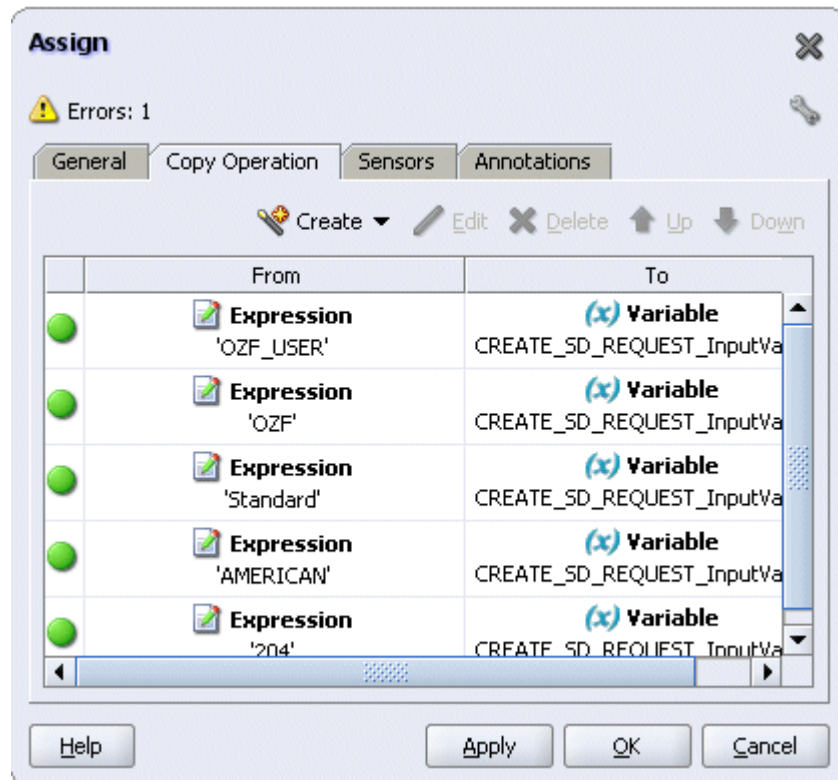


- Click **OK**.
6. Enter the second pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
- In the From navigation tree, select type Expression and then enter 'OZF' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variables > Process > Variables > CREATE\_SD\_REQUEST\_InputVariable > header > ns1:SOAHeader** and select **ns1:RespApplication**. The XPath field should contain your selected entry.
  - Click **OK**.
7. Enter the third pair of parameters:
- In the From navigation tree, select type Expression and then enter 'STANDARD' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variables > Process > Variables > CREATE\_SD\_REQUEST\_InputVariable > header >**

**ns1:SOAHeader** and select **ns5:SecurityGroup**. The XPath field should contain your selected entry.

- Click **OK**.
8. Enter the fourth pair of parameters:
- In the From navigation tree, select type Expression and then enter 'AMERICAN' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variables > Process > Variables > CREATE\_SD\_REQUEST\_InputVariable > header > ns1:SOAHeader** and select **ns1:NLSLanguage**. The XPath field should contain your selected entry.
  - Click **OK**.
9. Enter the fifth pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
- In the From navigation tree, select type Expression and then enter '204' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variables > Process > Variables > CREATE\_SD\_REQUEST\_InputVariable > header > ns1:SOAHeader** and select **ns1:Org\_Id**. The XPath field should contain your selected entry.
  - Click **OK**.
10. The Edit Assign dialog box appears.

### Assign Parameters



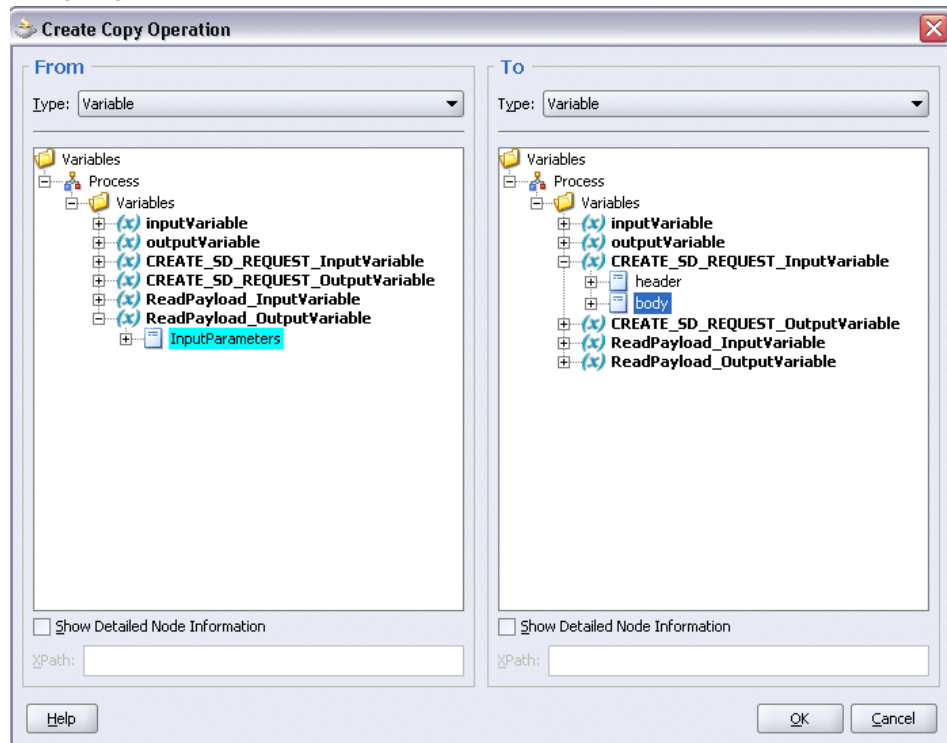
11. Click **Apply** and then **OK** to complete the configuration of the Assign activity.

**To enter the second Assign activity to pass payload information to the Invoke\_EBS\_SDR\_Service Invoke activity:**

1. Add the second Assign activity by dragging and dropping the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the 'SetSOAHeader' **Assign** activity and the 'Invoke\_EBS\_SDR\_Service' **Invoke** activity.
2. Repeat Step 2 to Step 4 described in creating the first Assign activity to add the second Assign activity called 'SetPayload'.
3. Enter the following information:
  - In the From navigation tree, navigate to **Variable > Process > Variables > ReadPayload\_OutVariable** and select **InputParameters**.
  - In the To navigation tree, select type Variable and then navigate to **Variable > Process > Variables > CREATE\_SD\_REQUEST\_InputVariable** and select **Body**.

- Click **OK**.

### Assigning Parameters



4. The Edit Assign dialog box appears.
5. Click **Apply** and then **OK** to complete the configuration of the Assign activity.

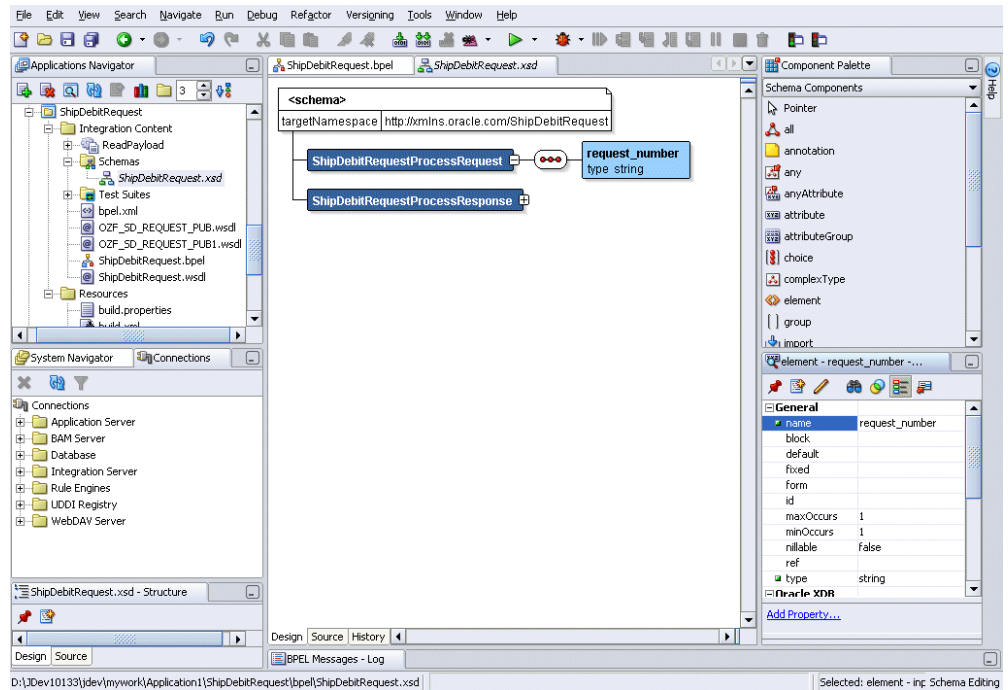
### Defining Schema for BPEL Process Input Request

Before setting the input request for the SOAP request, you need to define necessary schema for BPEL process request.

1. From the Applications Navigator window, expand the **ShipDebitRequest > Integration Content > Schemas** folder to open the `ShipDebitRequest.xsd` file.
2. In the Design mode, expand 'ShipDebitRequestProcessRequest' to view elements within process request.



## Defining Schema for BPEL Process Request

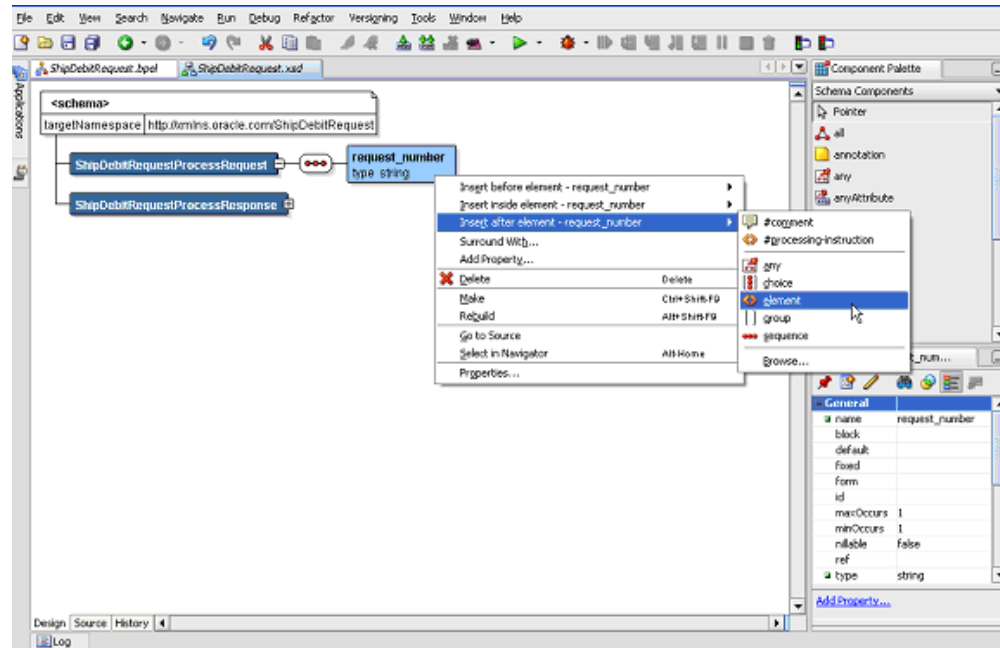


3. From element properties, change the name from 'input' to 'request\_number'.
4. Select and right-click on the 'request\_number' element to open the pop-up menu.

Select **Insert after element – request\_number > element** option. New element 'element1' is displayed in the schema design window underneath the 'request\_number' element.

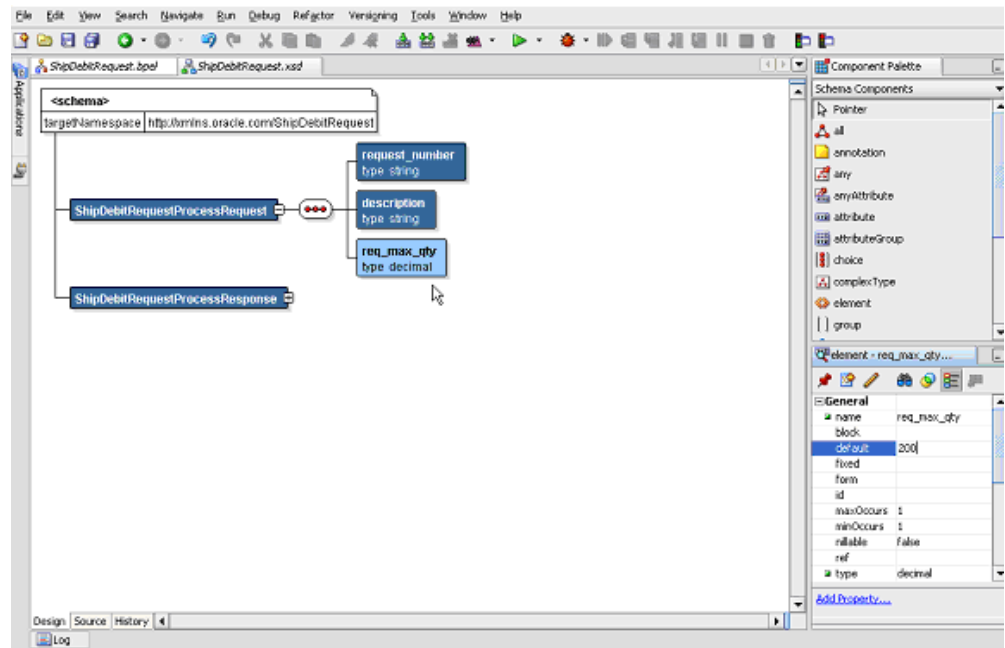
From element properties section, change the name from 'element1' to 'description' and enter type as 'string'.

## Adding Schema Elements



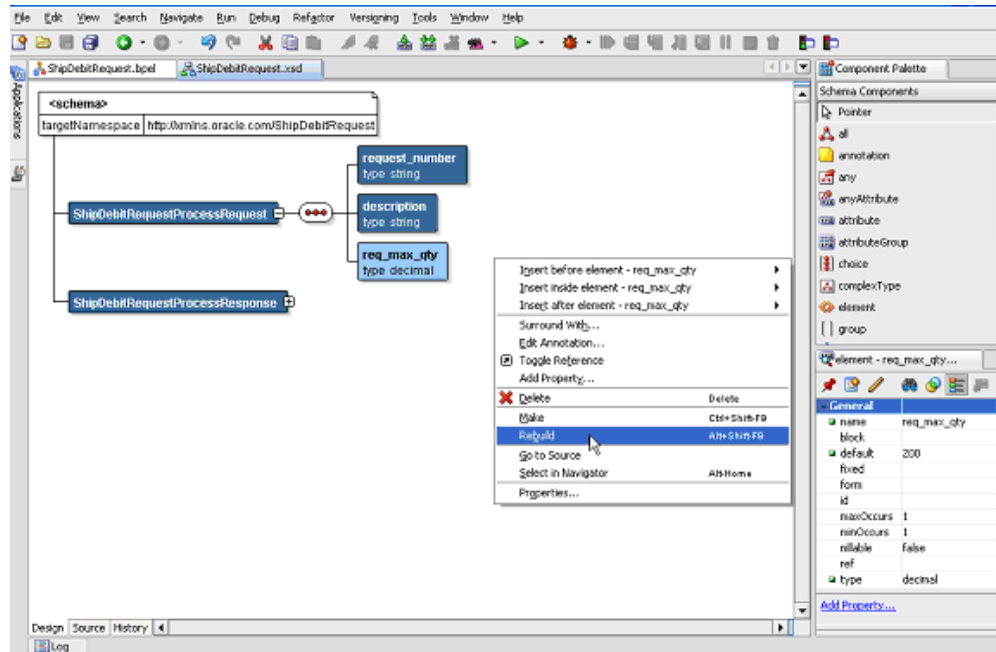
5. Similarly insert another element called 'req\_max\_qty' after element 'description'. Enter default value as '200' and type as 'decimal'.

## Defining Schema Elements



Right-click on mouse and select **Rebuild** option.

## Rebuilding Schema Elements

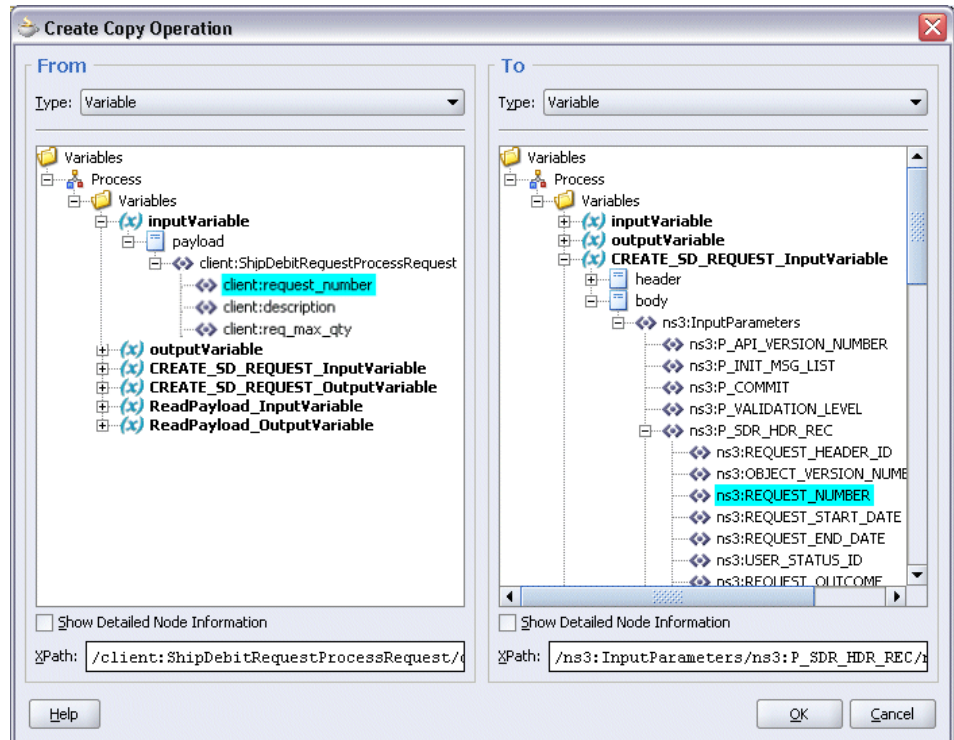


Look for compilation messages in Log to ensure the successful compilation.

To set the third Assign activity to pass the input request to the Invoke\_EBS\_SDR\_Service Invoke activity:

1. Add the third Assign activity by dragging and dropping the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the second **Assign** activity 'SetPayload' and the Invoke\_EBS\_SDR\_Service **Invoke** activity.
2. Repeat Step 2 to Step 4 described in creating the first Assign activity to add the third Assign activity called 'SetInput'.
3. Enter the following information:
  - In the From navigation tree, navigate to **Variable > Process > Variables > inputVariable > Payload > client:ShipDebitRequestProcessRequest** and select **client:request\_number**. The XPath field should contain your selected entry.
  - In the To navigation tree, select type Variable and then navigate to **Variable > Process > Variables > Create\_SD\_REQUEST\_InputVariable > Body > ns3:InputParameters > ns3:P\_SDR\_HDR\_REC** and select **ns3:REQUEST\_NUMBER**. The XPath field should contain your selected entry.

## Assigning Parameters



- Click **OK**.
4. Enter the second pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
    - In the From navigation tree, navigate to **Variable > Process > Variables > inputVariable > Payload > client:ShipDebitRequestProcessRequest** and select **client:description**. The XPath field should contain your selected entry.
    - In the To navigation tree, select type Variable and then navigate to **Variable > Process > Variables > Create\_SD\_REQUEST\_InputVariable > Body > ns3:InputParameters > ns3:P\_SDR\_HDR\_REC** and select **ns3:REQUEST\_DESCRIPTION**. The XPath field should contain your selected entry.
    - Click **OK**.
  5. Enter the third pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
    - In the From navigation tree, navigate to **Variable > Process > Variables >**

**inputVariable > Payload > client:ShipDebitRequestProcessRequest** and select **client:req\_max\_qty**. The XPath field should contain your selected entry.

- In the To navigation tree, select type Variable and then navigate to **Variable > Process > Variables > Create\_SD\_REQUEST\_InputVariable > Body > ns3:InputParameters > ns3:P\_SDR\_LINES\_TBL > ns3:P\_SDR\_LINES\_TBL\_ITEM** and select **ns3:MAX\_QTY**. The XPath field should contain your selected entry.
- Click **OK**.

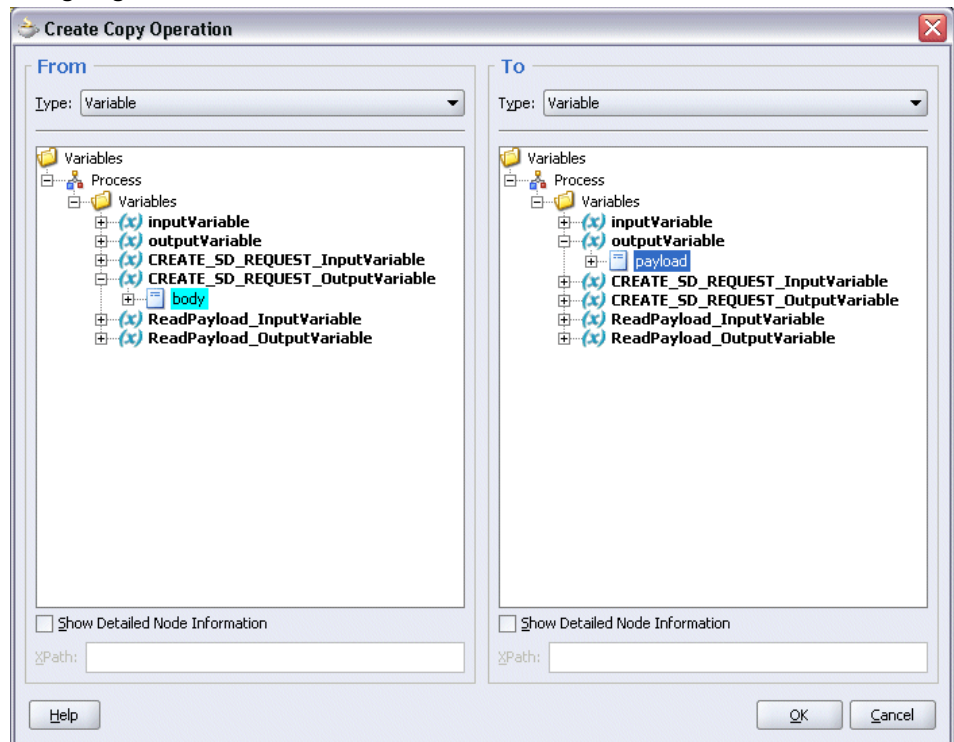
6. The Edit Assign dialog box appears.

Click **Apply** and then **OK** to complete the configuration of the Assign activity.

**To add the fourth Assign activity to set SOAP response to output:**

1. Add the third Assign activity by dragging and dropping the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the **Invoke\_EBS\_SDR\_Service Invoke** and the **Reply** activities.
2. Repeat Step 2 to Step 4 described in creating the first Assign activity to add the fourth Assign activity called 'SetResponse'.
3. Enter the following information:
  - In the From navigation tree, select type Variable. Navigate to **Variable > Process > Variables > CREATE\_SD\_REQUEST\_OutputVariable** and select **body**.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > outputVariable** and select **payload**.

### Assigning Parameters



- Click **OK**.
4. The Edit Assign dialog box appears.  
Click **Apply** and then **OK** to complete the configuration of the Assign activity.

## Deploying and Testing the BPEL Process

After creating a BPEL process using the WSDL URL generated from a PL/SQL interface definition, you can deploy it to a BPEL server if needed. To ensure that this process is modified or orchestrated appropriately, you can also manually test the BPEL process by initiating the business process contained in the BPEL process to test the interface integration.

### Prerequisites

Before deploying the BPEL process using Oracle JDeveloper, you must ensure that you have established the connectivity between the design-time environment and the run-time servers including the application server and the integration server.

How to configure the necessary server connection, see *Configuring Server Connection*, page B-1.

The payload information for the creation of supplier ship and debit request, see Sample Payload for Creating Supplier Ship and Debit Request, page C-1.

To validate your BPEL process, perform the following run-time tasks:

1. Deploy the BPEL process, page 3-38

Once you deploy the process to a BPEL server, it becomes available so that you can run the process manually to test it for validation.

2. Test the BPEL process, page 3-39

After deploying a BPEL process, you can manage the process from the BPEL console to manually initiate the business process and test the interface integration contained in your BPEL process.

## Deploying the BPEL Process

You must deploy the BPEL process (`ShipDebitRequest.bpel`) that you created earlier before you can run it.

**To deploy the BPEL process:**

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

2. Right-click the project and click **Make** action from the menu.

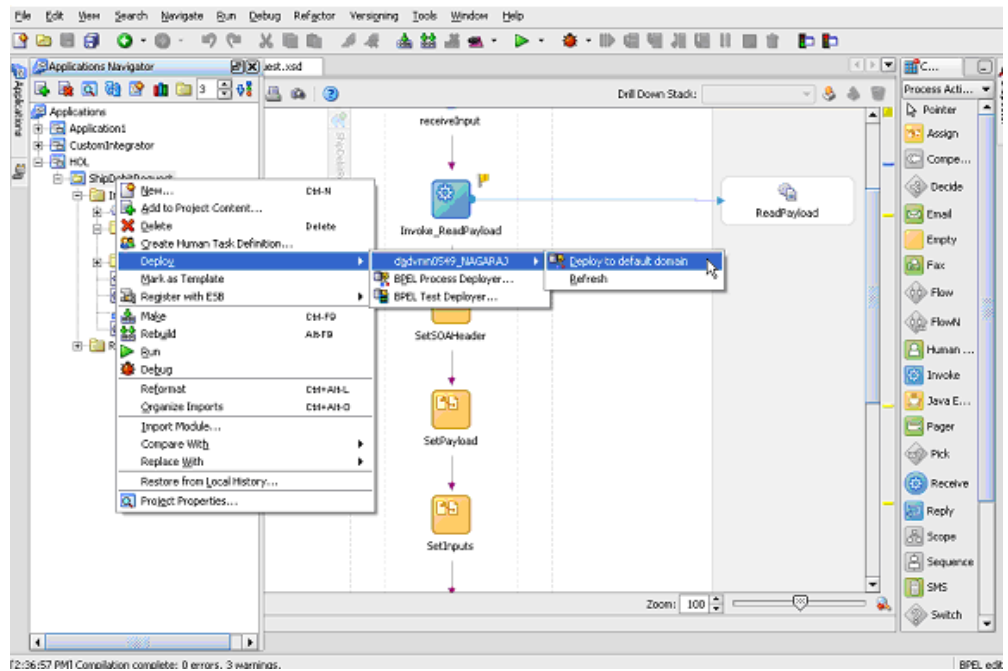
Look for any compilation error messages in Message Log.

Right-click the project and select **Deploy > Integration Server Connection name > Deploy to Default Domain** action from the menu.

For example, you can select **Deploy > BPEServerConn > Deploy to Default Domain** to deploy the process if you have the BPEL Process Manager setup appropriately.



## Deploying the BPEL Process



Look for 'Build successful' message in Apache Ant – Log to ensure that the BPEL project is compiled and successfully deployed.

## Testing the BPEL Process

To validate whether the BPEL process that you created works or not, you need to manually initiate the process after it has been successfully deployed to the BPEL server. Therefore, the validation starts with the BPEL console to ensure that you can find the deployed BPEL process listed in the console. Then, you can log on to Oracle E-Business Suite to manually initiate the purchase order approval and acknowledgement processes and to confirm that the relevant event is raised and the updated purchased order details is also written in the XML file.

### To test the BPEL process:

1. Log into Oracle Application Server 10g BPEL Console (<http://<soaSuiteServerHostName>:<port>/BPELConsole>). The BPEL Console login page appears.
2. Enter the username and password and click **Login**.
3. In the BPEL Console, confirm that ShipDebitRequest has been deployed.

## Deployed BPEL Processes

ORACLE® Enterprise Manager 10g  
BPEL Control

Logout | Support

Dashboard Processes Instances Activities Configuration Administration

Deployed BPEL Processes		In-Flight BPEL Process Instances		
Name	Instance	BPEL Process	Last Modified ↑	
CreateSingleInvoice_BPEL				
Create_Invoice				
GEHCTest_MLR9				
GetOrder ( v. 1.0 )				
GetOrder ( v. 2.0 )				
GetOrder ( v. 3.0 )				
GetOrder ( v. 3.1 )				
GetOrder ( v. 3.2 ) *				
ISGTest_BPEL				
ShipDebitRequest ( v. 1.0 )				
ShipDebitRequest ( v. 2.0 )				
ShipDebitRequest ( v. 2.1 )				
ShipDebitRequest ( v. 3.0 )				
ShipDebitRequest ( v. 3.0 ) *				
TaskActionHandler				
TaskManager				

Recently Completed BPEL Process Instances ([More...](#))

✓ 560067 : Instance #560067 of ShipDebitRequest	ShipDebitRequest (v. 3.0)	9/18/09 9:54:09 AM
✓ 560066 : Instance #560066 of ShipDebitRequest	ShipDebitRequest (v. 3.0)	9/18/09 9:52:57 AM
560063 : Instance #560063 of GetOrder	GetOrder (v. 3.2)	9/17/09 7:07:38 AM
560062 : Instance #560062 of GetOrder	GetOrder (v. 3.2)	9/17/09 6:50:30 AM
560061 : Instance #560061 of GetOrder	GetOrder (v. 3.1)	9/17/09 6:33:37 AM

Deploy New Process

4. Click the ShipDebitRequest link to open the Initiate tab

5. In the payload region, enter the following fields:

- request\_number: Enter an unique number in this field, such as BPEL-1.

**Note:** The Request Number entered here should be unique each time that you initiate. The Supplier Ship and Debit Request Number should be unique across users in Supplier Ship and Debit of Oracle Trade Management.

- description: Enter appropriate description information.
- req\_max\_qty: Enter 100 as the value.

## Specifying Input Payload Information

The screenshot shows the Oracle Enterprise Manager 10g BPEL Control interface. The top navigation bar includes 'Dashboard', 'Processes', 'Instances', 'Activities', 'Configuration', and 'Administration'. The 'Processes' tab is active, showing details for the 'ShipDebitRequest' BPEL process, version 5.0, with a lifecycle of 'Active'. Below this, there are links for 'Open Instances' and 'Closed Instances'. The 'Initiate' sub-tab is selected, displaying a form to create a new test instance. The form includes a dropdown for 'Operation' set to 'process', and radio buttons for 'HTML Form' (selected) and 'XML Source'. There are checkboxes for 'WS-Security' and 'WS-Addressing', each with an 'Include In Header' option. A 'payload' section contains three input fields: 'request\_number' with value 'BPEL-1' (xsd:string), 'description' with value 'Ship debit request fr' (xsd:string), and 'req\_max\_qty' with value '100' (xsd:decimal). Below the form, there are checkboxes for 'Save Test' and 'Perform stress test', and a 'Post XML Message' button. A note states: 'Note: XML source view contents will not be reflected in the HTML form view'.

ORACLE® Enterprise Manager 10g  
BPEL Control

Logout | Support

Dashboard Processes Instances Activities Configuration Administration

BPEL Process: ShipDebitRequest Version: 5.0 Lifecycle: Active  
Statistics: [Open Instances](#) | [Closed Instances](#)

Manage **Initiate** Descriptor WSDL Sensors Source Analytics Validate XML Test Reports

Testing this BPEL Process Through SOAP | Through Java Delivers API

**Initiating a test instance**  
To create a new 'test' instance of this BPEL Process, fill this form and click on the 'Post XML Message' button.

Operation: process HTML Form XML Source

☒ WS-Security ☐ Include In Header

☒ WS-Addressing ☐ Include In Header

☒ payload

request\_number: BPEL-1 xsd:string  
description: Ship debit request fr xsd:string  
req\_max\_qty: 100 xsd:decimal

Note: XML source view contents will not be reflected in the HTML form view

☐ Save Test  
☐ Perform stress test

Post XML Message

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

## 6. Verifying SOAP Response in BPEL Console

## Verifying SOAP Response in BPEL Console

The screenshot shows the Oracle Enterprise Manager 10g BPEL Console interface. The top navigation bar includes 'Dashboard', 'Processes', 'Instances', 'Activities', 'Configuration', and 'Administration'. The 'Processes' tab is active, showing the 'ShipDebitRequest' process with version '5.0' and lifecycle 'Active'. Below this, there are links for 'Open Instances' and 'Closed Instances'. The 'Test' tab is selected, displaying the 'Test Instance Initiated' status. The main content area shows the SOAP response for the test request, which was processed synchronously. The response is an XML document with the following structure:

```
<ShipDebitRequestProcessResponse xmlns="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.oracle.com/ShipDebitRequest" >
  <X_RETURN_STATUS xmlns="http://xmlns.oracle.com/apps/ozf/soapprovider/plsql/ozf_sd_request_pub/create_sd_request/"
  >S</X_RETURN_STATUS>
  <X_MSG_COUNT xmlns="http://xmlns.oracle.com/apps/ozf/soapprovider/plsql/ozf_sd_request_pub/create_sd_request/"
  >23</X_MSG_COUNT>
  <X_REQUEST_HEADER_ID
  xmlns="http://xmlns.oracle.com/apps/ozf/soapprovider/plsql/ozf_sd_request_pub/create_sd_request/"
  >182</X_REQUEST_HEADER_ID>
</ShipDebitRequestProcessResponse>
```

Below the response, there are three links for more information: 'Visual Flow', 'Audit Instance', and 'Debug Instance'. Each link is accompanied by a small icon representing its respective function.

You can view the SOAP response displayed synchronously in BPEL Console. Look for 'S' in X\_RETURN\_STATUS (for success). If 'E' is displayed in X\_RETURN\_STATUS, then it means error has occurred while processing the service. Look for detailed exception message in SOA Monitor.

## 7. Verifying Created Supplier Ship and Debit Request in Oracle Trade Management

Log on to Oracle E-Business Suite with the Oracle Trade Management User responsibility. Select the **Supplier Ship and Debit** link from the navigation menu to open the Ship and Debit Overview window.

## Verifying in Oracle Trade Management

ORACLE Trade Management

Home Trade Management Direct-ADS-Disco Profile

Customer Product Quota Budget Trade Planning Claim Indirect Sales Management Administration Calendar Supplier Ship and Debit

Requests Batch Creation Batch Summary

Quick Find Offer Go

Logged In As TRADEMG

Ship and Debit Overview

Views

View My Ship and Debit Requests Go Personalize

Advanced Search

Create Ship and Debit Request

Request Number	Accrual Type	Supplier	Supplier Location	Authorization Number	Status
<a href="#">BPEL-1</a>	Supplier	Abbott Laboratories, Inc.	CORP HQ		Assigned
<a href="#">SD163</a>	Supplier	Advanced Network Devices	SANTA CLARA-ERS		Pending Supplier Approval
<a href="#">SD-PERFORMANCE-TEST-004</a>	Supplier	CDG, Inc	PITTSBURGH		Draft
<a href="#">SD-PERFORMANCE-TEST-003</a>	Supplier	CDG, Inc	PITTSBURGH		Active
<a href="#">SD-PERFORMANCE-TEST-002</a>	Supplier	CDG, Inc	PITTSBURGH		Active
<a href="#">SD-PERFORMANCE-TEST-001</a>	Supplier	CDG, Inc	PITTSBURGH		Active
<a href="#">test10</a>	Supplier	SRT supplier moac	24 NORTH AVENUE		Draft
<a href="#">333</a>	Supplier	SRT supplier moac	57 BERLINGTON R		Draft
<a href="#">111</a>	Supplier	SRT supplier moac	145 COMMERCIAL		Draft
<a href="#">test1</a>	Supplier	SRT supplier moac	24 NORTH AVENUE		Draft

- Notice that the Request Number BPEL-1 entered earlier is displayed in the list. Click the request number BPEL-1 link to open the Ship and Debit Request Details page for the created request. Verify the details.

## Verifying the Ship and Debit Request Details

**ORACLE** Trade Management

Home Trade Management Direct-ADS-Disco Profile Sign

Customer Product Quota Budget Trade Planning Claim Indirect Sales Management Administration Calendar Supplier Ship and Debit Bu

Requests Batch Creation Batch Summary

Quick Find Offer  Go

Logged In As TRADEMG

---

**Ship and Debit Request Details**

\* Indicates required field

Cancel Save Apply Copy Report

Accrual Type Request Type * Request Number Supplier Contact Supplier Contact Email Address * Requestor Assignee Start Date End Date Request Currency Sales Order Currency Description	Supplier <b>Bid Request</b> BPFL-1  sdr-supplier@testing.com Sonneshein, Mr. Evans Sonneshein, Mr. Evans 18-Aug-2008 18-Oct-2008 US dollar  Ship and debit request from BPFL	Operating Unit Authorization Number * Supplier Vision Operations Abbott Laboratories, Inc. 1015 Sansome St, SAN FRANCISCO, CA 94111 Supplier Contact Phone Number 2255 Supplier Response By Date Assignee Response By Date Status Request Outcome Supplier Quota Number Internal Order Number <input checked="" type="checkbox"/> Request Only
--	---	---

Created By: Sonneshein, Mr. Evans, Creation Date: 2009-09-15, Last Update By: Sonneshein, Mr. Evans

Product Customer Offers Claims Notes & Attachments

**Price Request Details**

Views

View All Lines Go Personalize

---

# Using XML Gateway Inbound and Outbound Interfaces

## Overview

Oracle E-Business Suite Integrated SOA Gateway provides a communication infrastructure between Oracle E-Business Suite and Web consumers. Inbound and outbound XML data is exchanged between the consumers and Oracle E-Business Suite through Oracle XML Gateway.

Oracle XML Gateway provides a common, standards-based approach for XML integration. XML is key to an integration solutions, as it standardizes the way in which data is searched, exchanged, and presented thereby enabling interoperability throughout the supply chain.

Oracle XML Gateway provides a set of services that can be easily integrated with Oracle E-Business Suite to support XML messaging. It uses the message propagation feature of Oracle Advanced Queuing to integrate with Oracle Transport Agent to deliver outbound XML messages to and receive inbound XML messages or transactions from business partners.

To enable bidirectional integration with Oracle E-Business Suite and consumers, Oracle E-Business Suite Integrated SOA Gateway supports XML Gateway Map interface type through the following approaches:

- For an inbound XML Gateway Map interface, once a Web service of an inbound XML Gateway interface is deployed, the deployed service representing in WSDL can be used in creating a BPEL process to insert inbound data into Oracle E-Business Suite.
- For an outbound XML Gateway Map interface, since an outbound message is first enqueued to the ECX\_OUTBOUND queue, Oracle E-Business Suite Integrated SOA Gateway supports it through subscription model by first dequeuing the message to retrieve outbound data from Oracle E-Business Suite by using a BPEL process. The

retrieved data can then be passed to trading partners or consumers who subscribed to the message.

To better understand how to use a deployed Web service of an inbound XML Gateway interface as well as understand how the subscription model works for an outbound XML Gateway, the following topics are introduced in this chapter:

- Using XML Gateway Inbound Web Services, page 4-2
  - Using XML Gateway Inbound Services at Design Time, page 4-2
  - Deploying and Testing the BPEL Process at Run Time, page 4-26
- Using XML Gateway Outbound Through Subscription Model, page 4-30
  - Using XML Gateway Outbound Messages in Creating a BPEL Process at Design Time, page 4-30
  - Deploying and Testing a BPEL Process at Run Time, page 4-28

For the examples described in the following sections, we use Oracle JDeveloper 10.1.3.3.0 as a design-time tool to create the BPEL processes and use Oracle SOA Suite BPEL server 10.1.3.3.0 for the process deployment.

## Using XML Gateway Inbound Services

This section includes the following topics:

- Using XML Gateway Inbound Services at Design Time, page 4-2
- Deploying and Testing the BPEL Process at Run Time, page 4-26

## Using XML Gateway Inbound Services at Design Time

### BPEL Process Scenario

Take the XML Gateway Inbound Process PO XML Transaction as an example to explain the BPEL process creation. In this example, the XML Gateway inbound message map is exposed as a Web service through `PROCESS_PO_007` inbound map. It allows sales order data including header and line items to be inserted into Order Management system while an associated purchase order is created.

When a purchase order is sent by a trading partner, the purchase order data is used as input to the BPEL process along with ECX Header properties such as `MESSAGE_TYPE`, `MESSAGE_STANDARD`, `TRANSACTION_TYPE`, `TRANSACTION_SUBTYPE`, `PARTY_SITE_ID`, and `DOCUMENT_NUMBER`. The BPEL process then pushes this purchase order in `ECX_INBOUND` queue. Agent Listeners running on `ECX_INBOUND` would enable further processing by the Execution Engine. Oracle XML Gateway picks



this XML message, does trading partner validation, and inserts order data to Order Management Application.

If the BPEL process is successfully executed after deployment, you should get the same order information inserted into the Order Management table once a purchase order is created.

### **Prerequisites to Configure a BPEL Process Using an XML Gateway Inbound Service**

Before performing the design-time tasks for XML Gateway Inbound services, you need to ensure the following tasks are in place:

- An integration repository administrator needs to successfully deploy the XML Gateway Inbound message map to the application server.
- An integration developer needs to locate and record the deployed WSDL URL for the inbound message map exposed as a Web service.
- XML Gateway header variables need to be populated for XML transaction.
- Agent listeners need to be up and running.

### ***Deploying XML Gateway Inbound WSDL URL***

An integration repository administrator must first create a Web service for the selected XML Gateway inbound map, and then deploy the service from Oracle Integration Repository to the application server.

For example, the administrator must perform the following steps before letting the integration developers use the deployed WSDL in creating a BPEL process:

1. To generate a Web service, locate the interface definition first (such as an XML Gateway inbound interface `INBOUND:Process Purchase Order XML Transaction (ONT:POI)`) and click **Generate WSDL** in the interface details page.

Once the service is successfully generated, the Web Service - SOA Provider region appears in the interface details page.

**Note:** Since XML Gateway Map interface types can be service enabled by Web Service Provider (for Oracle E-Business Suite Release 12.0) and SOA Provider (after the Release 12.0), you may also find the Web Service - Web Service Provider region as well if there are Web services generated through Web Service Provider from Oracle E-Business Suite Release 12.0.

However, there is no design difference in terms of creating BPEL processes using XML Gateway inbound services whether the services are enabled through SOA Provider or Web Service Provider.

For detailed instruction on how to generate a Web service, see *Generating Web Services, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

2. To deploy a generated Web service, select at least one authentication type and click **Deploy** in the Web Service - SOA Provider region of the interface details page to deploy the service.

Once the service is successfully deployed, the selected authentication type(s) will be displayed along with 'Deployed' Web Service Status. For more information on securing Web services with authentication types, see *Managing Web Service Security, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

For information on how to deploy a Web service, see *Deploying, Undeploying, and Redeploying Web Services, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

### ***Searching and Recording WSDL URL***

An integration developer also needs to log on to the system to locate and record the deployed Web service WSDL URL for the inbound message map.

This WSDL information will be used later in creating a partner link for the inbound map exposed as a Web service during the BPEL process creation at design time.

## Confirming and Recording a Deployed WSDL URL

The screenshot displays the Oracle Integration Repository web application. The top navigation bar includes links for Home, Logout, Preferences, Help, and Diagnostics. The main content area shows the details for an XML Gateway named 'INBOUND: Process Purchase Order XML Transaction'. Key information includes: Internal Name (ONT:POI), Type (XML Gateway Map), Product (Order Management), Status (Active), Business Entity (Sales Order), Standard (OAG 7.2 Process\_PO\_007), and Standard Ready (ROSETTANET:02.02.00:PIP3A4-Request Purchase Order). The 'Full Description' section explains that this message map supports the inbound Process PO (Purchase Order Processing) XML transaction. The 'Web Service - SOA Provider' section shows the Web Service Status as 'Deployed' and the WSDL URL as 'View WSDL'. The 'Authentication Type' section indicates that 'SAML Token (Sender Vouches)' is selected. The 'Source Information' section lists the Source File as 'patch/115/xml/US/ONT\_3A4R\_OAG72\_IL.xgm', Source Version as '120.3', and Source Product as 'ONT'. The 'Methods' section contains a table with one entry: POI, POI, Active, Process transaction. The bottom of the page features a footer with links for About this Page, Privacy Statement, and Copyright information (© 2009, Oracle. All rights reserved).

ORACLE Integration Repository

Home Logout Preferences Help Diagnostics

Integration Repository >

XML Gateway : INBOUND: Process Purchase Order XML Transaction

Browse Search Printable Page

Internal Name: ONT:POI  
Type: XML Gateway Map  
Product: Order Management  
Status: Active  
Business Entity: Sales Order  
Standard: OAG 7.2 Process\_PO\_007  
Standard Ready: ROSETTANET:02.02.00:PIP3A4-Request Purchase Order

Scope: Public  
Interface Source: Oracle

**Full Description**

This is the message map to support the inbound Process PO (Purchase Order Processing) XML transaction which, through the population of data in the header and line open interface tables, allows users to create sales orders in the Order Management system. The OAG name for this message is PROCESS PO.

**Web Service - SOA Provider**

Web Service Status: Deployed  
WSDL: View WSDL

\* Authentication Type: ☐ Username Token ☒ SAML Token (Sender Vouches)

**Source Information**

Source File: patch/115/xml/US/ONT\_3A4R\_OAG72\_IL.xgm  
Source Version: 120.3  
Source Product: ONT

**Methods**

Name	Internal Name	Status	Description
POI	POI	Active	Process transaction.

Browse Search Printable Page

Integration Repository Home Logout Preferences Help Diagnostics

About this Page Privacy Statement

Copyright (c) 2009, Oracle. All rights reserved.

How to search for an interface and review the interface details, see Searching and Viewing Integration Interfaces, page 2-1.

### Populating XML Gateway Header Variables

You need to populate certain variables in the BPEL PM in order to provide XML Gateway header information for Oracle E-Business Suite. The MESSAGE\_TYPE, MESSAGE\_STANDARD, TRANSACTION\_TYPE, TRANSACTION\_SUBTYPE, DOCUMENT\_NUMBER and PARTY\_SITE\_ID are the mandatory header variables that you need to populate for the XML transaction to complete successfully.

Refer to Adding an Assign activity, page 4-21 for more information.

### Ensuring Agent Listeners Are Up and Running

You also need to ensure that listeners on the ECX\_INBOUND, ECX\_TRANSACTION queues are up and running. Use the following steps to configure these listeners in Oracle E-Business Suite:

1. Log in to Oracle E-Business Suite with the Workflow Administrator responsibility.
2. Click the **Workflow Administrator Web Applications** link from the Navigator.
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** if they are not up and running.

### **BPEL Process Creation Flow**

After deploying the BPEL process, you should get the same order information inserted into the Order Management table once a purchase order is created.

Based on the XML Gateway Inbound Process PO XML Transaction business scenario, the following design-time tasks are discussed in this chapter:

1. Create a new BPEL project, page 4-7  
Use this step to create a new BPEL project called `XMLGatewayInbound.bpel` using an Synchronous BPEL Process template. This automatically creates two dummy activities - Receive and Reply - to receive input from a trading partner and to reply output of the BPEL process back to the request application.
2. Create a Partner Link, page 4-9  
Use this step to create a partner link to allow the inbound message to be inserted to the Oracle E-Business Suite.
3. Add a Partner Link for File Adapter, page 4-12  
Use this step to add a partner link for File Adapter in order to pick up an XML file received from the trading partner to get the XML message.
4. Add Invoke activities, page 4-18  
Use this step to add two Invoke activities in order to:
  1. To get the XML message details that is received from the Receive activity.
  2. To enqueue the purchase order information to the `ECX_INBOUND` queue.
5. Add Assign activities, page 4-21  
Use this step to create two Assign activities in order to:
  1. To pass XML message obtained from the first Invoke activity to the second Invoke activity.
  2. To pass ECX header variables to the second Invoke activity as input variables.

For general information and basic concept of a BPEL process, see *Understanding BPEL Business Processes*, page D-1 and *Oracle BPEL Process Manager Developer's Guide* for details.

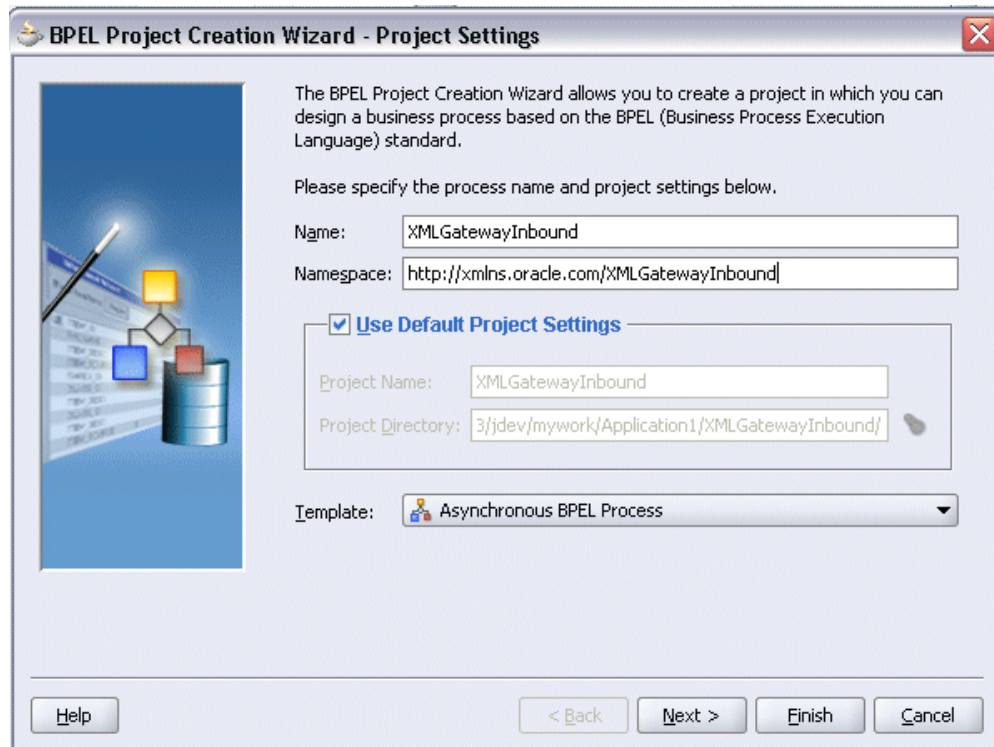
## Creating a New BPEL Project

Use this step to create a new BPEL project that will contain various BPEL process activities.

### 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 produces a list of available categories.
4. Expand the **General** node and then select **Projects**.
5. Select **BPEL Process Project** from the **Items** group.
6. Click **OK**. The BPEL Process Project dialog box appears.

### Entering BPEL Project Information



The BPEL Project Creation Wizard allows you to create a project in which you can design a business process based on the BPEL (Business Process Execution Language) standard.

Please specify the process name and project settings below.

Name:

Namespace:

☒ Use Default Project Settings

Project Name:

Project Directory:

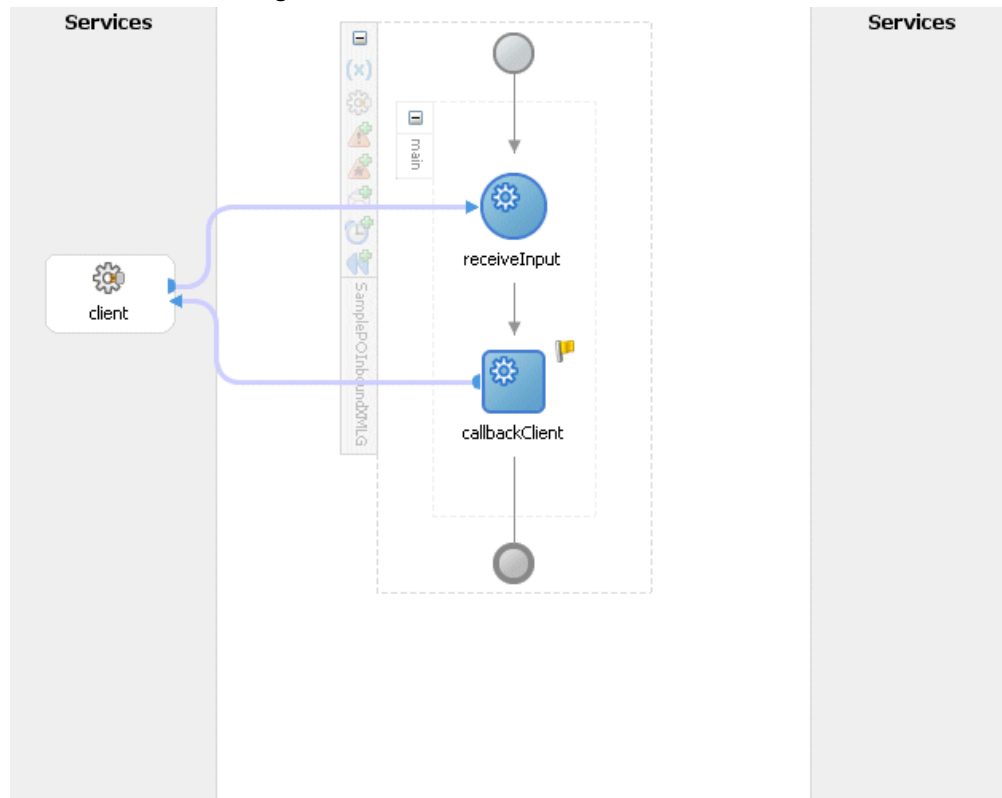
Template:

Buttons: Help, < Back, Next >, Finish, Cancel

7. In the **Name** field, enter a descriptive name such as XMLGatewayInbound.
8. From the Template list, select **Asynchronous BPEL Process** and then select **Use Default Project Settings**.
9. Use the default input and output schema elements in the Input/Output Elements dialog box.
10. Click **Finish**.

A new asynchronous BPEL process is created with the Receive and Callback activities. The required source files including `bpel.xml`, using the name you specified (for example, `XMLGInbound.bpel`) are also generated.

### New BPEL Process Diagram



### Creating a Partner Link

Use this step to create a Partner Link called `ONT_POI` to insert sales order data to Oracle E-Business Suite.

#### To create a partner link to insert sales data to Oracle E-Business Suite:

1. In JDeveloper BPEL Designer, drag and drop the **PartnerLink** service from the Component Palette into the Partner Link border area of the process diagram. The Service Name dialog box appears.
2. Copy the WSDL URL corresponding to the XML Gateway inbound map `INBOUND:Process Purchase Order XML Transaction (ONT:POI)` that you recorded earlier in the WSDL File field.

A Partner Link Type message dialog box appears asking whether you want the system to create a new WSDL file that will by default create partner link types for you.

Click **Yes** to have the Partner Name value populated automatically. The name is defaulted to `ONT_POI`.

Select Partner Role and My Role fields from the drop-down lists.

#### Create Partner Link

**Edit Partner Link**

General Image Property

Name: ONT\_\_POI

Process: XMLGatewayInbound

**WSDL Settings**

WSDL File: file:/D:/JDev10133/jdev/mywork/Application1/XMLGat

Partner Link Type: ONT\_\_POI\_PortType\_PL

Partner Role: ONT\_\_POI\_PortType\_Role

My Role: ONT\_\_POI\_PortType\_Role

Help Apply OK Cancel

Click **Apply**.

3. Select the Property tab and click the **Create Property** icon to select the following properties from the property name drop-down list in order to pass the security header along with the SOAP request:

- wsseUsername

Specify the username to be passed in the Property Value box.

- wssePassword

Specify the corresponding password for the username to be passed in the Property Value box.

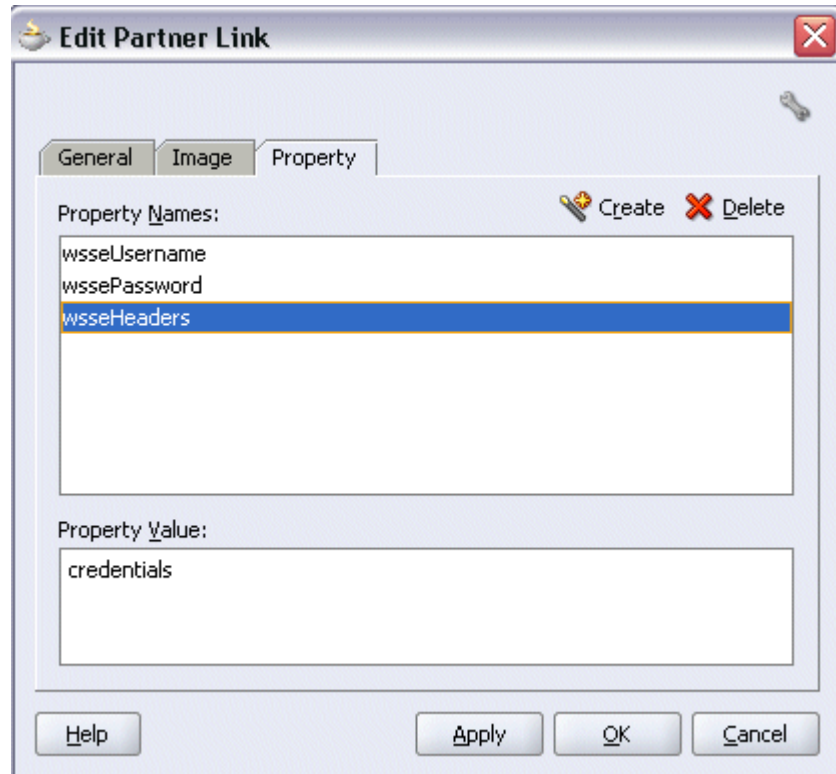
- wsseHeaders

Enter `credentials` as the property value.

Click **Apply** to save the selected property values.

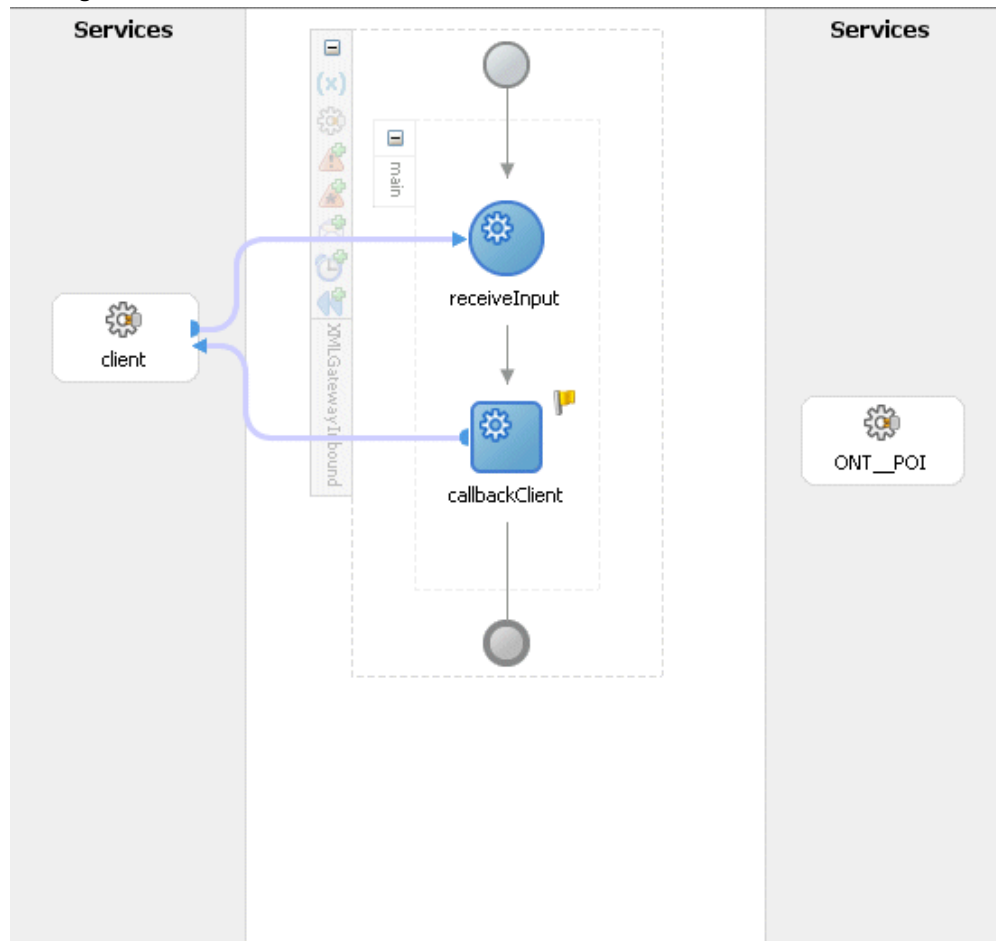


#### Adding Properties



4. 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.

### Adding the Partner Link



### Adding Partner Links for File Adapter

Use this step to configure a BPEL process by adding a partner link for File Adapter to get the XML Message.

#### To add the Partner Link for File Adapter to get the XML Message:

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 welcome page appears.
2. Click **Next**. The Service Name dialog box appears.
3. Enter a name for the file adapter service, such as `GetXMLMsg`. You can add an optional description of the service.
4. Click **Next** and the Operation dialog box appears.

### Specifying the Operation

**Adapter Configuration Wizard - Step 2 of 5: Operation**

The File Adapter supports three operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, and a Synchronous Read File operation that reads the current contents of a file. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type:

☐ Read File

☐ Write File

☒ Synchronous Read File

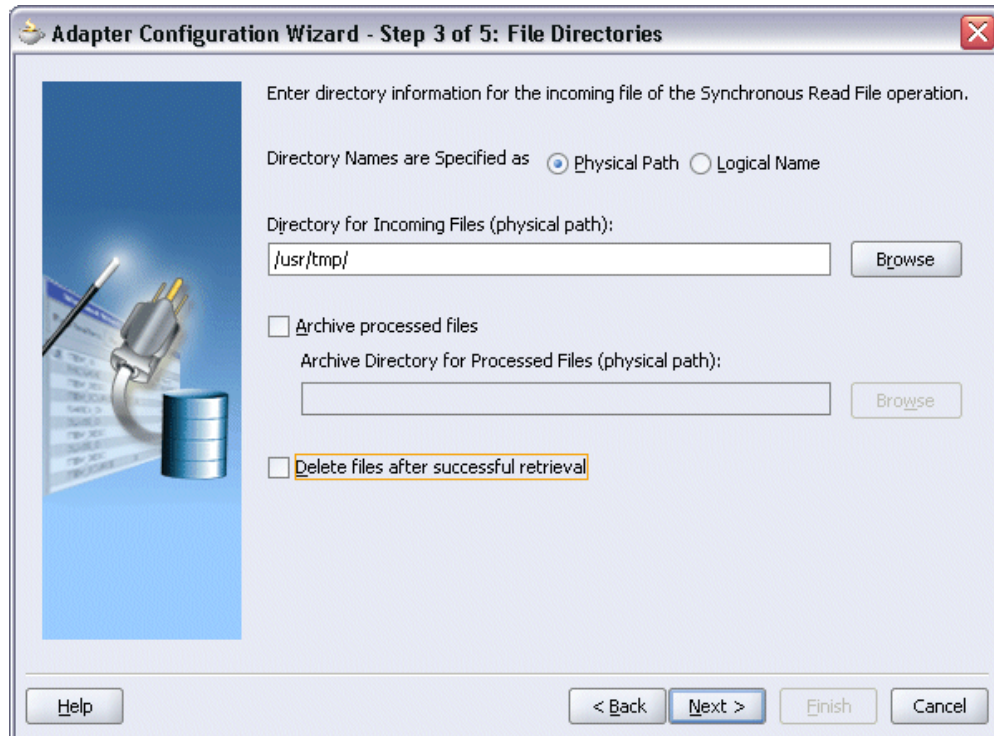
Operation Name:

Help < Back Next > Finish Cancel

5. Specify the operation type, for example **Synchronous Read File**. This automatically populates the **Operation Name** field.

Click **Next** to access the File Directories dialog box.

### Configuring the Input File



6. Select **Physical Path** radio button and enter the physical path for incoming file directory information. For example, enter `/usr/tmp/`.

**Note:** To be able to locate the file from the physical directory you specified here, you must first place the input payload file (such as `order_data_xmlg.xml`) to the specified directory.

Alternatively, click **Browse** to locate the incoming file directory information.

Uncheck the **Delete files after successful retrieval** check box. Click **Next**.

7. Enter the name of the file for the synchronous read file operation. For example, enter `order_data_xmlg.xml`. Click **Next**. The Messages dialog box appears.

8. Select **Browse** to open the Type Chooser.

Click **Import Schema Files** button on the top right corner of the Type Chooser window. This opens the Import Schema File pop-up window.

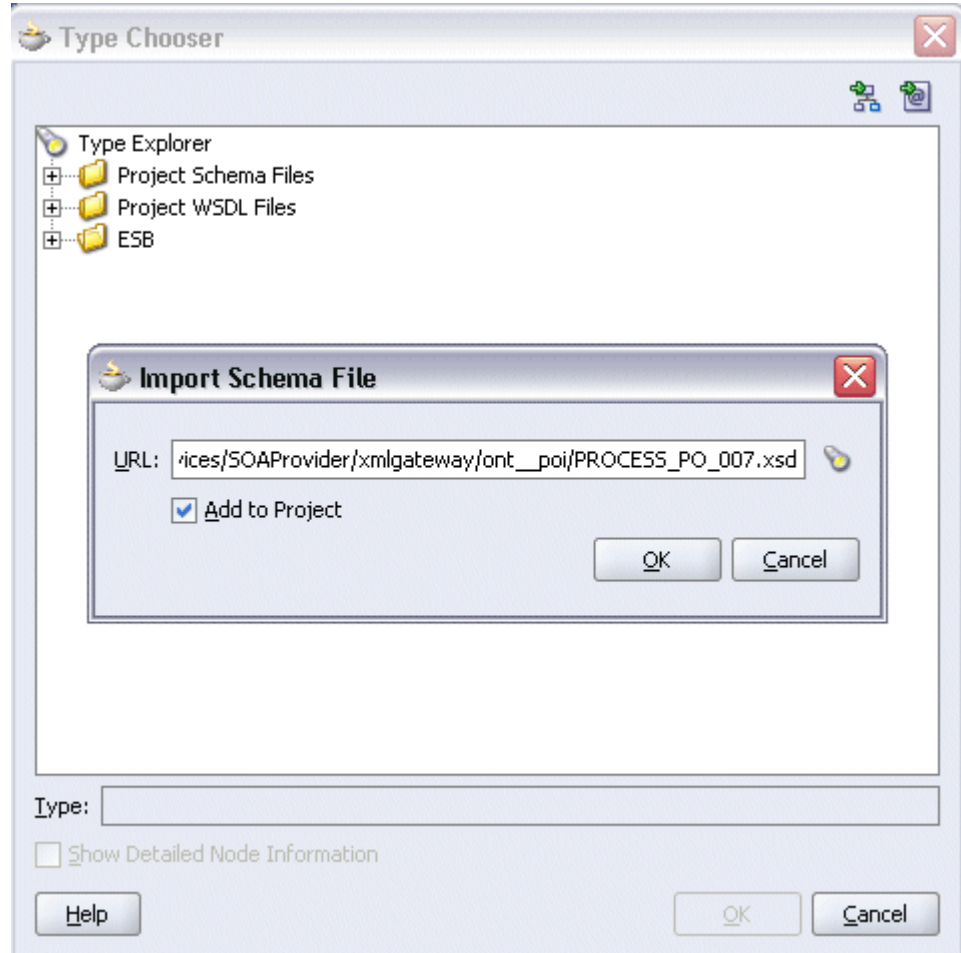
Enter the schema location for the service. Such as `http://<myhost>:<port>/web services/SOAPProvider/xmlgateway/ont__poi/PROCESS_PO_007.xsd`.

Schema location for your service can be found from the service WSDL URL (for

example,

http://<myhost>:<port>/webservices/SOAPProvider/xmlgateway/ont\_\_poi/?wsdl).

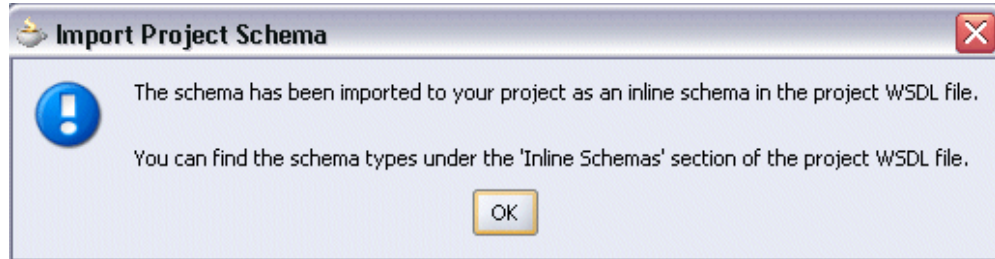
### Importing Project Schema



Select the **Add to Project** check box and click **OK**.

9. Click **OK** to the Import Project Schema message prompt.

### Importing Project Schema Message Prompt



The Imported Schemas folder is automatically added to the Type Chooser window.

10. Expand the Imported Schemas folder and select **PROCESS\_PO\_007.xsd** > **PROCESS\_PO\_007**. Click **OK**.

The selected xsd is displayed as Schema Location, and PROCESS\_PO\_007 is selected as Schema Element.

11. Click **OK** to populate the selected values in the Messages dialog box.

### Populating Selected Message Schema and Element

Define the message for the Synchronous Read File operation. Specify the Schema File Location and select the Schema Element that defines the messages in the incoming files. Use the Browse button to find an existing schema definition. If you check 'Schema is Opaque', then you do not need to specify a Schema.

**Message Schema**

☐ Native format translation is not required (Schema is Opaque) Define Schema for Native Format

Schema Location:  Browse

Schema Element:

Help < Back Next > Finish Cancel

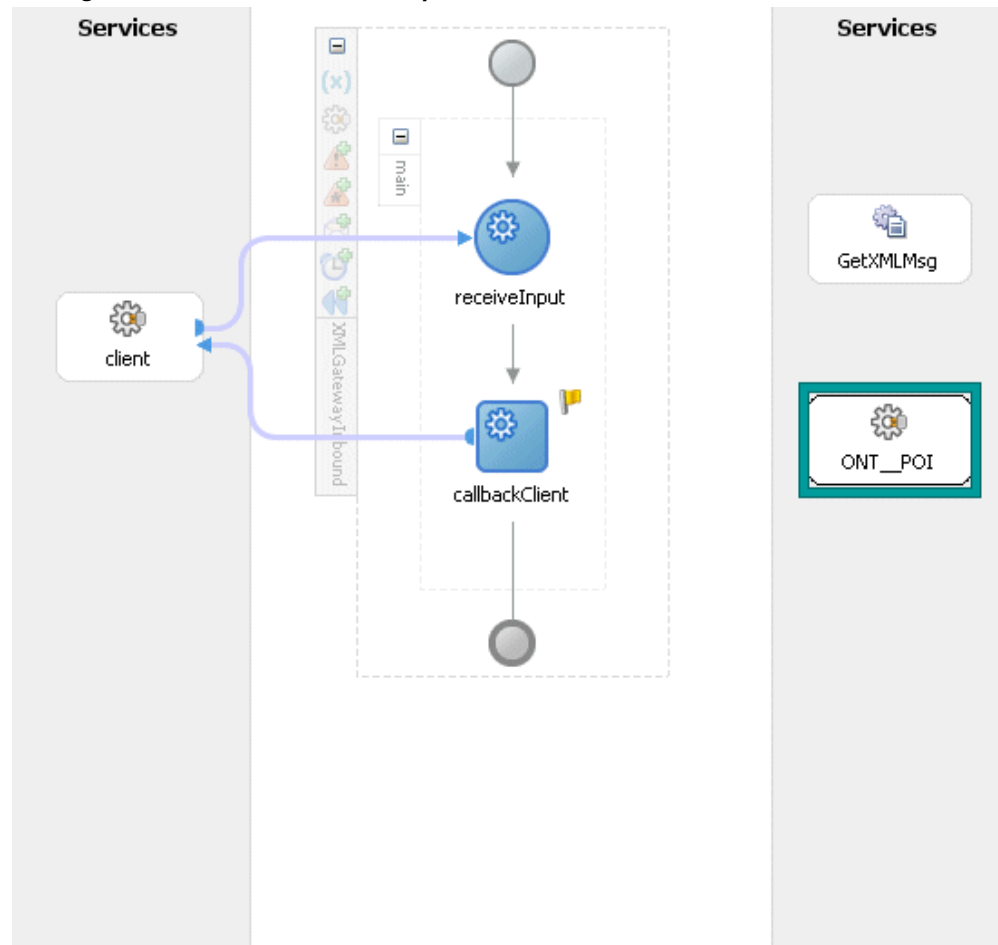
12. Click **Next** and then **Finish**. The wizard generates the WSDL file corresponding to the partner link. The main Create Partner Link dialog box appears, specifying the new WSDL file `GetXMLMsg.wsdl`.

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

The `GetXMLMsg` Partner Link appears in the BPEL process diagram.



### Adding the Partner Link for File Adapter



### Adding Invoke Activities

This step is to configure three Invoke activities:

1. To get the XML message details that is received from the Receive activity by invoking the `GetXMLMsg` partner link in an XML file.
2. To enqueue the purchase order information to the `ECX_INBOUND` queue by invoking `ONT_POI` partner link in an XML file.

**To add the first Invoke activity for a partner link to get XML message:**

1. In JDeveloper BPEL Designer, drag and drop the first **Invoke** activity from the Component Palette into the Activity box of the process diagram, between the **Receive** and **Callback** activities.
2. Link the Invoke activity to the `GetXMLMsg` service. The Edit Invoke dialog box



appears.

3. Enter a name for the Invoke activity and then click the **Create** icon next to the **Input Variable** field to create a new variable. The Create Variable dialog box appears.
4. Select **Global Variable** and then enter a name for the variable. You can also accept the default name. Click **OK**.
5. Enter a name for the Invoke activity and click the **Create** icon next to the **Output Variable** field to create a new variable. The Create Variable dialog box appears.
6. Select **Global Variable**, and enter a name for the variable. You can also accept the default name. Click **OK**.

#### *Editing Invoke Activity*

The screenshot shows the 'Invoke' dialog box with the following details:

- Title:** Invoke
- Tabs:** General, Correlations, Sensors, Adapters, Annotations (General is selected).
- Name:** Invoke
- Partner Role Web Service Interface:**
  - Partner Link:** GetXMLMsg
  - Operation:** SynchRead
- Input Variable:** Invoke\_SynchRead\_InputVariable
- Output Variable:** Invoke\_SynchRead\_OutputVariable
- Buttons:** Help, Apply, OK, Cancel

Click **Apply** and then **OK** in the Edit Invoke dialog box to finish configuring the Invoke activity.

The first Invoke activity appears in the process diagram.

#### **To add the second Invoke activity for a partner link to enqueue PO information:**

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

**Invoke** and **Callback** activities.

2. Link the Invoke activity to the `ONT_POI` service. The Edit Invoke dialog box appears.
3. Enter a name for the Invoke activity and then click the **Create** icon next to the **Input Variable** field to create a new variable. The Create Variable dialog box appears.
4. Select **Global Variable** and then enter a name for the variable. You can also accept the default name. Click **OK**.

#### *Editing Invoke Activity*

The screenshot shows the 'Invoke' dialog box with the following configuration:

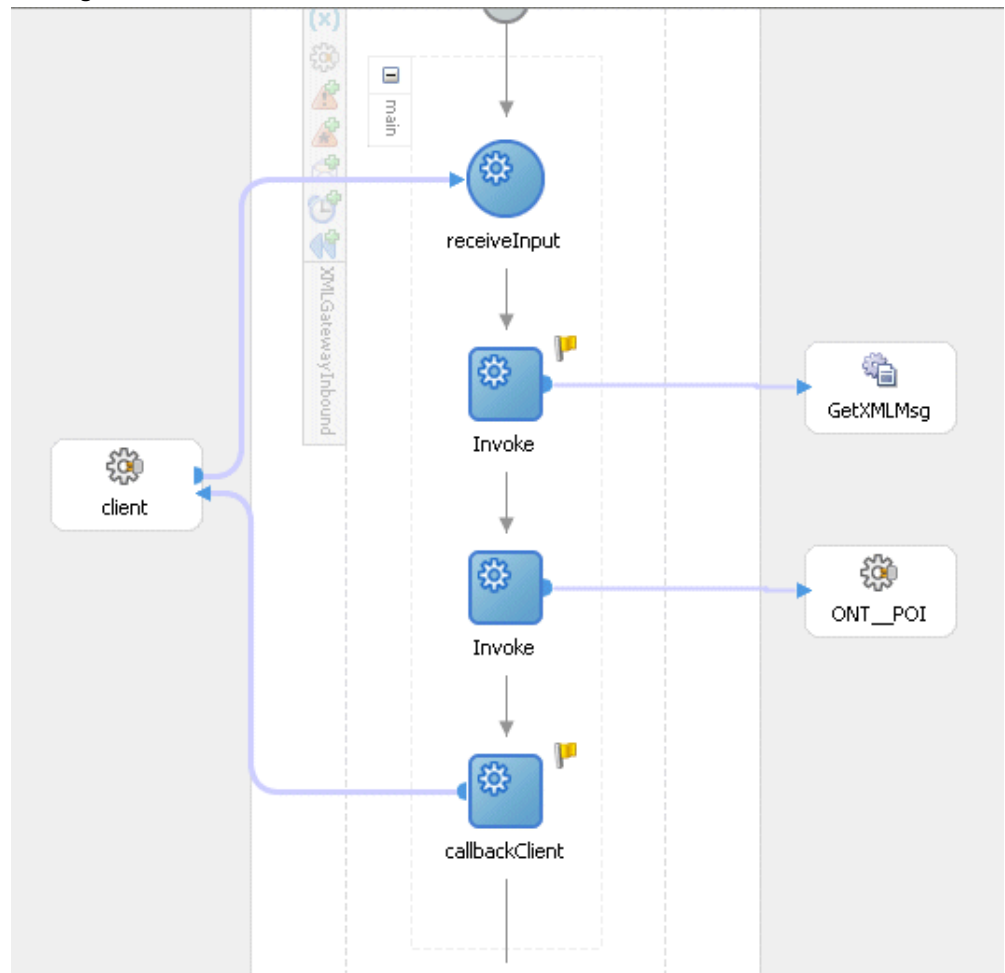
- Name:** Invoke
- Partner Role Web Service Interface:**
  - Partner Link:** ONT\_POI
  - Operation:** PROCESSPO
- Input Variable:** Invoke\_PROCESSPO\_InputVariable
- Output Variable:** (empty)

Buttons at the bottom: Help, Apply, OK, Cancel.

Click **Apply** and then **OK** in the Edit Invoke dialog box to finish configuring the Invoke activity.

5. The process diagram appears.

### Adding Invoke Activities



### Adding Assign Activities

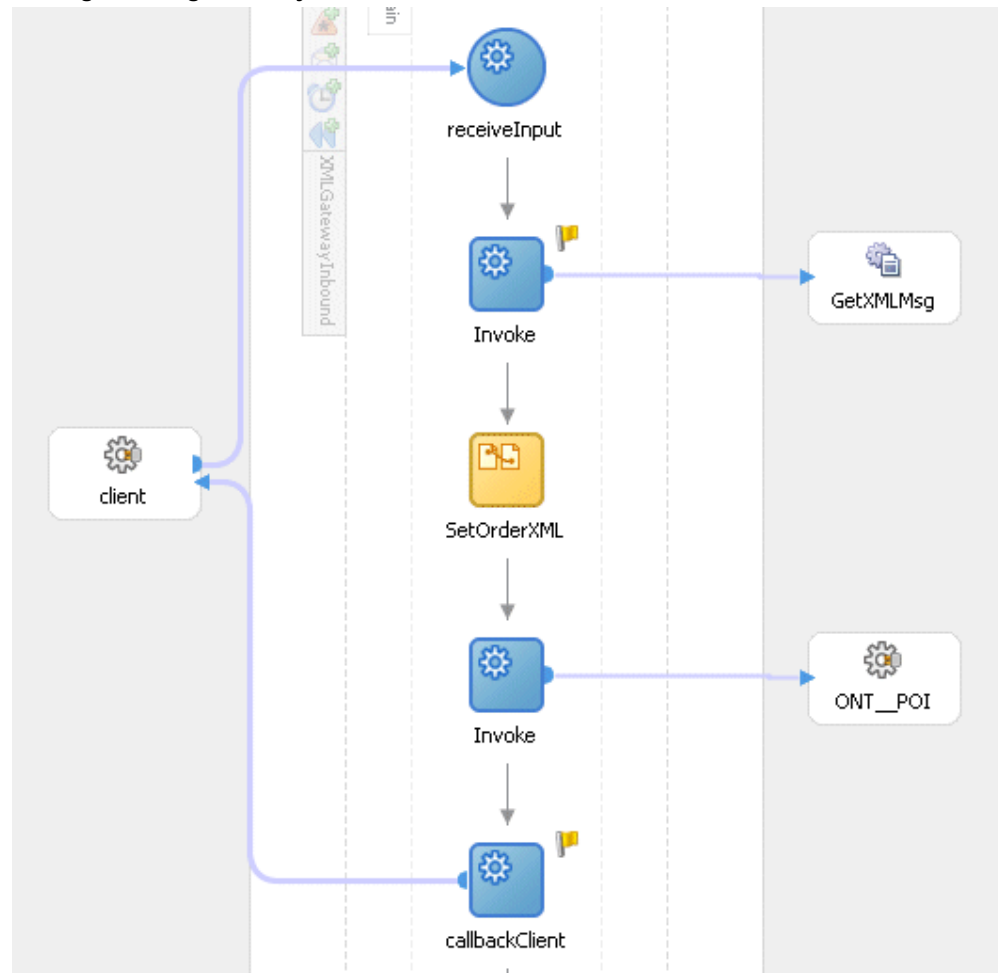
This step is to configure two Assign activities:

1. To pass XML message as an input to the Invoke activity for enqueueing message.
2. To pass XML Gateway header variables as input variables to the Invoke activity in order to provide context information for Oracle E-Business Suite.

**To add the first Assign activity to pass XML message as input to the Invoke 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 two **Invoke** activities.

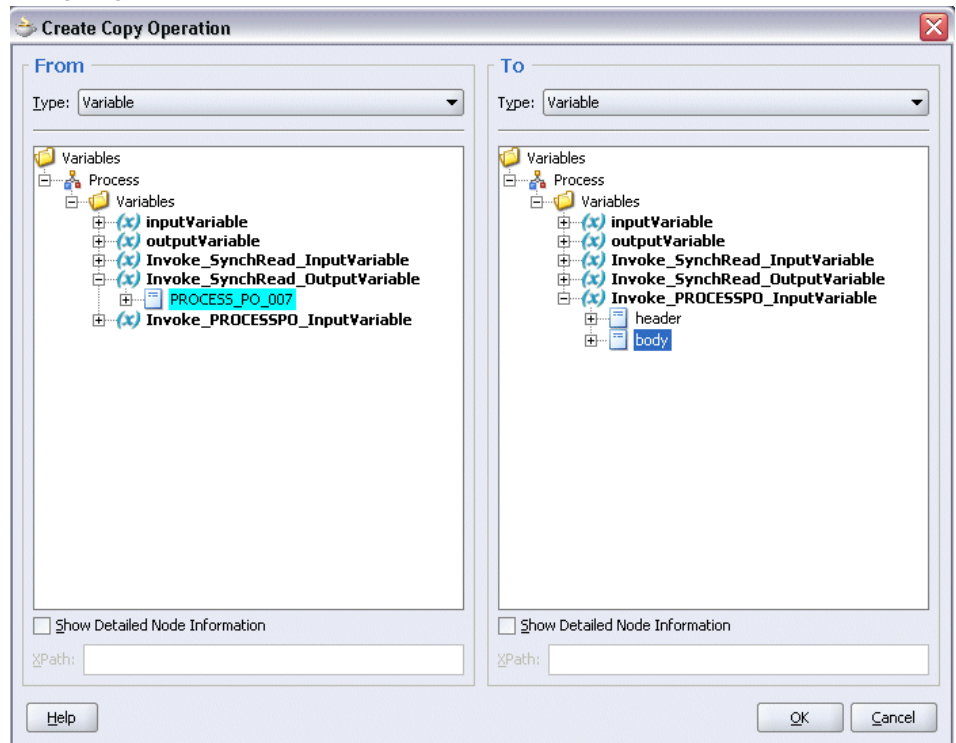
### Adding an Assign Activity



2. Double-click the **Assign** activity to access the Edit Assign dialog box.
3. Click the General tab to enter the name for the Assign activity, such as 'SetOrderXML'.
4. On the Copy Operation tab, click **Create** and then select **Copy Operation** from the menu. The Create Copy Operation window appears.
5. Enter the following information:
  - In the From navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Invoke\_SynchRead\_OutputVariable** and select **Process\_PO\_007**.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process >**

Variables > Invoke\_PROCESSPO\_InputVariable and select **body**.

### Assigning Parameters

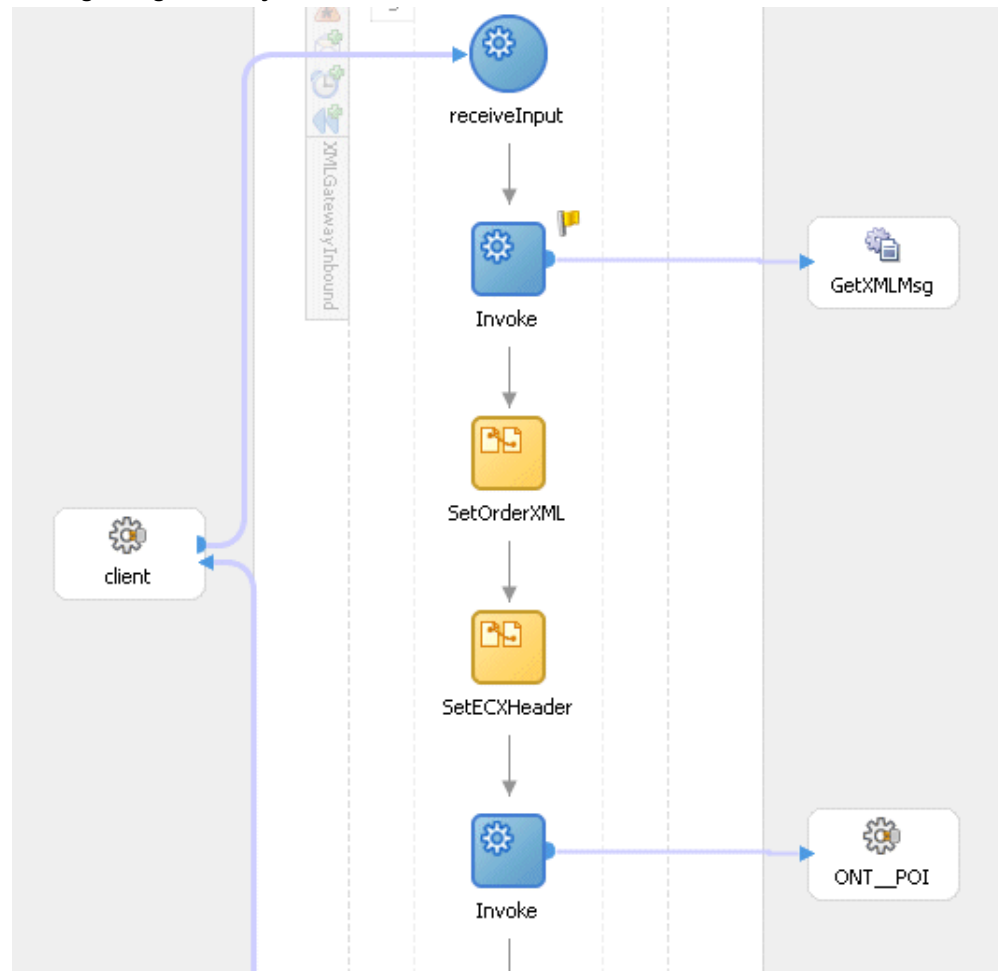


- Click **OK**. The Edit Assign dialog box appears.
6. Click **Apply** and then **OK** to complete the configuration of the Assign activity.

**To add the second Assign activity to pass XML Gateway header variables to the Invoke activity:**

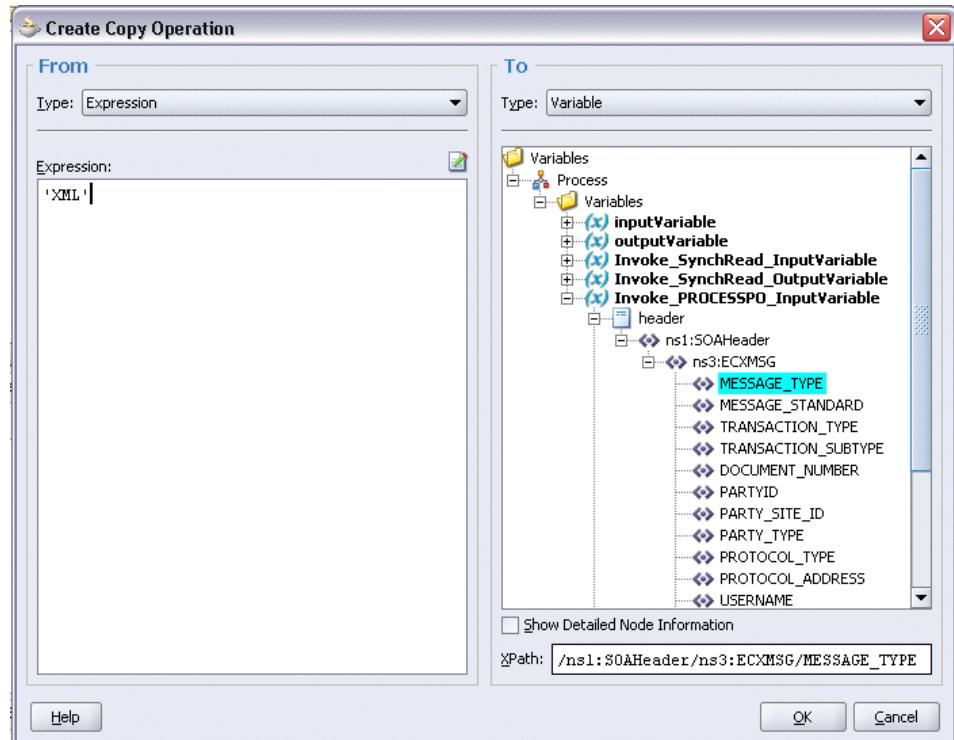
1. Add the second Assign activity by dragging and dropping the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the SetOrderXML **Assign** activity and the second **Invoke** activity.

#### Adding Assign Activity



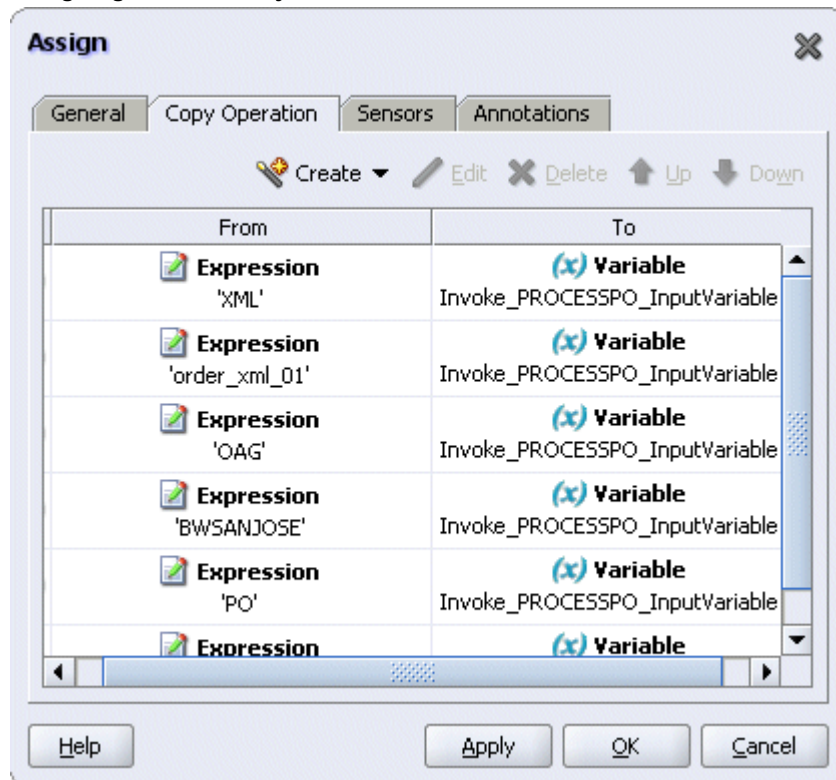
2. Repeat Step 2 to Step 4 described in creating the first Assign activity to add the second Assign activity called 'SetECXHeader'.
3. On the Copy Operation tab, click **Create** and then select **Copy Operation** from the menu. The Create Copy Operation window appears.
4. Enter the first pair of parameters:
  - In the From navigation tree, select type Expression and then enter 'XML' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Invoke\_PROCESSPO\_InputVariable > header > ns1:SOAHeader > ns3:ECXMSG** and select **MESSAGE\_TYPE**.  
The XPath field should contain your selected entry.

### Assigning XML Gateway Header Parameter Value



- Click **OK**.
5. Use the same mechanism described in step 3 and 4 to enter the following additional parameters:
- MESSAGE\_STANDARD: 'OAG'
  - TRANSACTION\_TYPE: 'PO'
  - TRANSACTION\_SUBTYPE: 'PROCESS'
  - DOCUMENT\_NUMBER: 'order\_xml\_01'
  - PARTY\_SIDE\_ID: 'BWSANJOSE'

### Assigning XML Gateway Header Parameters



6. Click **Apply** and **OK** to complete the configuration of the Assign activity.

## Deploying and Testing the BPEL Process at Run Time

After creating a BPEL process using the WSDL URL generated from the XML Gateway inbound message map interface definition, you can deploy it to a BPEL server if needed. To ensure that this process is modified or orchestrated appropriately, you can also manually test the BPEL process by initiating the business process contained in the BPEL process to test the interface integration.

### Prerequisites

Before deploying the BPEL process using Oracle JDeveloper, you must ensure that you have established the connectivity between the design-time environment and the run-time servers including the application server and the integration server.

How to configure the necessary server connection, see *Configuring Server Connection*, page B-1.

To validate your BPEL process, perform the following run-time tasks:

1. Deploy the BPEL process, page 4-27



Once you deploy the process to a BPEL server, it becomes available so that you can run the process manually to test it for validation.

## 2. Test the BPEL process, page 4-28

After deploying a BPEL process, you can manage the process from the BPEL console to validate the interface integration contained in your BPEL process.

## Deploying the BPEL Process

You must deploy the BPEL process (`XMLGatewayInbound.bpel`) that you created earlier before you can run it. The BPEL process is first compiled and then deployed to the BPEL server.

**Note:** Before deploying the BPEL Process for XML Gateway Inbound service, you should:

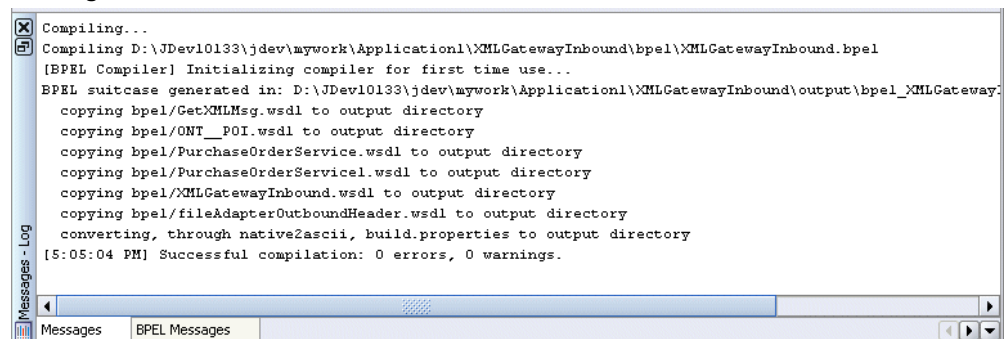
- Load the `order_data_xmlg.xml` file into the specified directory `'/usr/tmp/'` folder of SOA Suite server (or `D:\HOL` in case of SOA Server in Windows machine).
- Edit the input file `order_data_xmlg.xml` by entering values for `<REFERENCEID>` and `<POID>` such as `'order_xml_01'`.

### To deploy the BPEL process:

1. In the Applications Navigator of JDeveloper BPEL Designer, select the **XMLGInbound** project.
2. Right-click the project and click **Make** action from the menu.

Look for any compilation error messages in Message Log.

### Messages Window



Right-click the project and select **Deploy > Integration Server Connection name > Deploy to Default Domain** action from the menu.

For example, you can select **Deploy > BPELServerConn > Deploy to Default Domain** to deploy the process if you have the BPEL Process Manager setup appropriately.

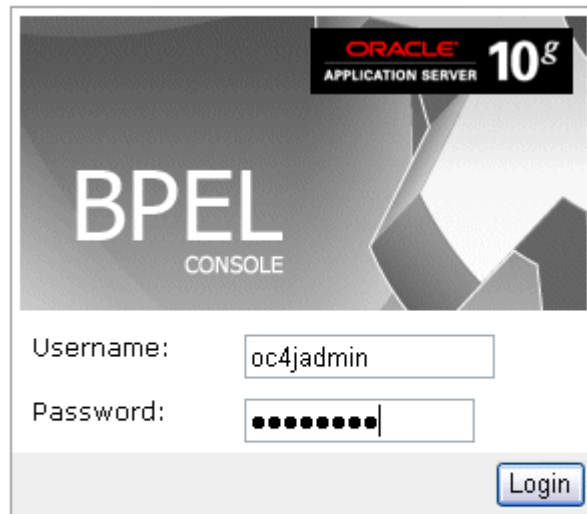
3. Look for 'Build successful' message in Apache Ant – Log to ensure that the BPEL project is compiled and successfully deployed.

## 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 Application Server 10g BPEL Console (`http://<soaSuiteServerHostName>:<port>/BPELConsole`). The BPEL Console login page appears.



You can also manage the list of domains using BPEL Admin:

➔ [Goto BPEL Admin](#)

2. Enter the username and password and click **Login**.  
The Oracle Enterprise Manager 10g BPEL Control appears.
3. In the BPEL Console, confirm that XMLGINbound has been deployed.
4. Click the XMLGINbound link to open the Initiate tab
5. Click **Post XML Message** to initiate the process.

## Verifying Records in Oracle E-Business Suite

Once the BPEL process is successfully initiated and completed, you can validate it through the relevant module in Oracle E-Business Suite.

### To validate it in Oracle Transaction Monitor:

You can validate it from the Transaction Monitor. The Transaction Monitor is a tool for monitoring the status of inbound and outbound transactions originating from and going into Oracle E-Business Suite that have been processed by the XML Gateway and delivered or received by the Oracle Transport Agent. It shows a complete history and audit trail of these documents.

1. Log on to Oracle E-Business Suite with the Workflow Administrator Web Applications responsibility.

Select the Transaction Monitor link to open the search window to search for the order.

### Searching from the Transaction Monitor

The screenshot shows the Oracle Transaction Monitor Search Criteria form. The form is titled "Transaction Monitor:Search" and has a "Go" button at the bottom right. The form is divided into two main sections: "Search Criteria" and "Transaction Details".

**Search Criteria:**

- ☒ Inbound Messages
  - Processing Status: All
- ☐ Outbound Messages
  - Generation Status: All
  - Delivery Status: All
  - Retry Status: All

**Transaction Details:**

- Transaction Type: [Text Field]
- Source TP Location Code: [Text Field]
- Document ID: order\_xml\_01
- Party Type: Customer
- From Date: [Text Field]
- To Date: [Text Field]
- Transaction Subtype: [Text Field]
- Trading Partner Name: [Text Field]
- Site Name: [Text Field]

The form also includes a "Go" button at the bottom right.

2. Clear From Date and To Date fields and enter 'order\_xml\_01' in the Document ID field.
3. Select Customer as the Party Type. Click **Go** to execute the search.  
This retrieves XML inbound transaction 'order\_xml\_01' in the Inbound Search Results region.
4. Confirm that the transaction 'order\_xml\_01' has status 'SUCCESS'.

### To validate it in Oracle Order Management:

1. Log on to the Forms-based Oracle E-Business Suite with the Order Management, Super User responsibility.
2. Select **Order Returns > Sales Order**. Sales Order Forms would open up.
3. Search for an order by entering the order number in the Customer PO field (such as 'order\_xml\_01'). This would bring up the details of newly created order.

### Sales Orders

You can also select the Items tab for item details.

## Using XML Gateway Outbound Through Subscription Model

This section includes the following topics:

- Using XML Gateway Outbound Messages in Creating a BPEL Process at Design Time, page 4-30
- Deploying and Testing a BPEL Process at Run Time, page 4-51

## Using XML Gateway Outbound Services at Design Time

For an outbound XML Gateway Map interface, since an outbound message is first enqueued to the ECX\_OUTBOUND queue, Oracle E-Business Suite Integrated SOA Gateway supports it through subscription model by first dequeuing the message to

retrieve outbound data and then invoking an appropriate outbound XML Gateway map to update Oracle E-Business Suite.

### **BPEL Process Scenario**

Take XML Gateway outbound interface 'PO acknowledgement XML Transaction' as an example. The XML Gateway outbound interface is exposed as a Web service through ECX\_CBODO\_OAG62\_OUT outbound map.

When a purchase order is created and approved, on approval of the purchase order, a workflow will be triggered which creates the Purchase Order Acknowledgement flow and sends out the PO Acknowledgement as an XML file. The workflow delivers the Confirm BOD as the PO Acknowledgement to ECX\_OUTBOUND queue for delivery to the other system.

The correlation Id for this message is set to "BPEL" and the Oracle BPEL PM listens to ECX\_OUTBOUND queue for the message with the correlation Id = "BPEL". Confirm BOD as the PO Acknowledgement is written as an output XML file using File Adapter.

If the BPEL process is successfully executed after deployment, you should get the same order book reference ID (Customer PO) information from the output XML file once a purchase order is approved.

### **Prerequisites to Create a BPEL Process Using XML Gateway Outbound Messaging**

You need to set up the correlation identifier in Oracle E-Business Suite. 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 E-Business Suite 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. Search for COMM\_METHOD in the **Type** field to see if it exists in the system.
5. Add a new record to the COMM\_METHOD type by entering BPEL for the **Code** field and **Meaning** field. Enter description information and save the record.

Oracle XML Gateway puts the correlation of BPEL when enqueueing the message on the ECX\_OUTBOUND queue.

Once you have the correlation identifier set up correctly, you also need to ensure the trading partner that you want to use has the Protocol Type field set to BPEL.

### **BPEL Process Creation Flow**

Based on the PO acknowledgement XML Transaction scenario, the following

design-time tasks are discussed in this chapter:

1. Create a new BPEL project, page 4-32  
Use this step to create a new BPEL project called `XMLGOutbound.bpel`.
2. Create a Partner Link for AQ Adapter, page 4-33  
Use this step to dequeue the event details from the `ECX_OUTBOUND` queue.
3. Add a Receive activity, page 4-39  
Use the Receive activity to take PO acknowledgement details as an input to the Assign activity.
4. Add a Partner Link for File Adapter, page 4-41  
This is to write PO acknowledgement details in an XML file as an output file.
5. Add an Invoke activity, page 4-47  
This is to write PO acknowledgement information to an XML file through invoking the partner link for File Adapter.
6. Add an Assign activity, page 4-49  
Use the Assign activity to take the output from the Receive activity and to provide input to the Invoke activity.

For general information and basic concept of a BPEL process, see *Understanding BPEL Business Processes*, page D-1 and *Oracle BPEL Process Manager Developer's Guide* for details.

## Creating a New BPEL Project

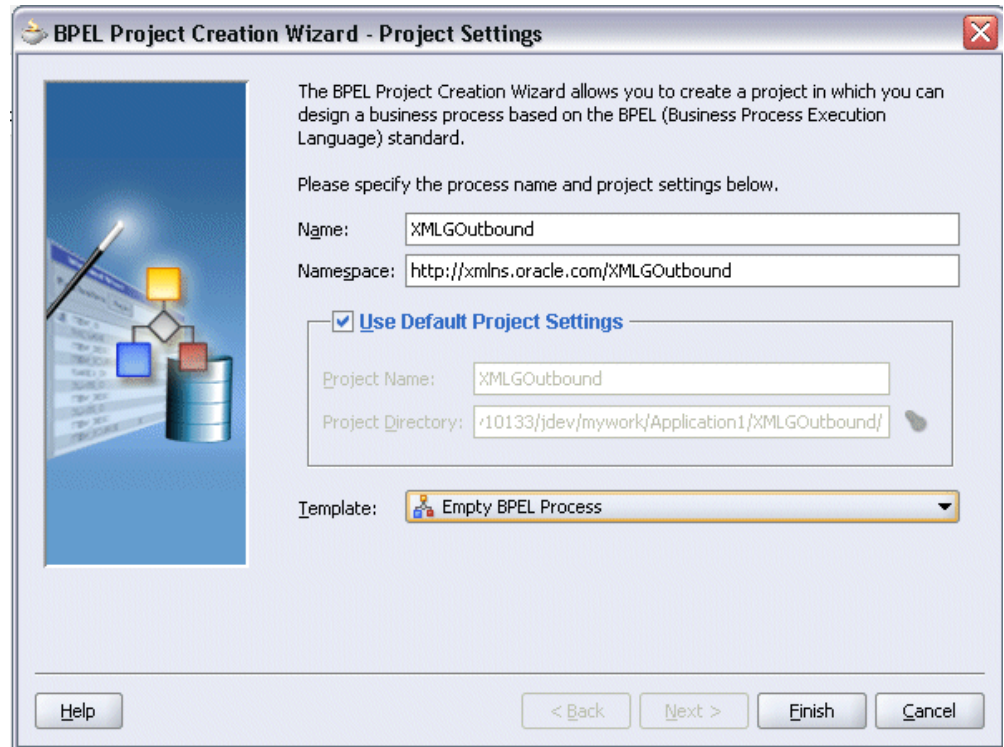
Use this step to create a new BPEL project that will contain various BPEL process activities.

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

7. In the **Name** field, enter a descriptive name such as XMLGOutbound.
8. From the Template list, select **Empty BPEL Process** and then select **Use Default Project Settings**.
9. Click **Finish**.

#### Creating a New BPEL Project



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

#### Creating a Partner Link for AQ Adapter

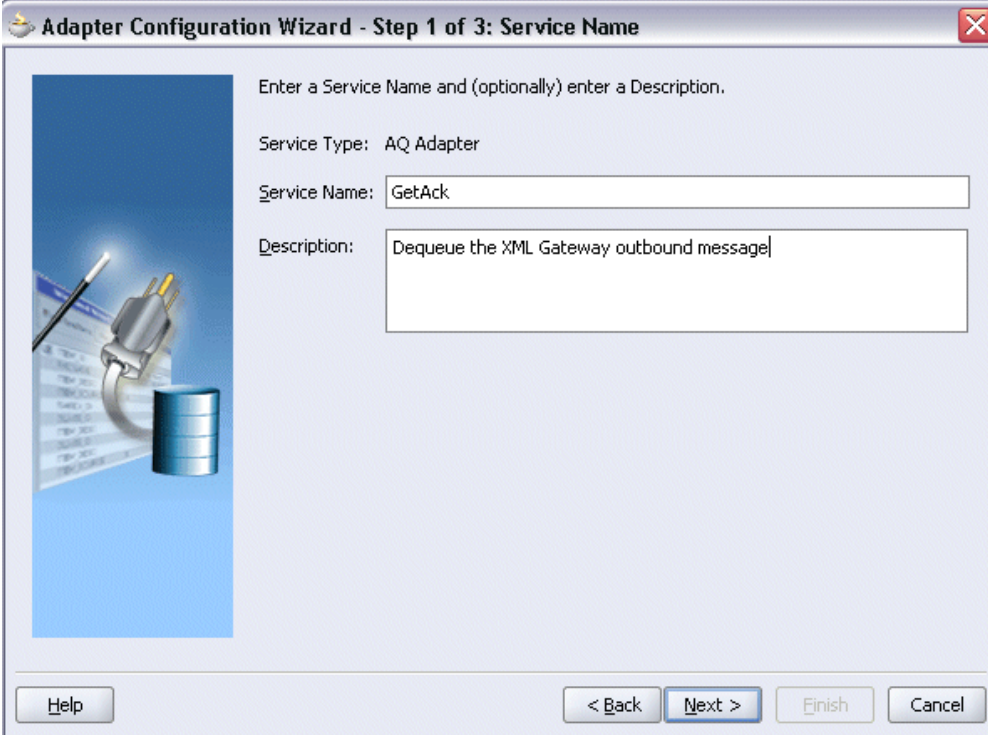
Use this step to create a Partner Link called `GetAck` for AQ Adapter to dequeue the XML Gateway outbound message (for example, `ECX_CBODO_OAG62_OUT`) in the `ECX_OUTBOUND` queue.

##### To create a partner link for AQ Adapter:

1. In JDeveloper BPEL Designer, drag and drop the **AQ Adapter** service from the Component Palette into the Partner Link border area of the process diagram. The Adapter Configuration Wizard appears.
2. Enter a service name in the Service Name dialog box, for example `GetAck`. You can

also add an optional description of the service.

### Entering Service Name



The screenshot shows a Windows-style dialog box titled "Adapter Configuration Wizard - Step 1 of 3: Service Name". On the left is a graphic of a USB cable and a database cylinder. The main area contains the text "Enter a Service Name and (optionally) enter a Description." Below this, "Service Type: AQ Adapter" is displayed. There are two input fields: "Service Name:" with the text "GetAck" and "Description:" with the text "Dequeue the XML Gateway outbound message". At the bottom are buttons for "Help", "< Back", "Next >", "Finish", and "Cancel".

3. Click **Next**. The Service Connection dialog box appears.
4. You can use an existing database connection by selecting a database connection from the **Connection** list or define a new database connection by clicking **New** to open the Create Database Connection Wizard.

**Note:** You need to connect to the database where Oracle E-Business Suite is running.

### To create a new database connection:

1. Click **New** to open the Create Database Connection Wizard. Click **Next** and enter an unique connection name and then select a connection type, such as Oracle (JDBC), for the database connection. Click **Next**.
2. Enter an appropriate username and password to authenticate the database connection in the Authentication dialog box. Click **Next**
3. Specify the following information in the Connection dialog box:



- Driver: Thin
- Host Name: Enter the host name for the database connection. For example, `myhost01.example.com`.
- JDBC Port: Enter JDBC port number (such as 1521) for the database connection.
- SID: Specify an unique SID value (such as `sid01`) for the database connection.

4. Click **Next** to test your database connection.

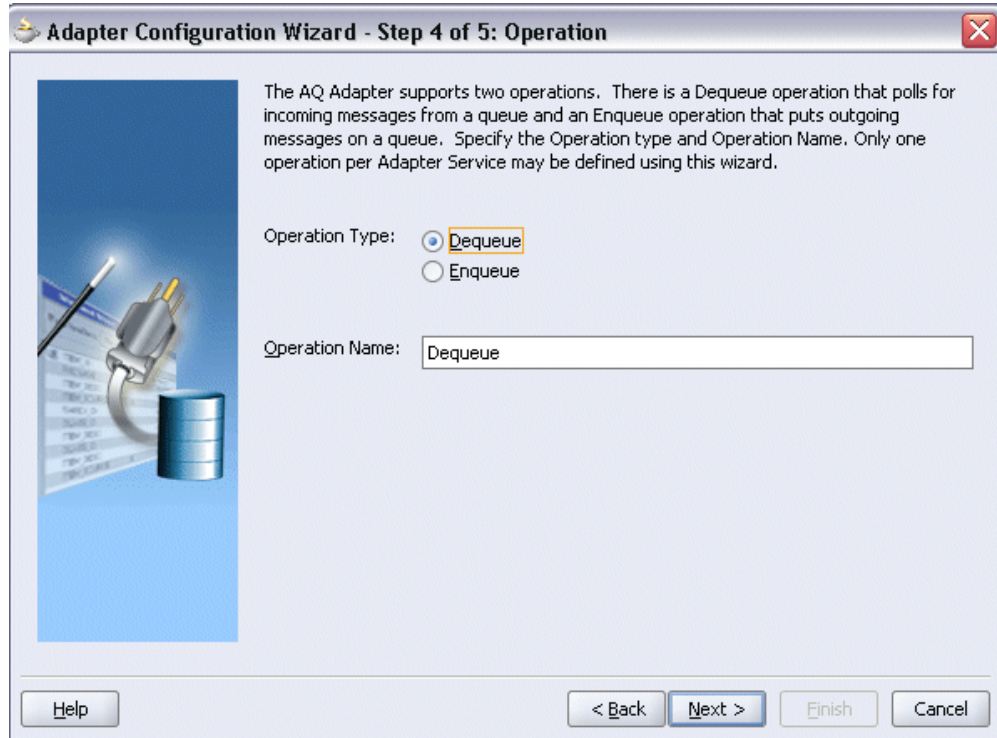
The status message "Success!" indicates a valid connection.

5. Click **Next** to return to the Service Connection dialog box providing a summary of the database connection.

5. 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.

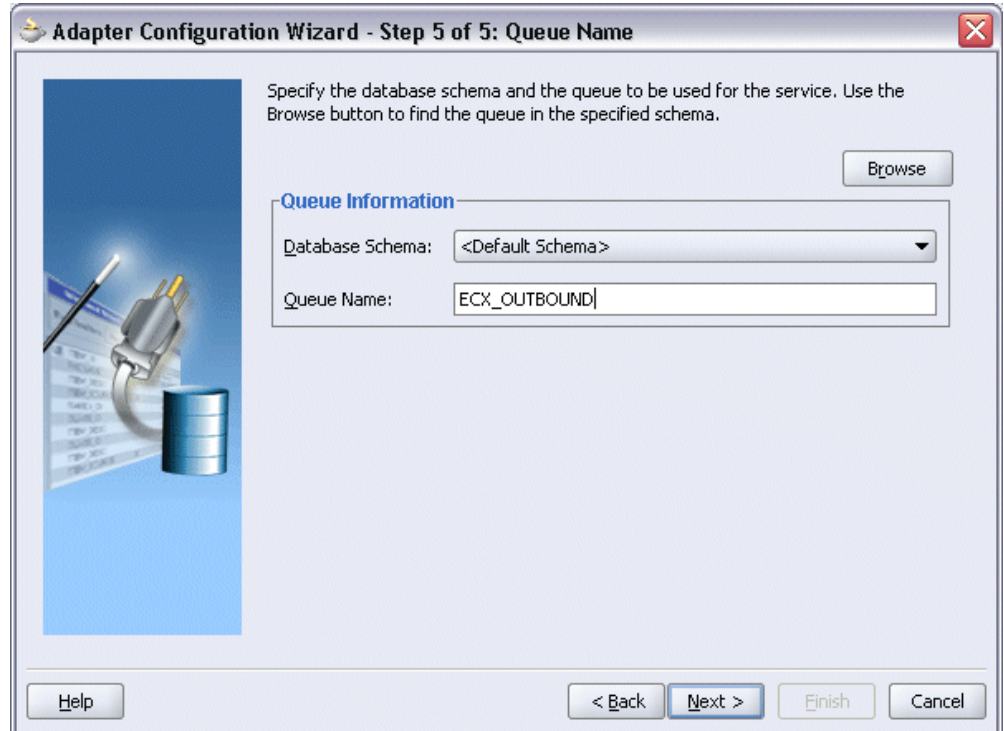
6. Click **Next** to open Operation dialog box.

Select **Dequeue** radio button and this selected value is also populated in the Operation Name field.

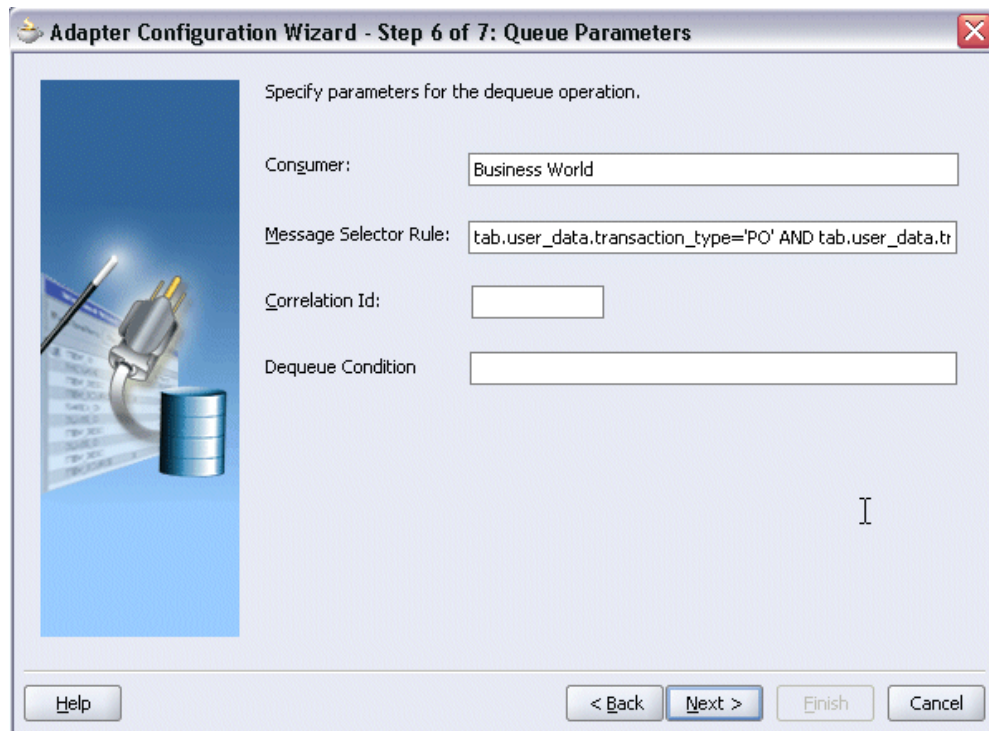


7. Click **Next** to open the Queue Name dialog box.

Select Default Schema as the Database Schema field. Enter 'ECX\_OUTBOUND' as the Queue Name field.



8. Click **Next** to open the Queue Parameter dialog box.



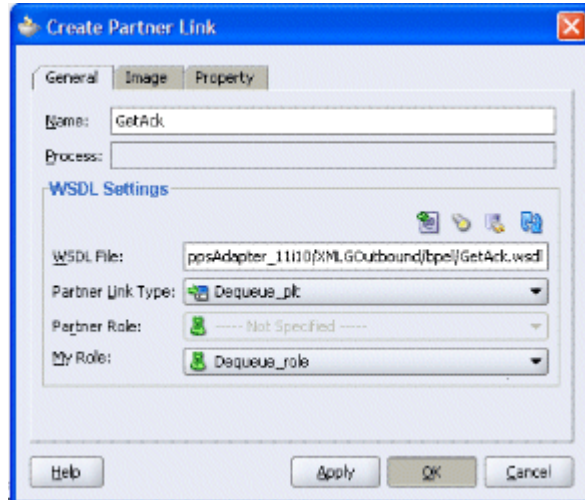
Enter the following information:

- Enter an unique consumer name.
- Enter message selector rule information (such as `tab.user_data.transaction_type='PO' AND tab.user_data.transaction_subtype='POO'`).

9. Click **Next**. The Messages dialog box opens.

Click **Browse** to open the Type Chooser window to select `CONFIRM_BOD_002.xsd` as the Schema Location and `CONFIRM_BOD_002` as the Schema Element.

10. Click **Next** to proceed to the Finish dialog box to confirm that you have finished defining the AQ Adapter for the `GetAck` service.
11. Click **Finish**. The wizard generates the WSDL file corresponding to the `GetAck` service.



Click **Apply** and then **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.

### Adding a Receive Activity

This step is to configure a Receive activity to receive XML data from the partner link `GetAck` that you configured for the AQ adapter service.

The XML data received from the Receive activity is used as an input variable to the Assign activity that will be created in the next step.

#### To add a Receive activity:

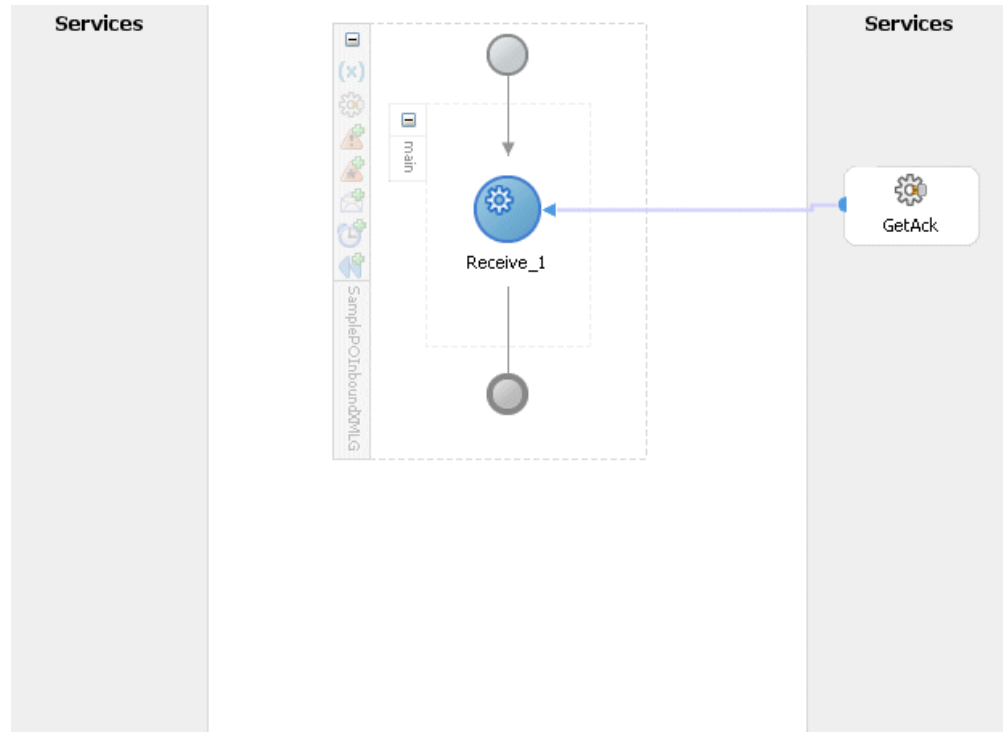
1. In JDeveloper BPEL Designer, drag and drop the **Receive** activity from the **BPEL Activities** section of the Component Palette into the Activity box of the process diagram.
2. Link the Receive activity to the `GetAck` partner link. The Receive activity will take event data from the partner link. The Edit Receive dialog box appears.
3. Enter a name for the receive activity. Click the **Create** icon next to the **Variable** field to create a new variable. The Create Variable dialog box appears.
4. Select **Global Variable**, and then enter a name for the variable. You can accept the default name. Click **OK** to return to the Edit Receive dialog box.
5. Select **Create Instance** check box. Click **Apply** and **OK** to finish configuring the Receive activity.

### Editing the Receive Activity

The screenshot shows the 'Edit Receive' dialog box. At the top, there is a title bar with a close button (X) and a warning icon with the text 'Errors: 1'. Below the title bar, there are five tabs: 'General', 'Correlations', 'Sensors', 'Adapters', and 'Annotations'. The 'General' tab is selected. Inside the 'General' tab, there is a 'Name' field with the value 'Receive'. Below this, there is a section titled 'My Role WebService Interface'. Inside this section, there is a 'Partner Link' field with the value 'GetAck' and a 'Operation' dropdown menu with the value 'Dequeue'. Below the 'Operation' dropdown, there is a 'Variable' field with the value 'Receive\_Dequeue\_InputVariable' and a 'Create Instance' checkbox which is checked. At the bottom of the dialog, there are four buttons: 'Help', 'Apply', 'OK', and 'Cancel'.

The Receive activity appears in the BPEL process diagram.

### Adding a Receive Activity



### Adding a Partner Link for File Adapter

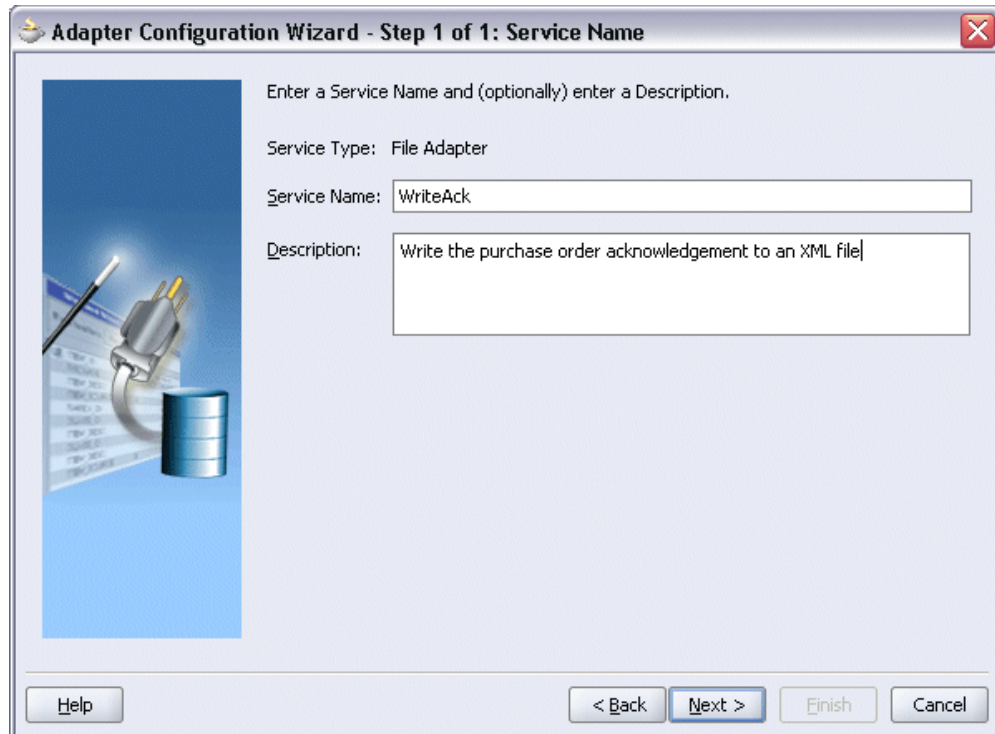
Use this step to configure a partner link by writing the purchase order acknowledgement to an XML file.

#### To add a Partner Link for 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. The Service Name dialog box appears.



### Specifying the Service Name



The screenshot shows a Windows-style dialog box titled "Adapter Configuration Wizard - Step 1 of 1: Service Name". On the left is a vertical blue bar with a graphic of a cable and a database cylinder. The main area contains the text "Enter a Service Name and (optionally) enter a Description." Below this, "Service Type: File Adapter" is displayed. There are two input fields: "Service Name:" with the text "WriteAck" and "Description:" with the text "Write the purchase order acknowledgement to an XML file". At the bottom are buttons for "Help", "< Back", "Next >" (which is highlighted with a blue border), "Finish", and "Cancel".

3. Enter a name for the File Adapter service, such as `WriteAck`. You can add an optional description of the service.
4. Click **Next** and the Operation dialog box appears.



### Specifying the Operation



The File Adapter supports three operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, and a Synchronous Read File operation that reads the current contents of a file. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type:

- ☐ Read File
- ☒ Write File
- ☐ Synchronous Read File

Operation Name:

Buttons: Help, < Back, Next >, Finish, Cancel

5. Specify the operation type, for example **Write File**. This automatically populates the **Operation Name** field.

Click **Next** to access the File Configuration dialog box.

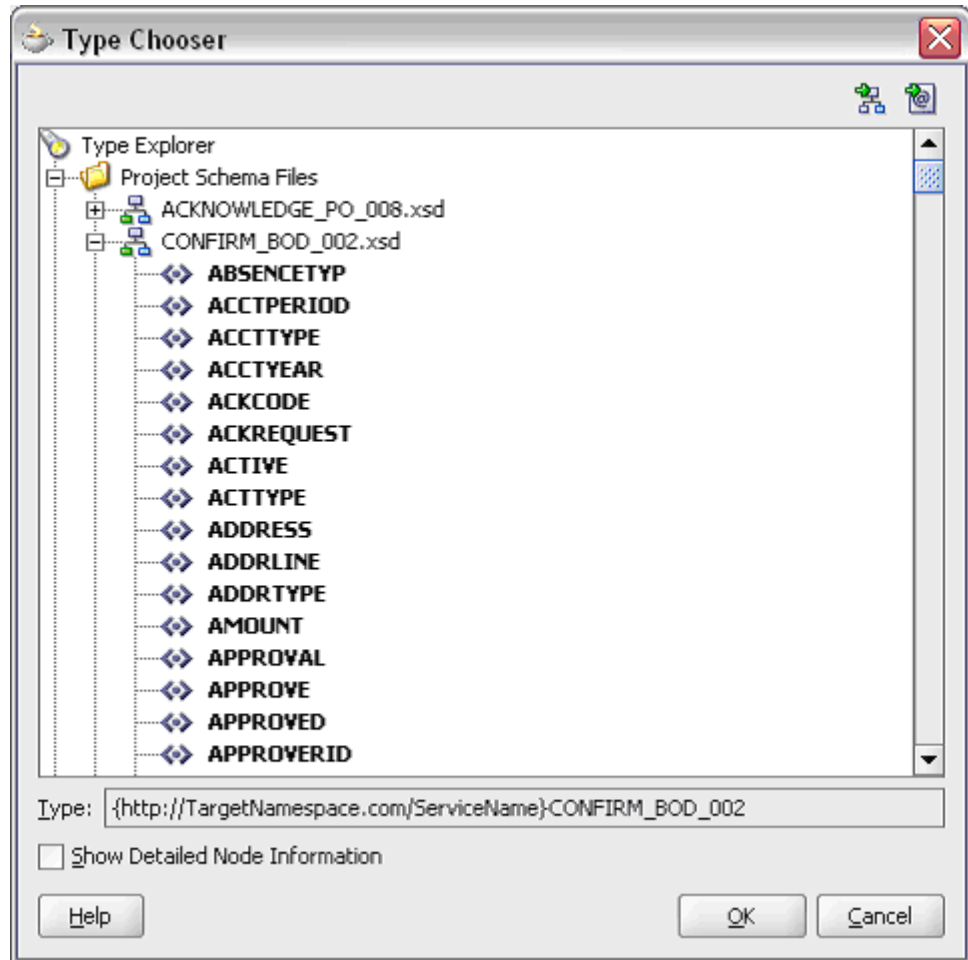
### Configuring the Output File

The screenshot shows the 'Adapter Configuration Wizard - Step 3 of 4: File Configuration' dialog box. On the left is a vertical panel with a blue background and a graphic of a USB cable and a hard drive. The main area has a light gray background and contains the following fields and controls:

- Text: 'Specify the parameters for the Write File operation.'
- Radio buttons: 'Directory specified as' with options 'Physical Path' (unselected) and 'Logical Name' (selected).
- Text field: 'Directory for Outgoing Files (logical name):' with the value 'outputDir'.
- Text field: 'File Naming Convention (po\_%SEQ%.txt):' with the value 'POAck%yyMMddHHmmss%.xml'.
- Section header: 'Write to new file when existing file meets any of these conditions'.
- Checkboxes and spinners:
  - ☒ Number of Messages Equals: 1 (spinner)
  - ☐ Elapsed Time Exceeds: 1 (spinner) minutes (dropdown)
  - ☐ File Size Exceeds: 1000 (spinner) kilobytes (dropdown)
- Buttons at the bottom: 'Help', '< Back', 'Next >', 'Finish', and 'Cancel'.

6. For the Directory specified as field, select **Logical Path**. Enter directory path in the Directory for Outgoing Files field, and specify a naming convention for the output file such as POAck%yyMMddJJmmss%.xml.
7. Confirm the default write condition: Number of Messages Equals 1. Click **Next**. The Messages dialog box appears.
8. Select **Browse** check box to locate the schema location and schema element.  
The Type Chooser dialog box appears. Expand the **Project Schema Files > CONFIRM\_BOD\_002.xsd** and select **CONFIRM\_BOD\_002**.

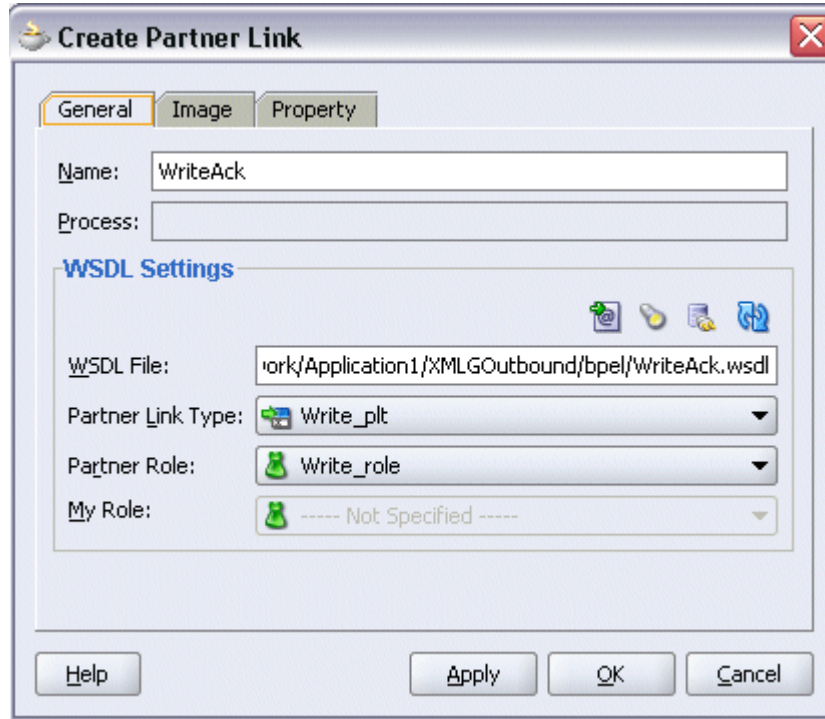
### Type Chooser



Click **OK** to populate the selected schema location and element.

9. Click **Next** and 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 `WriteAck.wsdl`.

### Completing the Partner Link Configuration



The image shows a 'Create Partner Link' dialog box with three tabs: 'General', 'Image', and 'Property'. The 'General' tab is active. It contains the following fields and controls:

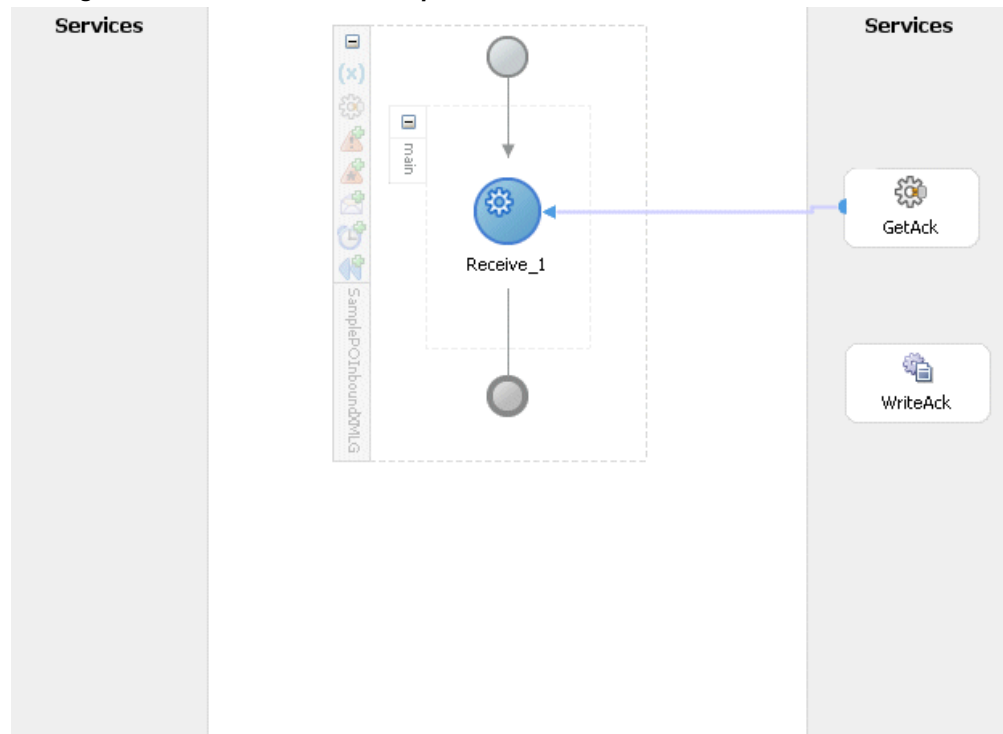
- Name:** A text field containing 'WriteAck'.
- Process:** An empty text field.
- WSDL Settings:** A section with a title bar and four icons (a green plus, a lightbulb, a document, and a blue circular arrow).
  - WSDL File:** A text field containing 'ork/Application1/XMLGOutbound/bpel/WriteAck.wsdl'.
  - Partner Link Type:** A dropdown menu showing 'Write\_plt' with a small icon to the left.
  - Partner Role:** A dropdown menu showing 'Write\_role' with a small icon to the left.
  - My Role:** A dropdown menu showing '---- Not Specified ----' with a small icon to the left.

At the bottom of the dialog are four buttons: 'Help', 'Apply', 'OK', and 'Cancel'.

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

The `WriteAck` Partner Link appears in the BPEL process diagram.

### Adding the Partner Link for File Adapter



### Adding an Invoke Activity

This step is to configure an Invoke activity to send the purchase order acknowledgement that is received from the Receive activity to the WriteAck partner link in an XML file.

#### To add an 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, after the **Receive** activity.
2. Link the Invoke activity to the WriteAck service. The Invoke activity will send event data to the partner link. The Edit Invoke dialog box appears.

### Editing the Invoke Activity

**Invoke**

Errors: 1

General Correlations Sensors Adapters Annotations

Name: Invoke

**Partner Role Web Service Interface**

Partner Link: WriteAck

Operation: Write

Input Variable: Invoke\_Write\_InputVariable

Output Variable:

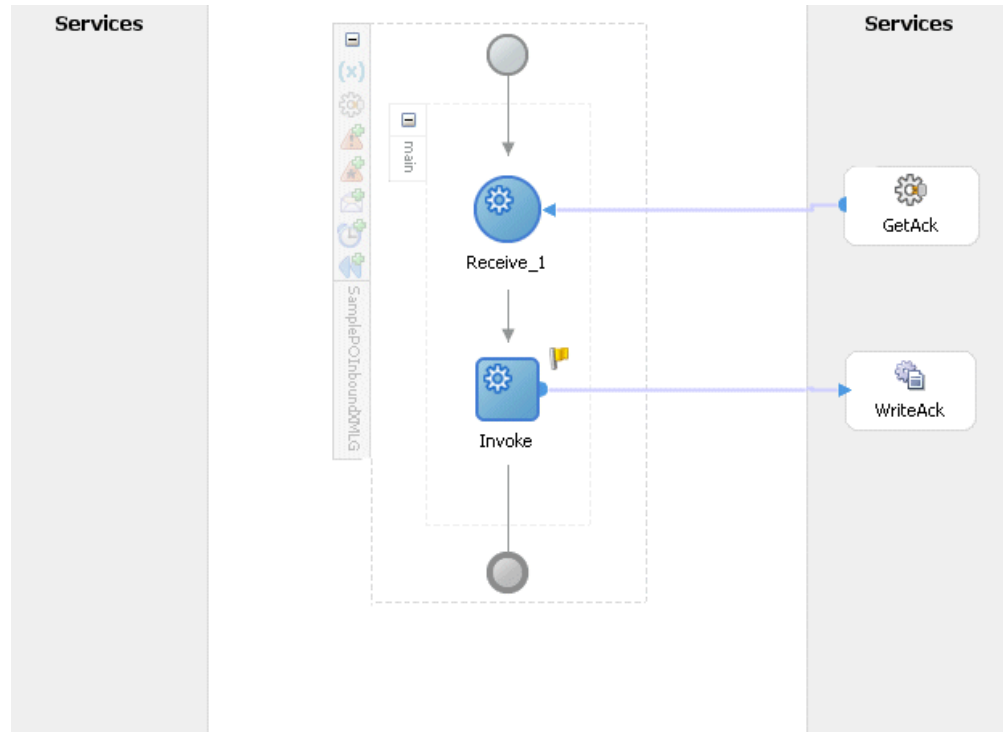
Help Apply OK Cancel

3. Enter a name for the Invoke activity, and then click the **Create** icon next to the **Input Variable** field to create a new variable. The Create Variable dialog box appears.
4. Select **Global Variable**, and then enter a name for the variable. You can also accept the default name. Click **OK**.

Click **Apply** and then **OK** in the Edit Invoke dialog box to finish configuring the Invoke activity.

The Invoke activity appears in the process diagram.

### Adding an Invoke Activity



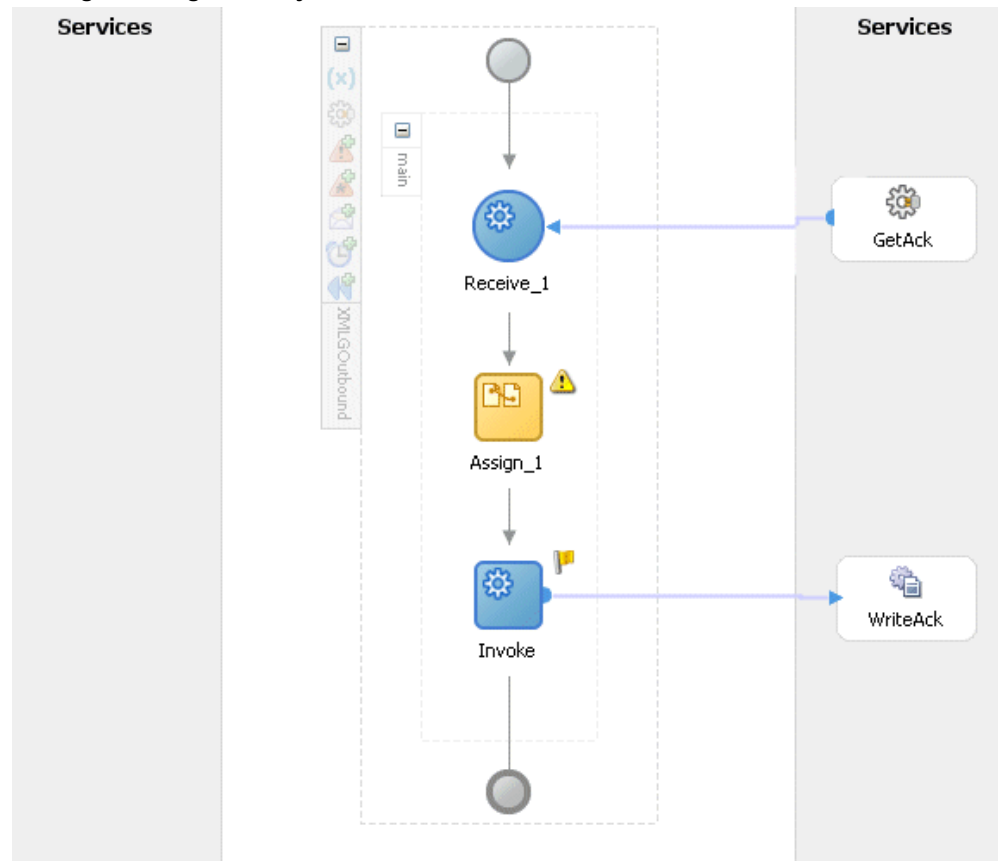
### Adding an Assign Activity

Use this step to pass the purchase order acknowledgement details from the Receive activity to the Invoke activity.

#### To add an 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.

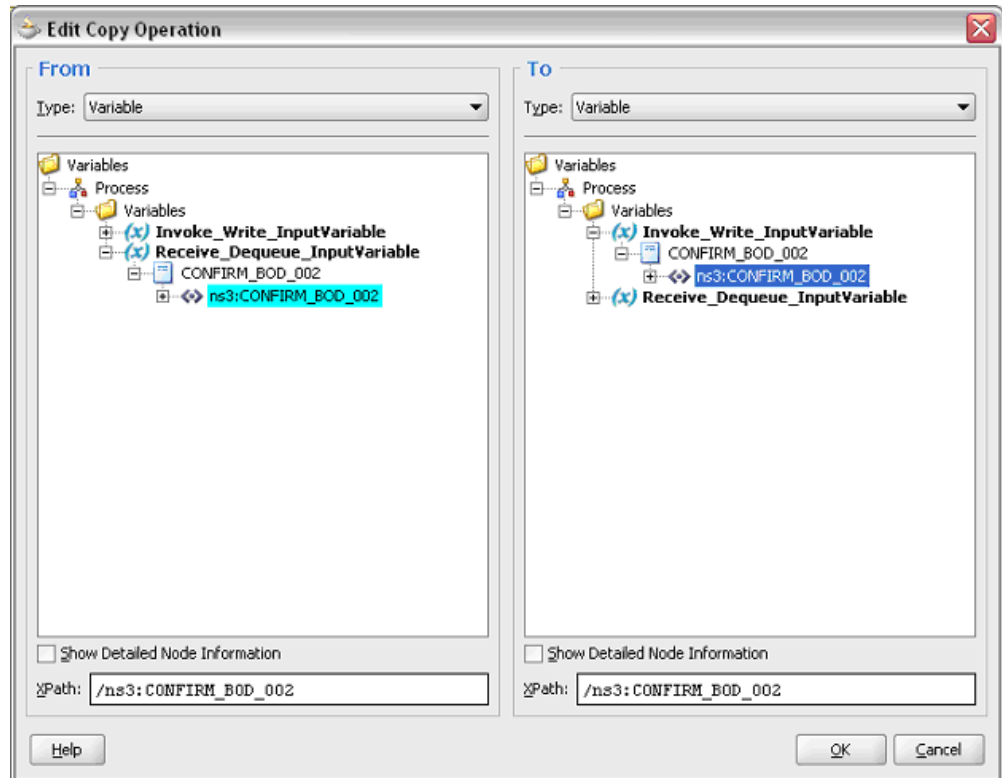
### Adding an Assign Activity



2. Double-click the **Assign** activity to access the Edit Assign dialog box.
3. On the Copy Operation tab, click **Create** and then select **Copy Operation** from the menu. The Create Copy Operation window appears.
4. In the From navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Receive\_DEQUEUE\_InputVariable** and select **CONFIRM\_BOD\_002**. The XPath field should contain your selected entry.
5. In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Invoke\_Write\_InputVariable** and select **CONFIRM\_BOD\_002**. The XPath field should contain your selected entry.



### Specifying Assign Parameters



6. Click **OK**.

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

## Deploying and Testing the BPEL Process at Run Time

After creating the BPEL process, you can deploy it to a BPEL server. To ensure that this process is modified or orchestrated appropriately, you can also test the BPEL process by initiating the business process contained in the BPEL process to test the interface integration.

### Prerequisites

Before deploying the BPEL process using Oracle JDeveloper, you must ensure that you have established the connectivity between the design-time environment and the run-time servers including the application server and the integration server.

How to configure the necessary server connection, see *Configuring Server Connection*, page B-1.

To validate the BPEL process, perform the following run-time tasks:

1. Deploy the BPEL process, page 4-52

Once you deploy the process to a BPEL server, it becomes available so that you can run the process manually to test it for validation.

2. Manually test the BPEL process, page 4-53

After deploying a BPEL process, you can manage the process from the BPEL console to manually initiate the business process and test the interface integration contained in your BPEL process.

## Deploying the BPEL Process

Before manually test the BPEL process, you first need to deploy it to the BPEL server.

### To deploy the BPEL process:

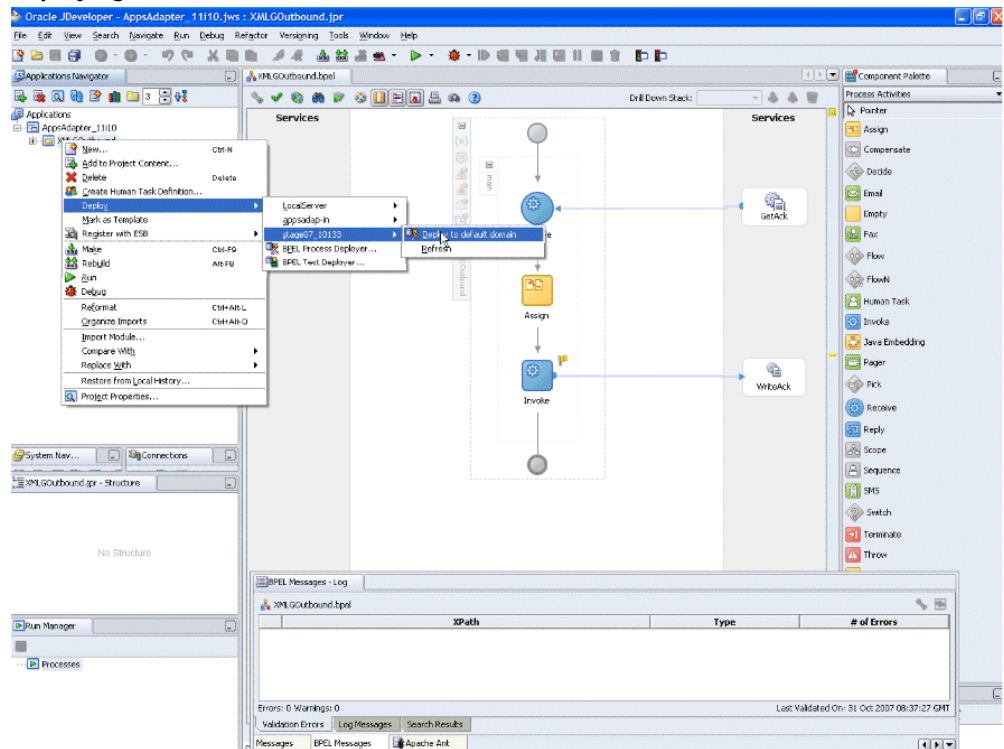
1. In the Applications Navigator of JDeveloper BPEL Designer, select the **XMLGOutbound** project.
2. Right-click the project and click **Make** action from the menu.

Look for any compilation error messages in Message Log.

Right-click the project and select **Deploy >Integration Server Connection name > Deploy to Default Domain** action from the menu.

For example, you can select **Deploy > BPELServerConn > Deploy to Default Domain** to deploy the process if you have the BPEL Process Manager setup appropriately.

### Deploying the BPEL Process



3. Look for 'Build successful' message in Apache Ant – Log to ensure that the BPEL project is compiled and successfully deployed.

### Testing the BPEL Process

To validate whether the BPEL process that you created works or not, you need to manually initiate the process after it has been successfully deployed to the BPEL console. You can log on to Oracle E-Business Suite to manually create and book the order as well as generate the order acknowledgement by submitting a Workflow Background Process concurrent request.

Login to the BPEL console to validate the BPEL process which writes purchase order acknowledgement in an output directory after receiving from the XML Gateway ECX\_OUTBOUND queue.

#### To manually test the BPEL process:

1. Log on to Oracle E-Business Suite with the XML Gateway responsibility.  
This is to ensure that the XML Gateway trading partner is set up correctly so that a purchase order can have a valid customer that has been defined.
2. Select Define Trading Partner from the navigation menu to access the Trading

Partner Setup window.

3. Enter the header values on the Trading Partner Setup form as follows:
  - Trading Partner Type: Customer
  - Trading Partner Name: For example, Business World
  - Trading Partner Site: Enter a trading partner site information. For example, 2391 L Street, San Jose, CA 95106
  - Company Admin Email: Enter a valid email address.
4. Enter the following trading partner details:
  - Transaction Type: ECX
  - Transaction SubType: CBODO
  - Standard Code: OAG
  - External Transaction Type: BOD
  - External Transaction SubType: CONFIRM
  - Direction: Out
  - Map: ECX\_CBODO\_OAG62\_OUT
  - Connection / Hub: DIRECT
  - Protocol Type: BPEL
  - Username: 'operation'
  - Password: enter 'welcome' twice
  - Protocol Address: 'http:ebssoa.sample.com'
  - Source Trading Partner Location Code: BWSANJOSE

### Trading Partner Setup

Transaction Type	Transaction SubType	Standard Code	External Transaction Type	External Transaction SubType	Direction	Map	Connection/Hub	Protocol Type
AR	CONFIRM_E	OAG	BOD	CONFIRM	IN	002_confirm_t		
ECX	CBODO	OAG	BOD	CONFIRM	IN	ECX_CBODI_C		
ECX	CBODO	OAG	BOD	CONFIRM	OUT	ECX_CBODO	DIRECT	BPEL
AR	PROCESS	OAG	INVOICE	PROCESS	OUT	171_process	DIRECT	HTTP
ONT	POA	OAG	PO	ACKNOWLED	OUT	ONT_3A4A_O	DIRECT	HTTP
ONT	POI	OAG	PO	PROCESS	IN	ONT_3A4R_O		
OZF	POSI	OAG	POS	PROCESS	IN	OZF_PROCES		
OZF	POSI	OAG	POS	PROCESS	IN	OZF_PROCES		
OZF	POSQ	OAG	POS	PROCESS	OUT	OZF_PROCES	DIRECT	HTTP

5. Save the trading partner details. Switch responsibility back to Order Management Super User, Vision Operations (USA) and select Customer > Standard from the navigation menu to open the Enter Customer form.
6. Search on the 'Business World' in the Name field and click **Find**.
7. Select the Business World with the following information from the search results.
  - Party Number: 2813
  - Customer Number: 1608
  - Account Name: Business World
  - Identifying check box: checked
  - Address: 2391 L Street, San Jose, CA 95106
8. Select and open this customer information. Enter 'BWSANJOSE' in the EDI Location field.
9. In Business Purposes tab, create a new row with the following values:

- Usage: Sold To
- Check on 'Primary' Check box

Save your work.

10. Use the following steps to generate acknowledgement for already created order.
  1. Select Order Returns > Sales Order to open the Sales Order form.
  2. Retrieve the order that you have created earlier by entering the order ID in the Customer PO field.
  3. Click **Book Order** to book the order.

### Booking an Order

The screenshot shows the 'Sales Orders (56707) - Business World' window. The 'Order Information' tab is active, and the 'Main' sub-tab is selected. The form displays the following details:

Field	Value
Customer	Business World
Customer Number	1600
Customer PO	order_xm1g_000
Customer Contact	
Ship To Location	San Jose (OPS)
	2391 L Street
	San Jose, CA, 95106, US
Bill To Location	San Jose (OPS)
	2391 L Street
	San Jose, CA, 95106, US
Order Number	56707
Order Type	Mixed
Date Ordered	19 JUL 2006 05:08:48
Price List	Corporate
Salesperson	Sprague, Mr. Howard

A 'Note' dialog box is open, displaying the message 'Order has been booked.' and a list of items with their prices:

	2,399.00
	215.91
	71.97
	2,686.88

The 'Book Order' button is visible at the bottom right of the window.

11. Switch to the System Administrator responsibility and select Request > Run.
12. Select **Single Request** and click **OK**.
13. Enter the following information in the Submit Request form:

### Specifying Parameters

The screenshot shows two overlapping windows. The background window is titled 'Submit Request' and contains the following fields: 'Name' with the value 'Workflow Background Process', 'Parameters' (empty), and 'Language' with the value 'American English'. Below these are sections for 'At these Times...' (Run the Job: As Scheduled), 'Upon Completion...' (Save: checked, Layout: empty, Notify: empty, Print to: noprint), and buttons for 'Help (C)', 'Submit', and 'Cancel'. The foreground window is titled 'Parameters' and contains: 'Item Type' with the value 'OM Send Acknowledgment', 'Minimum Threshold' (empty), 'Maximum Threshold' (empty), 'Process Deferred' with a 'Yes' button, 'Process Timeout' with a 'No' button, and 'Process Stuck' with a 'No' button. It also has 'OK', 'Cancel', 'Clear', and 'Help' buttons at the bottom.

- Name: Workflow Background Process
- Enter the following parameters:
  - Item Type: OM Send Acknowledgement
  - Process Deferred: Y
  - Process Timeout: N
  - Process Stuck: N
- Click **OK**.

14. Click **Submit** to submit the 'send acknowledgement' request.

15. View your request by entering the request ID to ensure its status is 'Success'.

### Validating Using Oracle Transaction Monitor

To validate it using Oracle Transaction Monitor, you need to log on to Oracle E-Business Suite with the Workflow Administrator Web Applications responsibility. Select Transaction Monitor to open the search window to search for the order.

## Searching from the Transaction Monitor

The screenshot shows the Oracle Transaction Monitor Search page. At the top, there is a blue header with the Oracle logo and the text 'Transaction Monitor'. To the right of the header are links for 'Diagnostics', 'Home', 'Logout', 'Preferences', and 'Help'. Below the header, there is a warning message: 'Warning: Low-level logging is currently enabled. Your application will not perform as well while low-level logging is on.' The main section is titled 'Transaction Monitor Search' and contains a 'Search Criteria' form. The form has two radio buttons: 'Inbound Messages' (selected) and 'Outbound Messages'. Under 'Inbound Messages', there is a 'Processing Status' dropdown menu set to 'All'. Under 'Outbound Messages', there are three dropdown menus: 'Generation Status' (set to 'All'), 'Delivery Status' (set to 'All'), and 'Retry Status' (set to 'All'). Below these are several text input fields: 'Transaction Type', 'Transaction Subtype', 'Source TP Location Code', 'Trading Partner Name', 'Document ID', and 'Site Name'. There are also date and time pickers for 'From Date' (set to '01-Nov-2007 00:00:0') and 'To Date' (set to '01-Nov-2007 23:59:5'). A 'Go' button is located at the bottom right of the form. At the bottom of the page, there is a blue footer with links for 'About this Page' and 'Privacy Statement', and a copyright notice: 'Copyright (c) 2006, Oracle. All rights reserved.'

## Validating Using Oracle BPEL Console

Log into Oracle BPEL Console to confirm that the `XMLGOutbound` process has been deployed. This process is continuously polling the `ECX_OUTBOUND` queue for purchase order acknowledgement.

To verify, select the instance of your deployed process which opens up in the Instances tab of your selected BPEL process.

Click on the Audit Tab to view the Receive activity. Click the **view xml document** link to open the received XML file. Note the Reference ID such as Customer PO.



## Viewing XML File for the Receive Activity

The screenshot shows the Oracle Enterprise Manager 10g BPEL Console interface. The main window displays the audit trail for a BPEL process instance titled "XMLGatewayOutbound". The audit trail shows the following activities:

- Receive:** Received "Receive\_Dequeue\_InputVa" at [2006/07/19 17:53:18].
- Assign:** Updated variable "Invoke\_Write\_Input" at [2006/07/19 17:53:18].
- Invoke:** Invoked 1-way operation "Write" on p at [2006/07/19 17:53:18].

The audit trail also shows the completion of the BPEL process instance "44" at [2006/07/19 17:53:18].

A pop-up window titled "http://152.69.246.100:8080/BPELConsole/default/xmlGetAu..." displays the XML content of the received message. The XML is as follows:

```
<LOGICALID />
<COMPONENT>BPEL</COMPONENT>
<TASK>POISSUE</TASK>

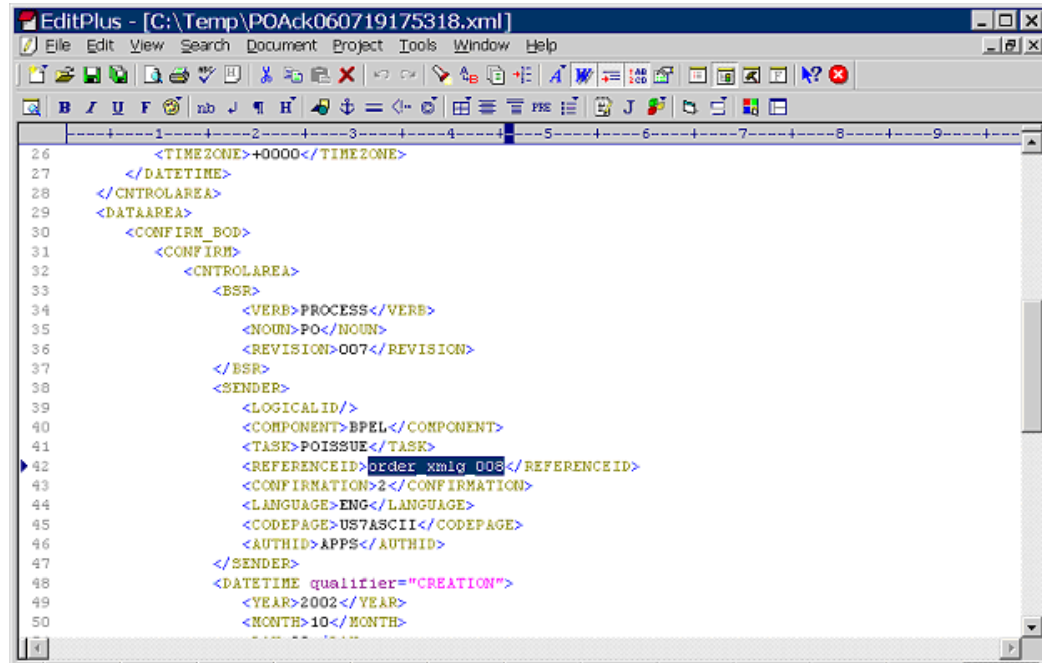
<REFERENCEID>order_xmlg_008</REFERENCEID>

<CONFIRMATION>2</CONFIRMATION>
<LANGUAGE>ENG</LANGUAGE>
<CODEPAGE>US7ASCII</CODEPAGE>
<AUTHID>APPS</AUTHID>
</SENDER>
- <DATETIME qualifier="CREATION">
  <YEAR>2002</YEAR>
  <MONTH>10</MONTH>
```

Go to the directory you specified for the write operation, for example `outputDir` - logical location (typically `c:\temp`) where the File Adapter has placed the file after writing the PO Acknowledgement in an XML file (such as 'POAck060719175318.xml').

Open this 'POAck060719175318.xml' file. You should find the Reference ID as `order_xmlg_008` (the order booked) for which the acknowledgement is generated.

### Validating the Output File



```
26      <TIMEZONE>+0000</TIMEZONE>
27    </DATETIME>
28  </CNTROLAREA>
29  <DATAAREA>
30    <CONFIRM_BOD>
31      <CONFIRM>
32        <CNTROLAREA>
33          <BSR>
34            <VERB>PROCESS</VERB>
35            <NOUN>PO</NOUN>
36            <REVISION>007</REVISION>
37          </BSR>
38          <SENDER>
39            <LOGICALID/>
40            <COMPONENT>BP&#226;L</COMPONENT>
41            <TASK>POISSUE</TASK>
42            <REFERENCEID>order xmig 008</REFERENCEID>
43            <CONFIRMATION>2</CONFIRMATION>
44            <LANGUAGE>ENG</LANGUAGE>
45            <CODEPAGE>US7ASCII</CODEPAGE>
46            <AUTHID>APPS</AUTHID>
47          </SENDER>
48          <DATETIME qualifier="CREATION">
49            <YEAR>2002</YEAR>
50            <MONTH>10</MONTH>
```

---

# Using Business Events Through Subscription Model

## Overview

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 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.

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.

Oracle E-Business Suite Integrated SOA Gateway supports business events through event subscription. An integration repository administrator can subscribe to a business event from the business event interface details page. The subscription to that event can be enqueued as an out agent. An integration developer can create a BPEL process in Oracle JDeveloper to include the subscribed event at design time and update application data if needed at run time.

To better understand how the subscription model works for business events, detailed tasks at design time and run time are included in this chapter. For the example described in the following sections, we use Oracle JDeveloper 10.1.3.3.0 as a design-time tool to create the BPEL process and use Oracle SOA Suite BPEL server 10.1.3.3.0 for the process deployment.

## Using a Business Event in Creating a BPEL Process at Design Time

### BPEL Process Scenario

Take a PO XML Raise business event as an example to explain the BPEL process creation.

When a purchase order is created and approved, a Purchase Order Approved business event `oracle.apps.po.evnt.xmlpo` is raised. Since the subscription to this event is created through the interface details page (internally, an event subscription is automatically created for the selected event with `WF_BPEL_QAGENT` as Out Agent), and enqueued in `WF_EVENT_T` structure to Advanced Queue `WF_BPEL_Q`, we will create a BPEL process to first dequeue the subscription from the `WF_BPEL_Q` queue to get the event details. The event details will be passed through BPEL process activities and then will be written in XML file as an output file.

If the BPEL process is successfully executed after deployment, you should get the same purchase order information from the output file once a purchase order is approved.

### **Prerequisites to Create a BPEL Process Using a Business Event**

Integration repository administrators must first subscribe to a business event from the Oracle Integration Repository user interface. Internally, an event subscription is automatically created for that event with `WF_BPEL_QAGENT` as Out Agent.

For example, a business event `oracle.apps.po.event.xmlpo` needs to be subscribed. A confirmation message appears if the event subscription is successfully created.

## Subscribing to a Business Event

The screenshot displays the Oracle Integration Repository web interface. At the top, the header includes the Oracle logo and the text 'Integration Repository'. Below the header, there is a navigation bar with links for 'Integration Repository', 'Diagnostics', 'Home', 'Logout', 'Preferences', 'Help', and 'Personalize Page'. The main content area shows a 'Confirmation' message: 'Successfully created subscription for business event 'oracle.apps.po.event.xmlpo' with Out Agent as 'WF\_BPEL\_QAGENT''. Below this, the 'Business Event Details : Send PO via XML' are listed. The details include: Internal Name (oracle.apps.po.event.xmlpo), Type (Business Event), Product (Purchasing), Status (Active), Business Entity (Standard Purchase Order), Scope (Public), and Interface Source (Custom). There are buttons for 'Browse', 'Search', 'Printable Page', and 'Unsubscribe'. Below the details, the 'Full Description' section shows 'Send PO via XML'. The 'Source Information' section lists: Source File (patch/115/po/US/rpodemoUpdatede.wfx), Source Version (12.0), and Source Product (PO). At the bottom, there are again buttons for 'Browse', 'Search', and 'Printable Page'.

To subscribe to a business event, the administrators will first locate an event from the Oracle Integration Repository, and then click **Subscribe** in the interface detail page to create the subscription.

For information on how to subscribe to business events, see *Subscribing to Business Events, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

**Note:** If a BPEL process is created with the business event that you have subscribed to it, in order for the subscribed business event to be successfully enqueued to WF\_BPEL\_Q queue, you need to make sure:

- The consumer name must be unique.
- The BPEL process is deployed before raising the business event.

Once the subscription is created and enqueued, an integration developer can then orchestrate the subscribed event into a meaningful business process in Oracle JDeveloper using BPEL language at design time.

### BPEL Process Creation Flow

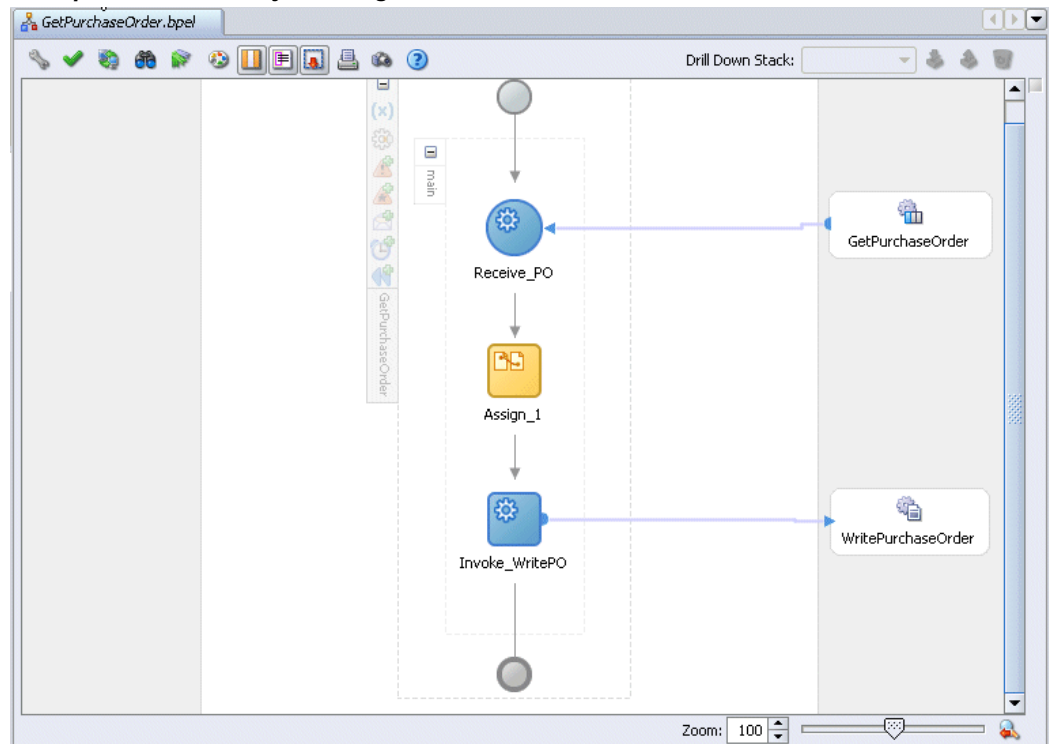
Based on the PO XML Raise business event scenario, the following design-time tasks are discussed in this chapter:

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

Use this step to create a new BPEL project called `GetPurchaseOrder.bpel`.

2. Create a Partner Link for AQ Adapter, page 5-6  
Use this step to dequeue the event details from the `WF_BPEL_Q` queue.
3. Add a Receive activity, page 5-13  
Use the Receive activity to take event details as an input to the Assign activity.
4. Create a Partner Link for File Adapter, page 5-15  
This is to write event details in an XML file as an output file.
5. Add an Invoke activity, page 5-21  
This is to write business event information to an XML file through invoking the partner link for File Adapter.
6. Add an Assign activity, page 5-23  
Use the Assign activity to take the output from the Receive activity and to provide input to the Invoke activity.

### Example of a BPEL Project Using Business Events



For general information and basic concept of a BPEL process, see Understanding BPEL Business Processes, page D-1 and *Oracle BPEL Process Manager Developer's Guide* for details.

## Creating a New BPEL Project

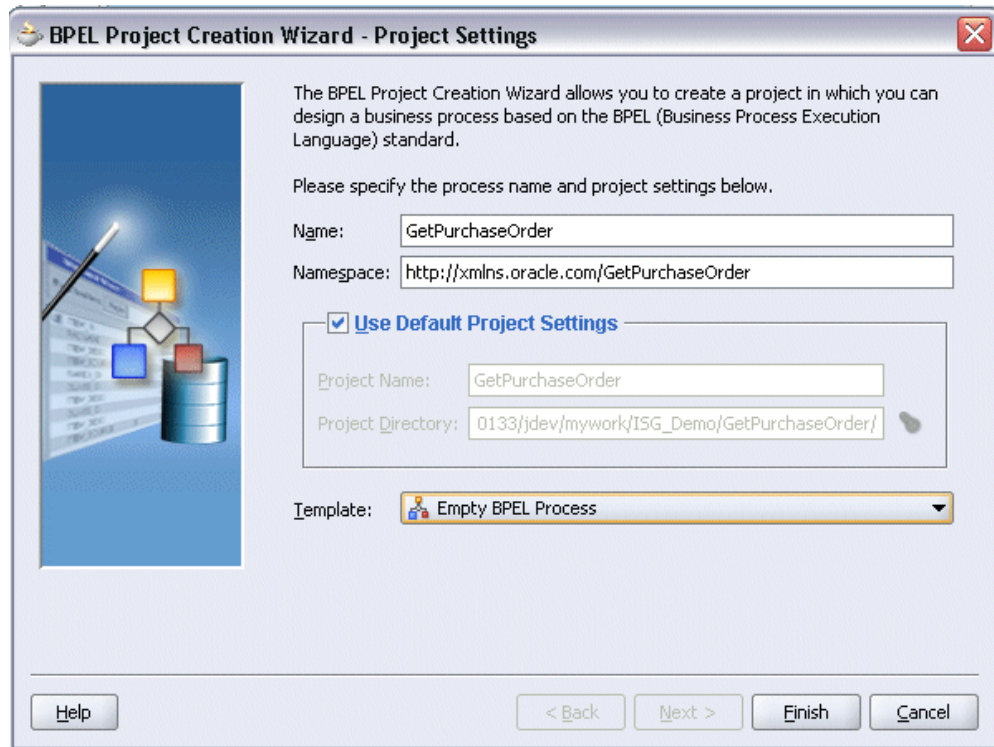
Use this step to create a new BPEL project that will contain various BPEL process activities.

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

7. In the **Name** field, enter a descriptive name such as `GetPurchaseOrder`.
8. From the Template list, select **Empty BPEL Process** and then select **Use Default Project Settings**.
9. Click **Finish**.

#### Creating a New Project



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

## Creating a Partner Link for AQ Adapter

Use this step to create a Partner Link called `GetPurchaseOrder` for AQ Adapter to dequeue the subscription to `oracle.apps.po.evnt.xmlpo` event.

**To create a partner link for AQ Adapter to dequeue the event subscription:**

1. In JDeveloper BPEL Designer, drag and drop the **AQ Adapter** service from the Component Palette into the Partner Link border area of the process diagram. The Adapter Configuration Wizard appears.
2. Enter a service name in the Service Name dialog box, for example `GetPurchaseOrder`. You can also add an optional description of the service.





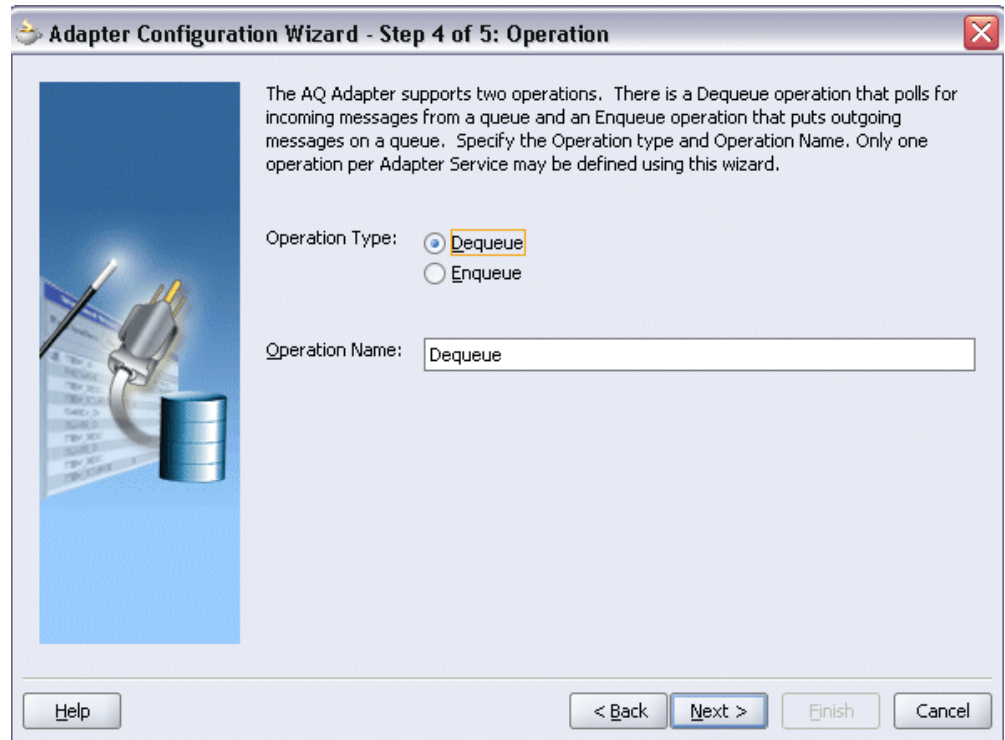
3. Click **Next**. The Service Connection dialog box appears.
4. You can use an existing database connection by selecting a database connection from the **Connection** list or define a new database connection by clicking **New** to open the Create Database Connection Wizard.

**Note:** You need to connect to the database where Oracle E-Business Suite is running.

**To create a new database connection:**

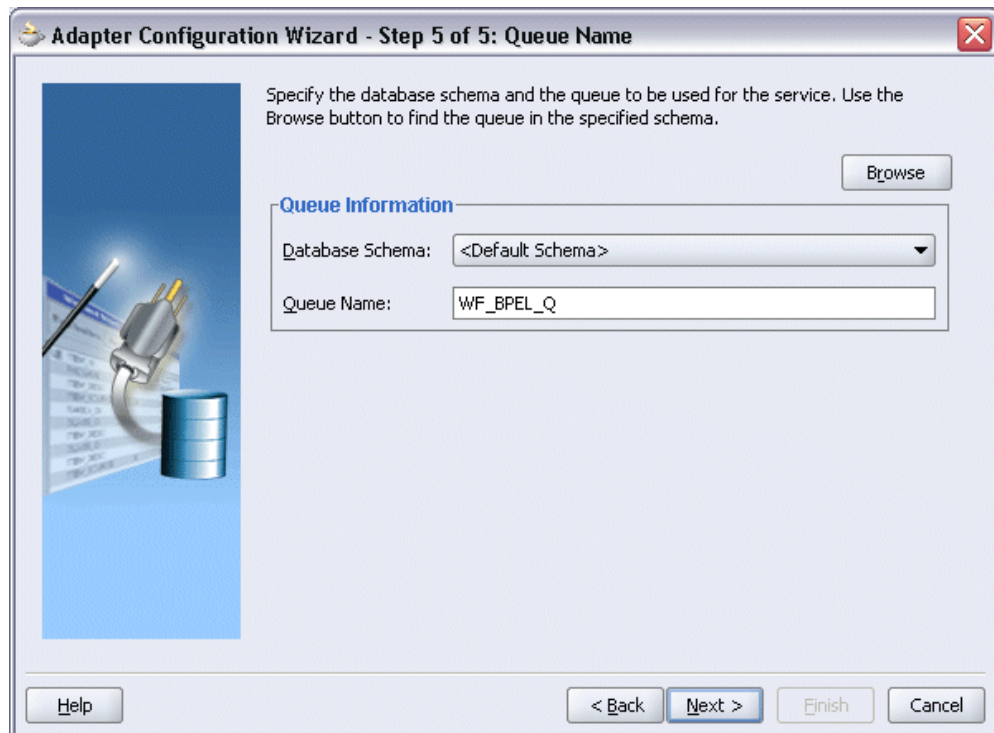
1. Click **New** to open the Create Database Connection Wizard. Click **Next** and enter an unique connection name and then select a connection type for the database connection. Click **Next**.
2. Enter an appropriate username and password to authenticate the database connection in the Authentication dialog box. Click **Next**
3. Specify the following information in the Connection dialog box:
  - Driver: Thin
  - Host Name: Enter the host name for the database connection. For example, myhost01.example.com.

- **JDBC Port:** Enter JDBC port number (such as 1521) for the database connection.
  - **SID:** Specify an unique SID value (such as sid01)for the database connection.
4. Click **Next** to test your database connection.  
The status message "Success!" indicates a valid connection.
  5. Click **Next** to return to the Service Connection dialog box providing a summary of the database connection.
  5. 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.
  6. Click **Next** to open Operation dialog box.  
Select **Dequeue** radio button and this selected value is also populated in the Operation Name field.

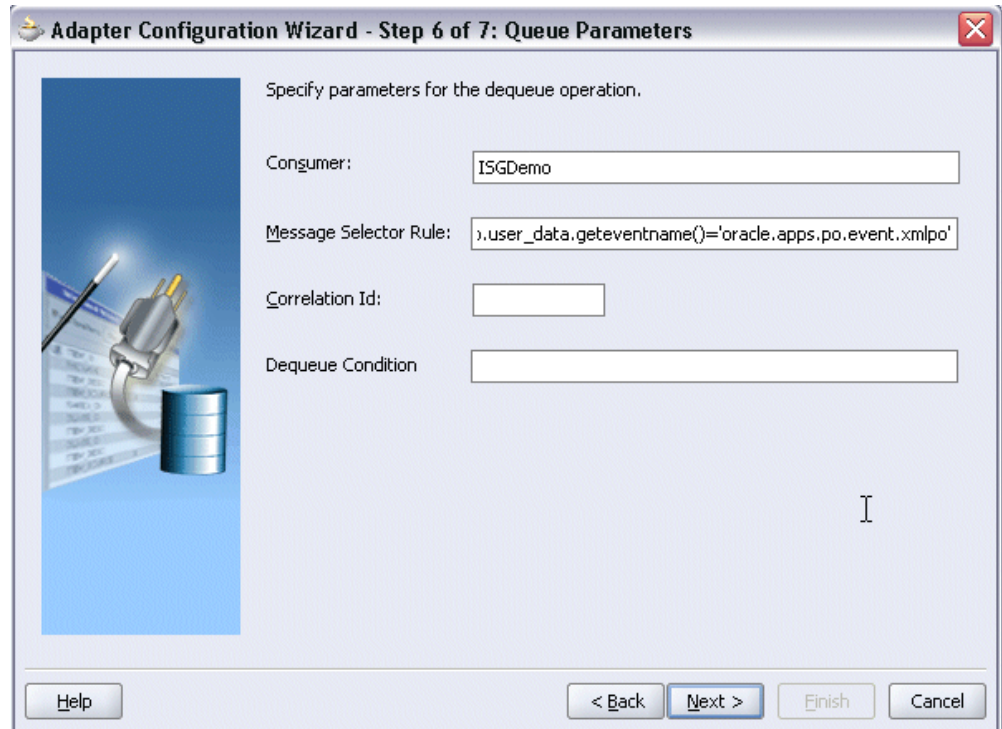


7. Click **Next** to open the Queue Name dialog box.

Select Default Schema as the Database Schema field. Enter 'WF\_BPEL\_Q' as the Queue Name field.



8. Click **Next** to open the Queue Parameter dialog box.



Enter the following information:

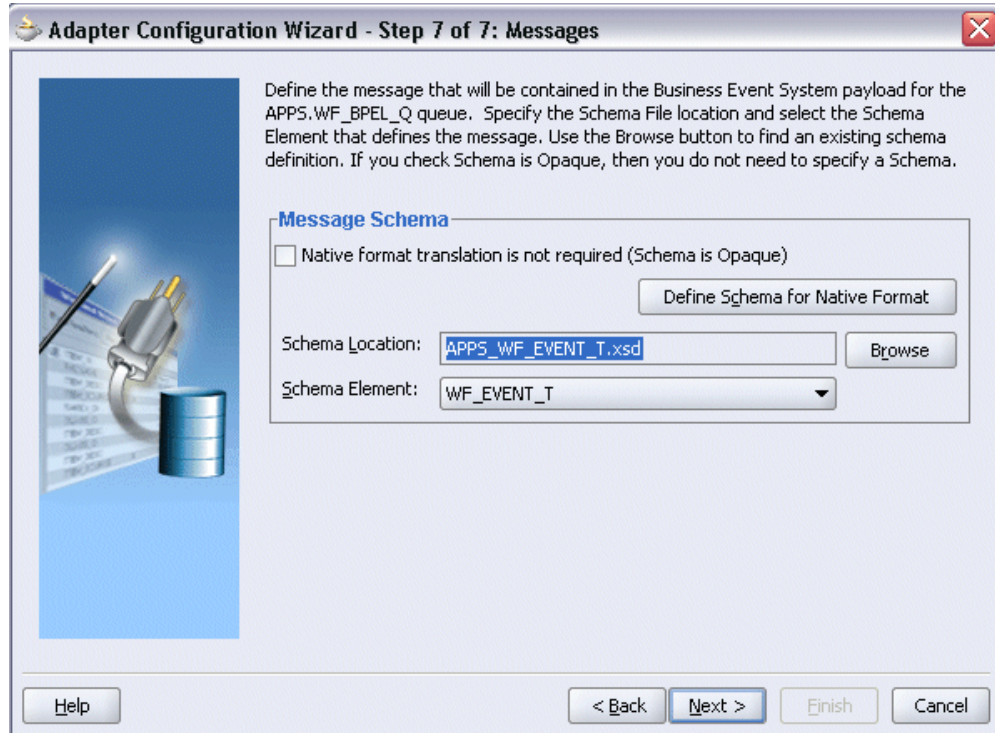
- Enter an unique consumer name.

**Important:** In order for the subscribed business event to be successfully enqueued to WF\_BPEL\_Q queue, the consumer name must be unique.

- Enter message selector rule information (such as `tab.user_data.geteventname()='oracle.apps.po.evnt.xmlpo'`).

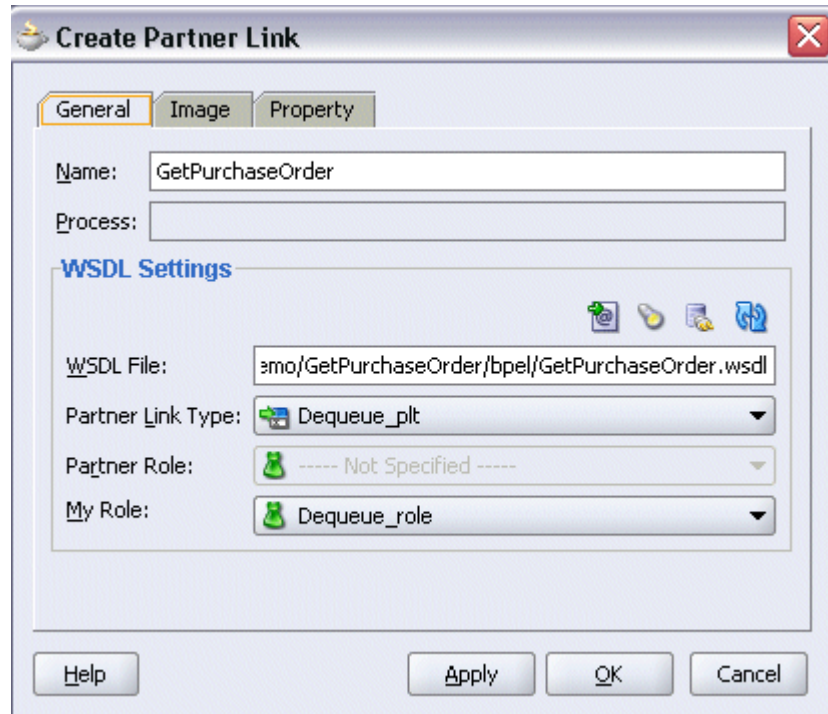
9. Click **Next**. The Messages dialog box opens where you can define the message that will be contained in the Business Event System payload for the APPS.WF\_BPEL\_Q queue.

Click **Browse** to open the Type Chooser window to select APPS\_WF\_EVENT\_T.xsd as the Schema Location and WF\_EVENT\_T as the Schema Element.



10. Click **Next** to proceed to the Finish dialog box to confirm that you have finished defining the AQ Adapter for the `GetPurchaseOrder` service.
11. Click **Finish**. The wizard generates the WSDL file corresponding to the `GetPurchaseOrder` service.





Click **Apply** and then **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.

## Adding a Receive Activity

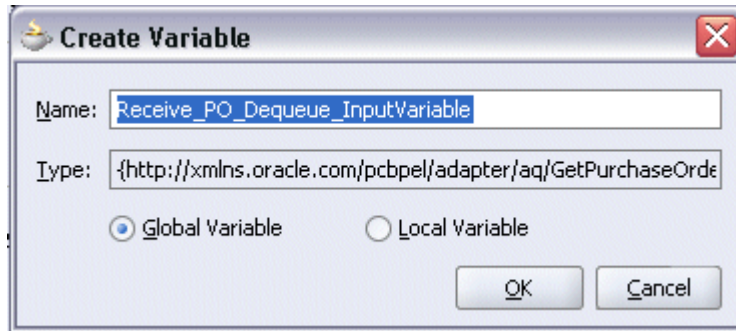
This step is to configure a Receive activity to receive XML data from the partner link `GetPurchaseOrder` that you configured for the AQ adapter service for the business event.

The XML data received from the Receive activity is used as an input variable to the Assign activity that will be created in the next step.

### To add a Receive activity to obtain Purchase Order XML data:

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.
2. Link the Receive activity to the `GetPurchaseOrder` partner link. The Receive activity will take event data from the partner link. The Edit Receive dialog box appears.
3. Enter a name for the receive activity such as 'Receive\_PO' and then click the **Create** icon next to the **Variable** field to create a new variable. The Create Variable dialog box appears.

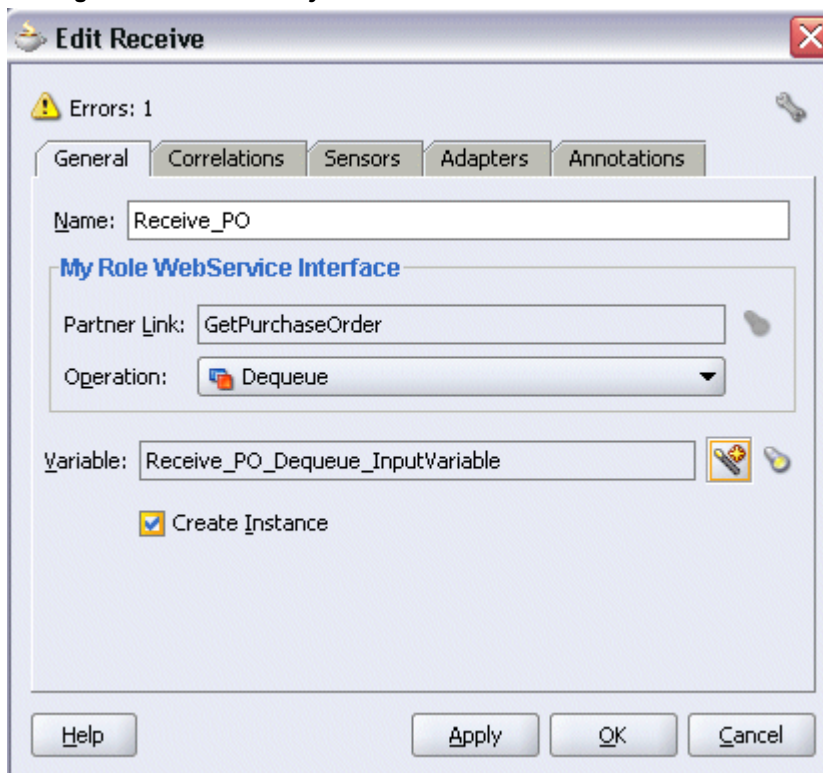
#### Creating a Variable



The 'Create Variable' dialog box is shown. It has a title bar with a close button. The 'Name' field contains 'Receive\_PO\_Dequeue\_InputVariable'. The 'Type' field contains '{http://xmlns.oracle.com/pcbpel/adapter/aq/GetPurchaseOrder}'. There are two radio buttons: 'Global Variable' (selected) and 'Local Variable'. At the bottom are 'OK' and 'Cancel' buttons.

4. Select **Global Variable** and then enter a name for the variable. You can accept the default name. Click **OK** to return to the Edit Receive dialog box.
5. Select **Create Instance** check box then click **Apply** and then **OK** to finish configuring the Receive activity.

#### Editing the Receive Activity

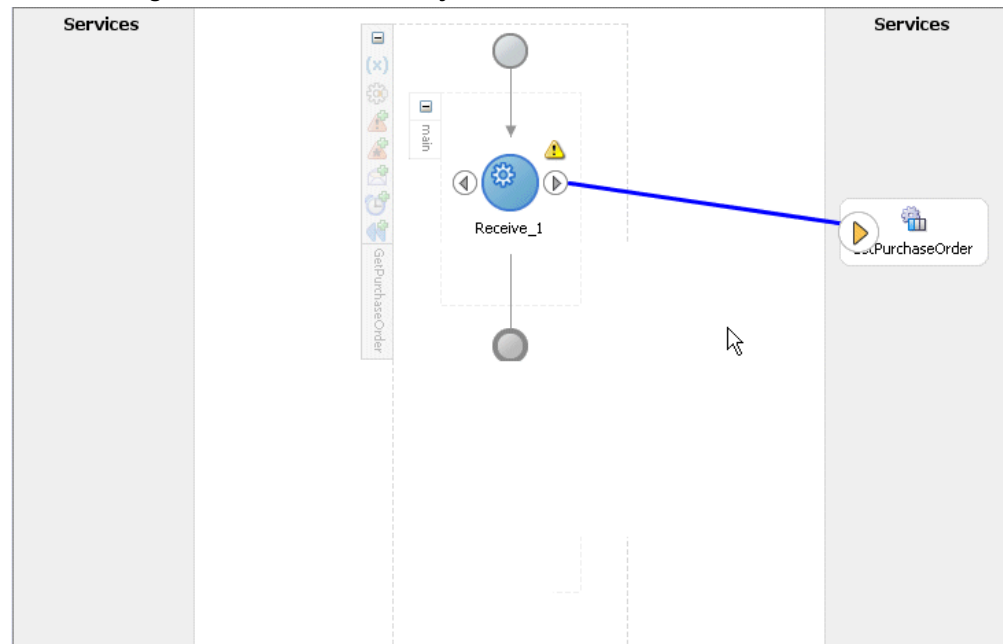


The 'Edit Receive' dialog box is shown. It has a title bar with a close button. There is a warning icon and 'Errors: 1' at the top left. Below are tabs: 'General' (selected), 'Correlations', 'Sensors', 'Adapters', and 'Annotations'. The 'Name' field contains 'Receive\_PO'. Below is a section titled 'My Role WebService Interface' with a 'Partner Link' field containing 'GetPurchaseOrder' and an 'Operation' dropdown menu showing 'Dequeue'. Below that is a 'Variable' field containing 'Receive\_PO\_Dequeue\_InputVariable' with a key icon. At the bottom is a checked checkbox 'Create Instance'. At the very bottom are 'Help', 'Apply', 'OK', and 'Cancel' buttons.

The Receive activity appears in the BPEL process diagram.



### Process Diagram with Receive Activity



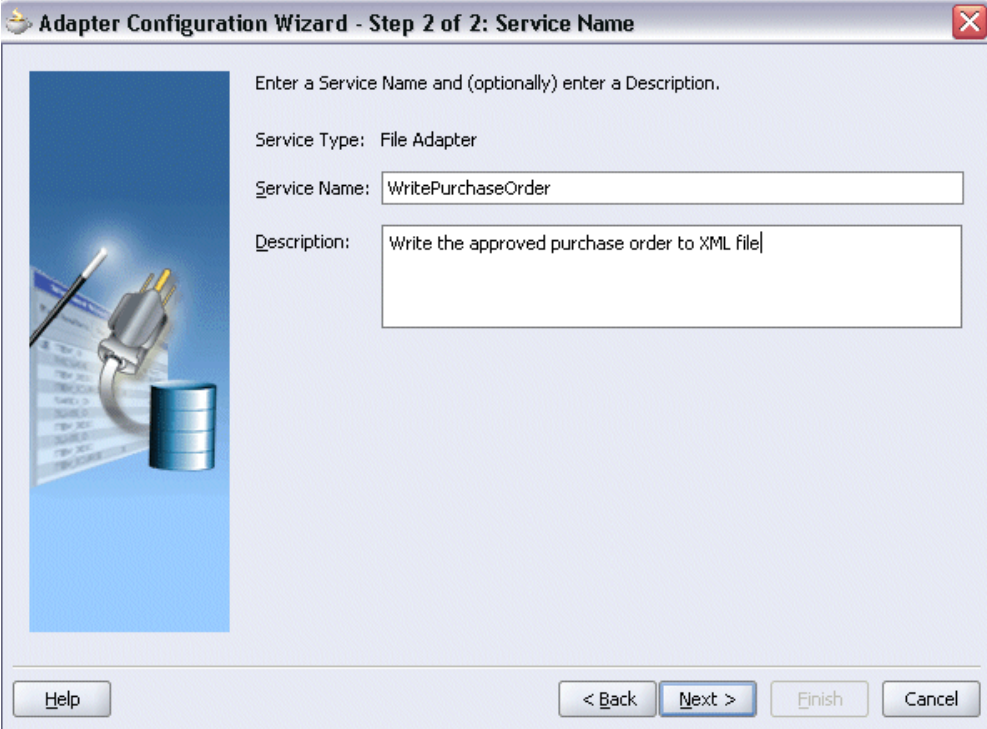
## Adding a Partner Link for File Adapter

Use this step to configure a business event by writing the event data to an XML file.

### To add a Partner Link for 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. The Service Name dialog box appears.

### Specifying the Service Name



The screenshot shows a Windows-style dialog box titled "Adapter Configuration Wizard - Step 2 of 2: Service Name". On the left is a vertical blue bar containing an illustration of a USB cable and a database cylinder. The main area has a light blue background. At the top, it says "Enter a Service Name and (optionally) enter a Description." Below this, "Service Type: File Adapter" is displayed. There are two input fields: "Service Name:" with the text "WritePurchaseOrder" and "Description:" with the text "Write the approved purchase order to XML file". At the bottom, there are four buttons: "Help", "< Back", "Next >" (which is highlighted with a blue border), "Finish", and "Cancel".

Adapter Configuration Wizard - Step 2 of 2: Service Name

Enter a Service Name and (optionally) enter a Description.

Service Type: File Adapter


Service Name: WritePurchaseOrder

Description: Write the approved purchase order to XML file

Help < Back Next > Finish Cancel

3. Enter a name for the file adapter service such as `WritePurchaseOrder`. You can add an optional description of the service.
4. Click **Next** and the Operation dialog box appears.

### Specifying the Operation



The File Adapter supports three operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, and a Synchronous Read File operation that reads the current contents of a file. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type:

- ☐ Read File
- ☒ Write File
- ☐ Synchronous Read File

Operation Name:

Help < Back Next > Finish Cancel

5. Specify the operation type, for example **Write File**. This automatically populates the **Operation Name** field.

Click **Next** to access the File Configuration dialog box.

### Configuring the Output File

The screenshot shows the 'Adapter Configuration Wizard - Step 4 of 5: File Configuration' dialog box. On the left is a vertical blue bar with an image of a USB cable and a hard drive. The main area has a light blue background with the text 'Specify the parameters for the Write File operation.' Below this, there are two radio buttons: 'Physical Path' (selected) and 'Logical Name'. A text field labeled 'Directory for Outgoing Files (physical path):' contains the text 'outputDir'. Below that, a text field labeled 'File Naming Convention (po\_%SEQ%.txt):' contains the text 'PO\_%SEQ%.xml'. A section titled 'Write to new file when existing file meets any of these conditions' contains three checkboxes: 'Number of Messages Equals:' (checked), 'Elapsed Time Exceeds:', and 'File Size Exceeds:'. The 'Number of Messages Equals:' checkbox has a value of '1' in a spinner box. The 'Elapsed Time Exceeds:' checkbox has a value of '1' in a spinner box and a 'minutes' dropdown menu. The 'File Size Exceeds:' checkbox has a value of '1000' in a spinner box and a 'kilobytes' dropdown menu. At the bottom are buttons for 'Help', '< Back', 'Next >', 'Finish', and 'Cancel'.

Specify the parameters for the Write File operation.

Directory specified as ☒ Physical Path ☐ Logical Name

Directory for Outgoing Files (physical path):  
outputDir

File Naming Convention (po\_%SEQ%.txt): PO\_%SEQ%.xml

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

☒ Number of Messages Equals: 1

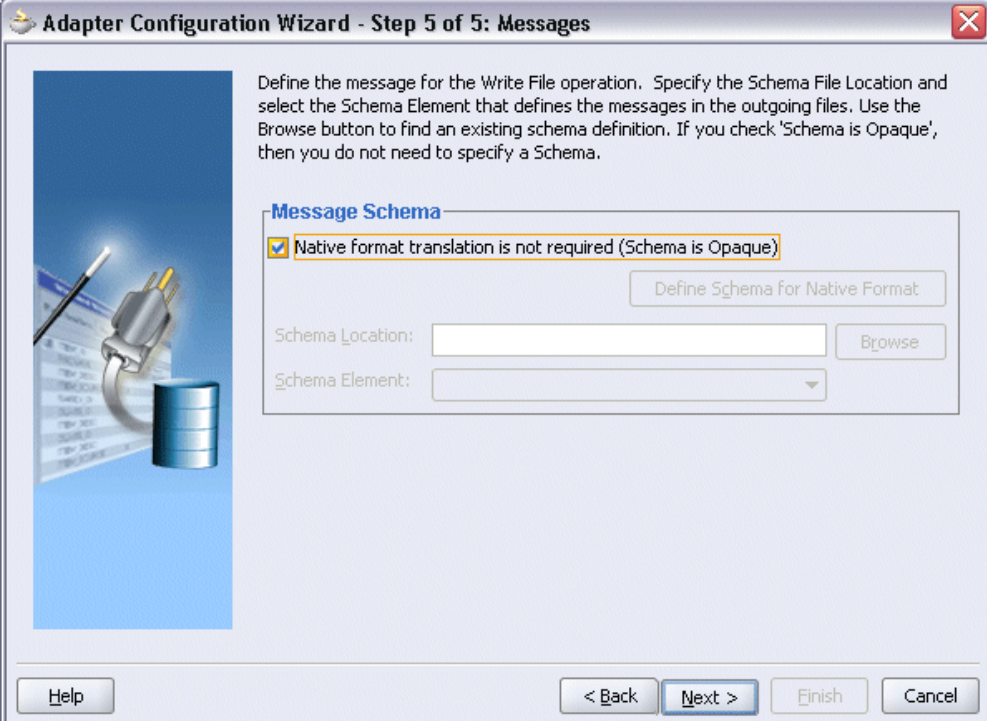
☐ Elapsed Time Exceeds: 1 minutes

☐ File Size Exceeds: 1000 kilobytes

Help < Back Next > Finish Cancel

6. For the Directory specified as field, select **Physical Path**. Enter directory path in the Directory for Outgoing Files field, and specify a naming convention for the output file such as PO\_%SEQ%.xml.
7. Confirm the default write condition: Number of Messages Equals 1. Click **Next**. The Messages dialog box appears.
8. Select **Native format translation is not required (Schema is Opaque)** check box.

### Specifying Message Schema



The screenshot shows a Windows-style dialog box titled "Adapter Configuration Wizard - Step 5 of 5: Messages". On the left is a vertical blue bar with a graphic of a USB cable and a database cylinder. To the right of the bar, there is instructional text: "Define the message for the Write File operation. Specify the Schema File Location and select the Schema Element that defines the messages in the outgoing files. Use the Browse button to find an existing schema definition. If you check 'Schema is Opaque', then you do not need to specify a Schema." Below this text is a section titled "Message Schema" containing a checked checkbox labeled "Native format translation is not required (Schema is Opaque)". To the right of this checkbox is a disabled button labeled "Define Schema for Native Format". Below these are two input fields: "Schema Location:" with a text box and a "Browse" button, and "Schema Element:" with a dropdown menu. At the bottom of the dialog are four buttons: "Help", "< Back", "Next >", and "Cancel".

Adapter Configuration Wizard - Step 5 of 5: Messages

Define the message for the Write File operation. Specify the Schema File Location and select the Schema Element that defines the messages in the outgoing files. Use the Browse button to find an existing schema definition. If you check 'Schema is Opaque', then you do not need to specify a Schema.

**Message Schema**

☒ Native format translation is not required (Schema is Opaque)

Define Schema for Native Format

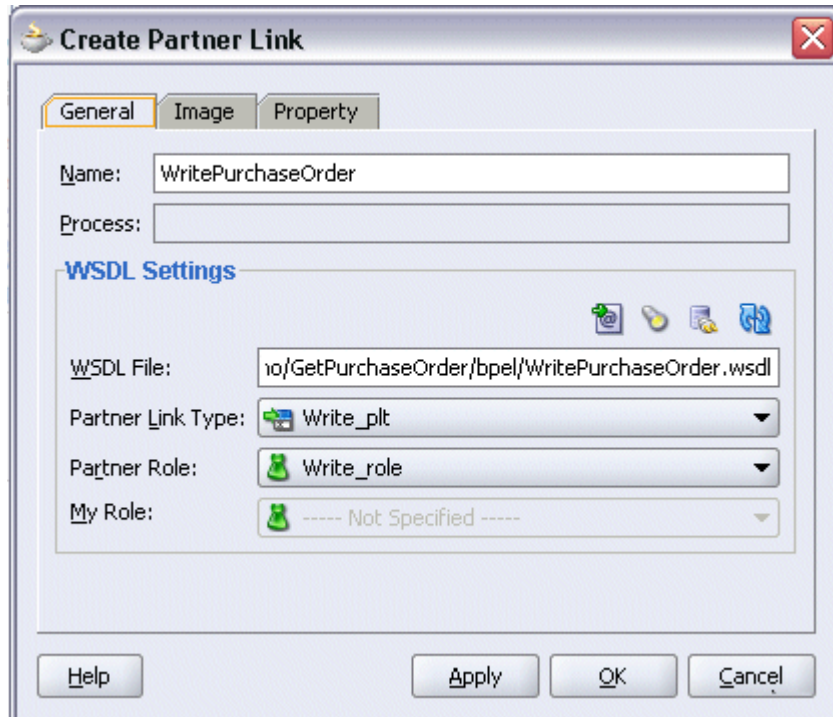
Schema Location:  Browse

Schema Element:

Help < Back Next > Finish Cancel

9. Click **Next** and 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 `WritePurchaseOrder.wsdl`.

### Completing the Partner Link Configuration



The image shows a 'Create Partner Link' dialog box with three tabs: 'General', 'Image', and 'Property'. The 'General' tab is selected. It contains the following fields and controls:

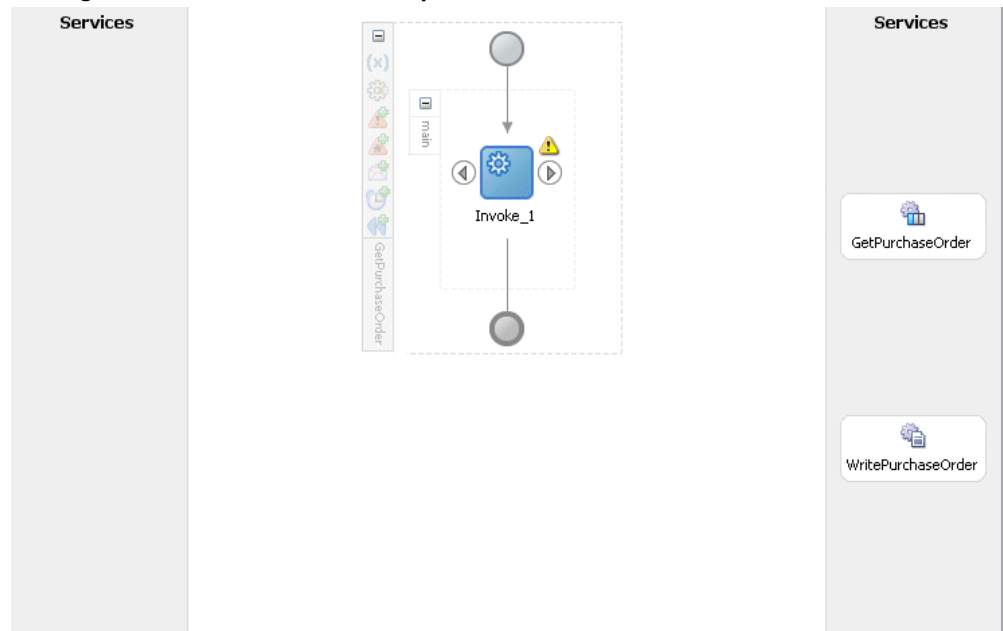
- Name:** WritePurchaseOrder
- Process:** (empty text box)
- WSDL Settings:** (section header with icons for file, key, and help)
- WSDL File:** io/GetPurchaseOrder/bpel/WritePurchaseOrder.wsdl
- Partner Link Type:** Write\_plt (dropdown menu)
- Partner Role:** Write\_role (dropdown menu)
- My Role:** ---- Not Specified ---- (dropdown menu)

At the bottom of the dialog are four buttons: 'Help', 'Apply', 'OK', and 'Cancel'.

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

The WritePurchaseOrder Partner Link appears in the BPEL process diagram.

### Adding the Partner Link for File Adapter



### Adding an Invoke Activity

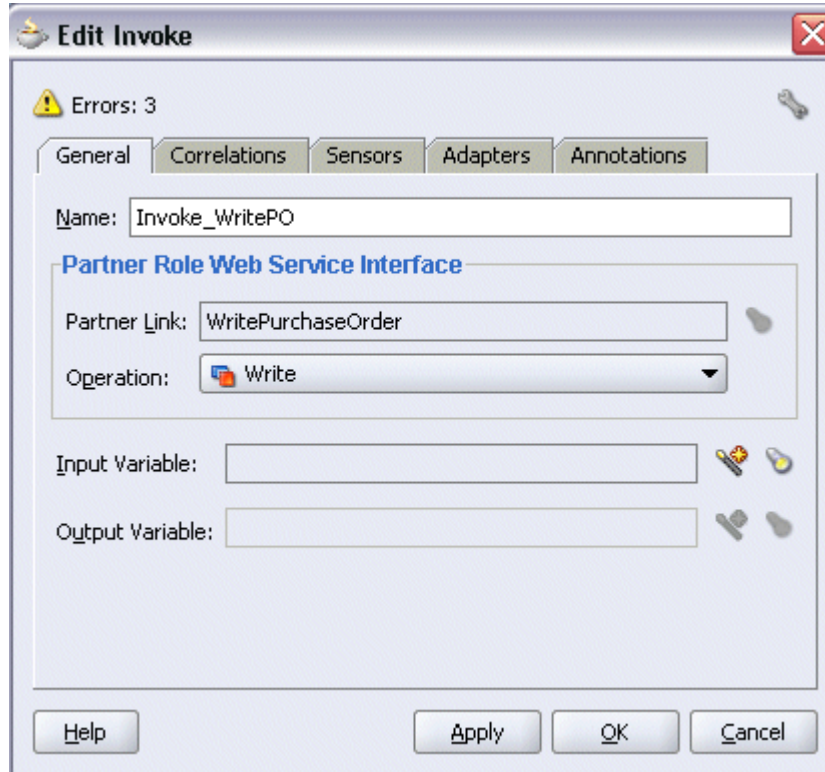
This step is to configure an Invoke activity to write the purchase order approved event details that is received from the Receive activity to the `WritePurchaseOrder` partner link in an XML file.

#### To add an 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, after the **Receive** activity.
2. Link the Invoke activity to the `WritePurchaseOrder` service. The Invoke activity will send event data to the partner link. The Edit Invoke dialog box appears.



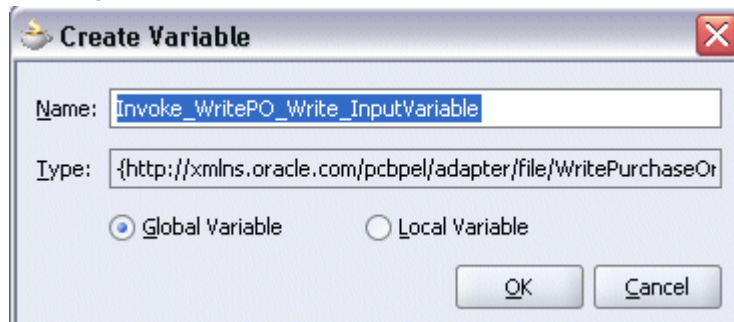
### Editing the Invoke Activity



The **Edit Invoke** dialog box is shown with the **General** tab selected. It features a **Name** field containing "Invoke\_WritePO". Below this is a section titled **Partner Role Web Service Interface** with a **Partner Link** field containing "WritePurchaseOrder" and an **Operation** dropdown menu set to "Write". At the bottom of this section are **Input Variable** and **Output Variable** fields, each with a **Create** icon (a lightbulb with a plus sign) to its right. The dialog also includes a **Help** button and **Apply**, **OK**, and **Cancel** buttons at the bottom.

3. Enter a name for the Invoke activity, and 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



The **Create Variable** dialog box is shown. It has a **Name** field containing "Invoke\_WritePO\_Write\_InputVariable" and a **Type** field containing "{http://xmlns.oracle.com/pcbpel/adapter/file/WritePurchaseOr". Below these fields are two radio buttons: **Global Variable** (which is selected) and **Local Variable**. At the bottom are **OK** and **Cancel** buttons.

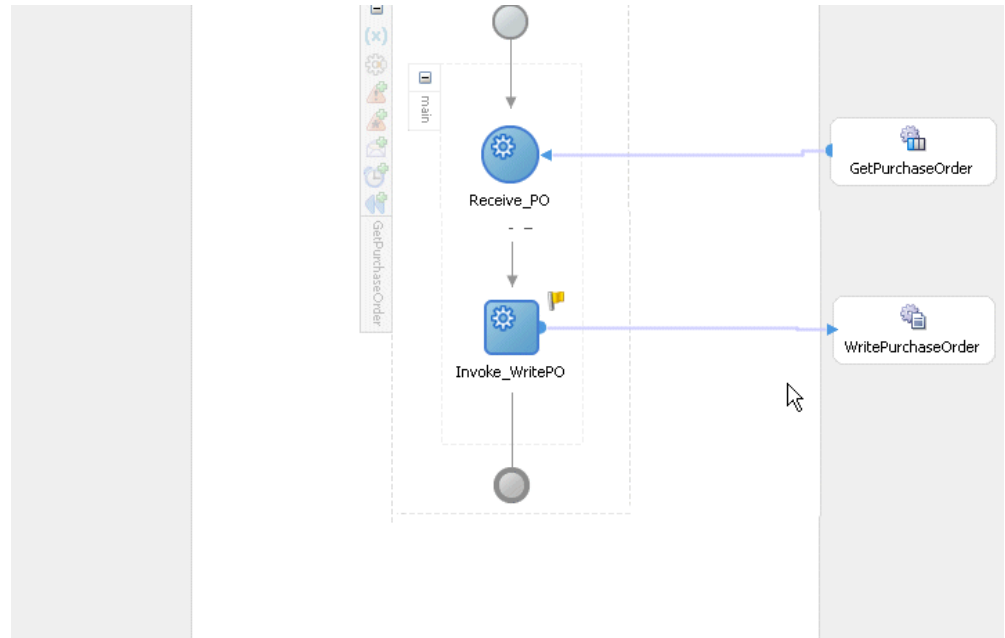
4. Select **Global Variable** and then enter a name for the variable. You can also accept the default name. Click **OK** to close the Create Variable dialog box.  
Click **Apply** and then **OK** in the Edit Invoke dialog box to finish configuring the



Invoke activity.

The Invoke activity appears in the process diagram.

### ***Process Diagram With Invoke Activity***



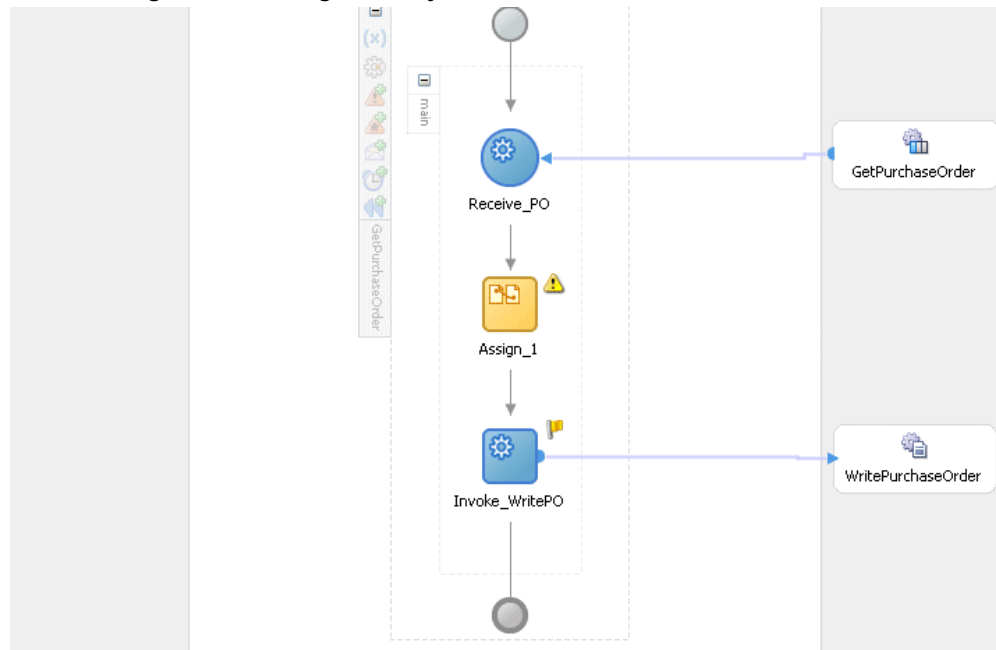
## **Adding an Assign Activity**

Use this step to pass the purchase order approved event details from the Receive activity to the Invoke activity.

**To add an Assign activity:**

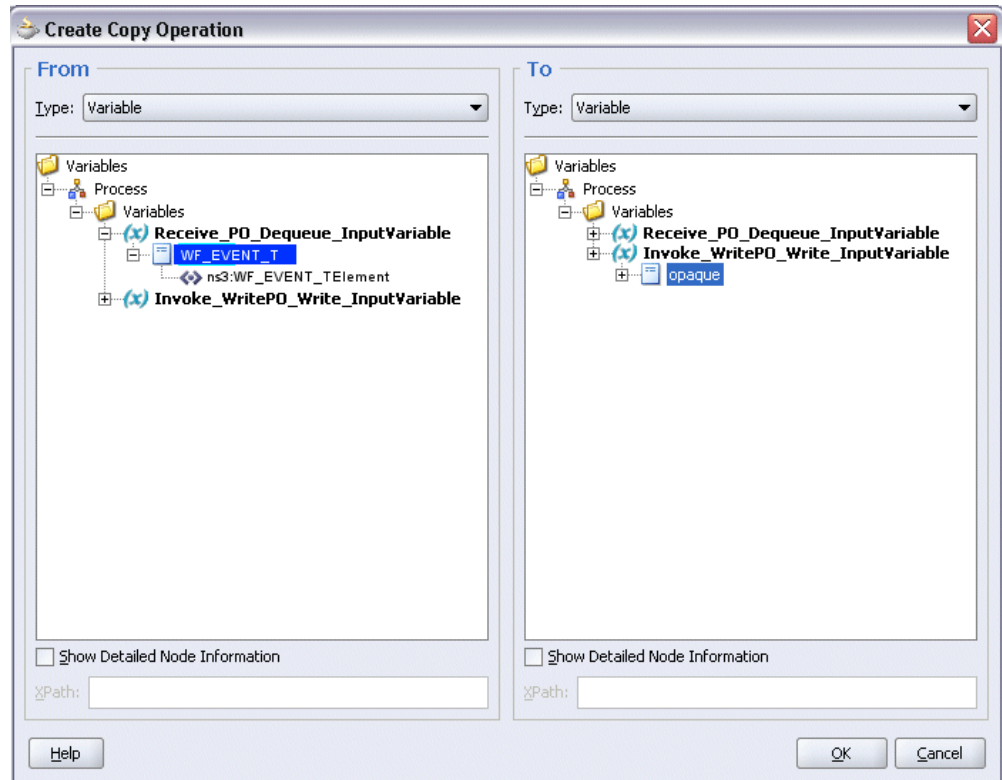
1. In JDeveloper BPEL Designer, drag and drop the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the **Receive** activity and the **Invoke** activity.

### Process Diagram with Assign Activity



2. Double-click the **Assign** activity to access the Edit Assign dialog box.
3. On the Copy Operation tab, click **Create** and then select **Copy Operation** from the menu. The Create Copy Operation dialog box appears.

### Defining the Copy Operation



4. In the From navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Receive\_PO\_DEQUEUE\_InputVariable > WF\_EVENT\_T** and select **ns3:WF\_EVENT\_T** element. The XPath field should contain your selected entry.
5. In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Invoke\_WritePO\_Write\_InputVariable > WF\_EVENT\_T** and select **Opaque**. The XPath field should contain your selected entry.
6. Click **OK** to close the Create Copy Operation dialog box.  
Click **Apply** and then **OK** in the Edit Assign dialog box to complete the configuration of the Assign activity.

## Deploying and Testing the BPEL Process at Run Time

After creating a BPEL process with the subscribed event, you can deploy it to a BPEL server if needed. To ensure that this process is modified or orchestrated appropriately, you can also test the BPEL process by initiating the business process contained in the BPEL process to test the interface integration.

### Prerequisites

Before deploying the BPEL process using Oracle JDeveloper, you must ensure that you have established the connectivity between the design-time environment and the run-time servers including the application server and the integration server.

How to configure the necessary server connection, see *Configuring Server Connection*, page B-1.

To validate the BPEL process, perform the following run-time tasks:

1. Deploy the BPEL process, page 5-26

Once you deploy the process to a BPEL server, it becomes available so that you can run the process manually to test it for validation.

2. Manually initiate the BPEL process, page 5-27

After deploying a BPEL process, you can manage the process from the BPEL console to manually initiate the business process and test the interface integration contained in your BPEL process.

## Deploying the BPEL Process

Before manually test the BPEL process, you first need to deploy it to the BPEL server.

**To deploy the BPEL process:**

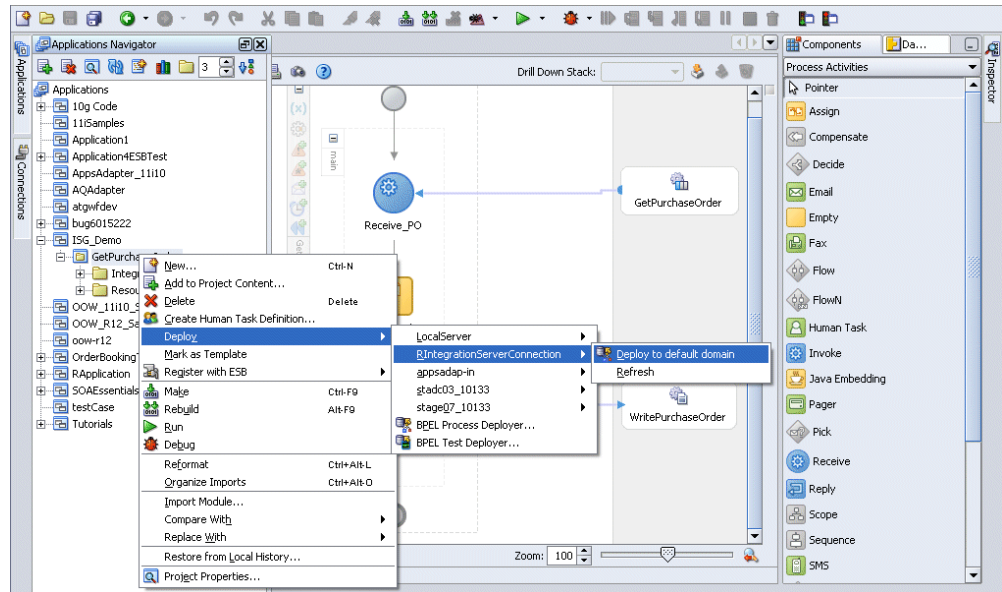
1. In the Applications Navigator of JDeveloper BPEL Designer, select the **GetPurchaseOrder** project.
2. Right-click the project and click **Make** action from the menu.

Look for any compilation error messages in Message Log.

Right-click the project and select **Deploy >Integration Server Connection name > Deploy to Default Domain** action from the menu.

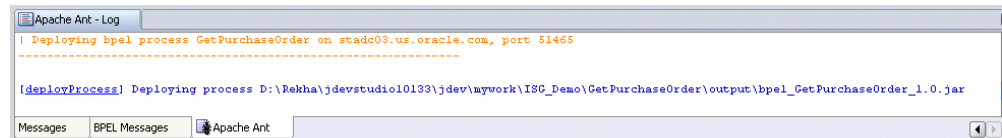
For example, you can select **Deploy > BPELServerConn > Deploy to Default Domain** to deploy the process if you have the BPEL Process Manager setup appropriately.

## Deploying the BPEL Process



3. Look for 'Build successful' message in Apache Ant – Log to ensure that the BPEL project is compiled and successfully deployed.

## Compilation and Deployment Message Logs



## Testing the BPEL Process

To validate whether the BPEL process that you created works or not, you need to manually initiate the process after it has been successfully deployed to the BPEL console. Therefore, the validation starts with the BPEL console to ensure that you can find the deployed BPEL process listed in the console. Then, you can log on to Oracle E-Business Suite to manually initiate the purchase order approval and acknowledgement processes and to confirm that the relevant event is raised and the updated purchased order details is also written in the XML file.

To manually test the BPEL process:

1. Log into Oracle Application Server 10g BPEL Console (<http://<soaSuiteServerHostName>:<port>/BPELConsole>). The BPEL Console login page appears.

2. Enter the username and password and click **Login**.

The Oracle Enterprise Manager 10g BPEL Control appears.

3. In the BPEL Console, confirm that `GetPurchaseOrder` has been deployed.

### Deployed BPEL Processes

ORACLE Enterprise Manager 10g BPEL Control			
		Manage BPEL Domain	Logout   Support
		Logged to domain: <b>default</b>	
Dashboard		BPEL Processes	Instances
Activities			
Deployed BPEL Processes		In-Flight BPEL Process Instances	
Name	Instance	BPEL Process	Last Modified ↑
API_1			
BPELProcess_BE			
BPELProcess_BESubscription			
BPEvent			
<a href="#">GetPurchaseOrder</a>			
Plsql			
SamplePOInboundXMLG			
TaskActionHandler			
TaskManager			
bug_66292			
psitest2			
© Deploy New Process		Recently Completed BPEL Process Instances ( <a href="#">More...</a> )	
		90005 : Instance #90005 of bug_66292	bug_66292 (v. 1.0) 6/6/08 4:01:29 AM
		90004 : Instance #90004 of bug_66292	bug_66292 (v. 1.0) 6/6/08 3:43:46 AM
		90003 : Instance #90003 of bug_66292	bug_66292 (v. 1.0) 6/6/08 3:20:24 AM
		90002 : Instance #90002 of bug_66292	bug_66292 (v. 1.0) 6/6/08 3:20:12 AM
		90001 : Instance #90001 of bug_66292	bug_66292 (v. 1.0) 6/6/08 3:20:12 AM
Oracle BPEL Console v10.1.3.3.0			

4. Log on to Oracle E-Business Suite with the XML Gateway responsibility.

This is to ensure that the XML Gateway trading partner is set up correctly so that a purchase order can have a valid supplier that has been defined.

5. Select Define Trading Partner from the navigation menu to access the Trading Partner Setup window.

6. 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: Enter a trading partner site information. For example, 2000 Century Way, Santa Clara, CA 95613-4565
- Company Admin Email: Enter a valid email address.

7. Enter the following trading partner details:

- Transaction Type: PO

- 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: SOAP

Transaction Type	Transaction SubType	Standard Code	External Transaction Type	External Transaction SubType	Direction	Map	Connection/Hub	Protocol Type
PO	PRO	OAG	PO	PROCESS	OUT	itg_process_p	DIRECT	SOAP

8. Save the trading partner details. Switch responsibility back to Purchasing, Vision Operations (USA) and select Purchase Order from the navigation menu.
9. Create a purchase order with the header values reflecting the trading partner you previously defined in the Purchase Order window:
  - Supplier: Enter a supplier information, such as 'Advanced Network Devices'
  - Site: Select a site information, such as 'SANTA CLARA-ERS'

10. On the Lines tab, enter a data row with the following values:
  - Type: Goods
  - Item: CM13139
  - Quantity: 1
  - Description: Hard Drive - 8GB
  - Promised: Enter any future date in the format of dd-mmm-yyyy (such as 23-JUN-2008)
11. Save your purchase order. The status of the purchase order is 'Incomplete'.
 

**Note:** Because the trading partner is set up and valid, the transmission method is automatically set to **XML**.
12. Click **Approve** to approve the purchase order.

The screenshot shows the 'Purchase Orders - 5789' window. The 'Operating Unit' is 'Vision Operations', 'PO, Rev' is '5789', and 'Type' is 'Standard Purchase Order'. The 'Supplier' is 'Advanced Network Devices', 'Ship-To' is 'M1- Seattle Mfg', and 'Buyer' is 'Stock, Ms. Pat'. The 'Status' is 'Approved'. The 'Description' field is empty. The 'Lines' tab is selected, showing a table with one line item: '1', 'Goods', 'CM13139', 'Rev', 'Job', 'Category', 'Description', 'UOM', 'Quantity', 'Price'. The 'Description' is 'Hard Drive - 8GB' and the 'Price' is '150.393'. The 'Total' is '150.39'. The 'Approve...' button is visible at the bottom right.

Num	Type	Item	Rev	Job	Category	Description	UOM	Quantity	Price
1	Goods	CM13139			PRODUCTN.DRN	Hard Drive - 8GB	Each	1	150.393

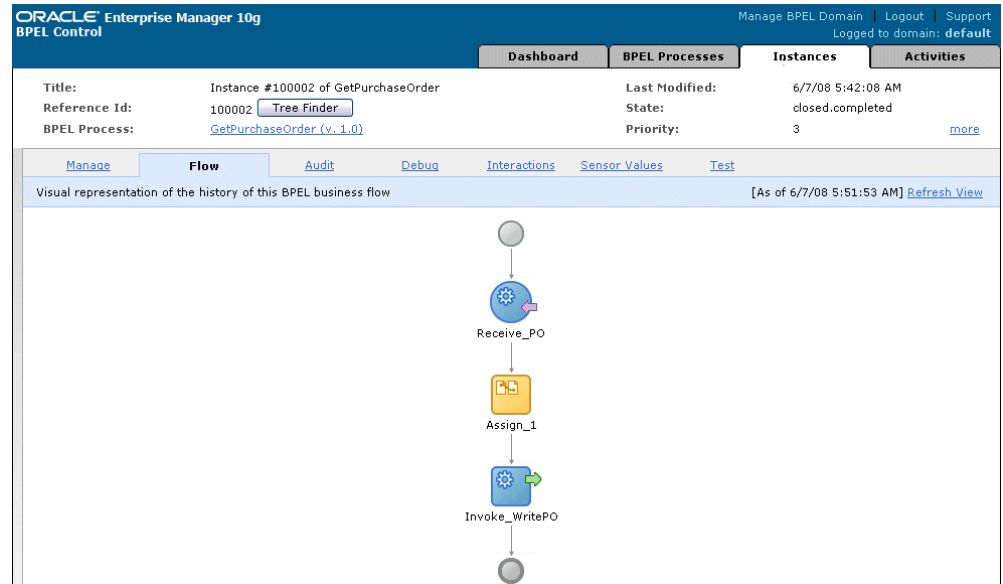
The status of the purchase order is now changed to 'Approved'. For future reference, note the value of the PO, Rev field (for example, the PO number 5789).

Once the purchase order is approved, the business event `oracle.apps.po.event.xmlpo` is raised.

13. Log into Oracle BPEL Process Manager, and return to the BPEL Console to confirm that the `GetPurchaseOrder` process has been completed.



To verify, select the instance of your deployed process which opens up in the Instances tab of your selected BPEL process.



14. Double-click on the Receive activity in the BPEL process diagram and click the View XML document link to open the XML file. Please note that the purchase order (number 5789) has been received.

### Examining the Receive Event Name

```
<?xml version="1.0" encoding="UTF-8" ?>
- <WF_EVENT_T xmlns="http://xmlns.oracle.com/xdb/APPS/GetPurchaseOrder/GetPurchaseOrder">
  <PRIORITY xmlns="">50</PRIORITY>
  <SEND_DATE xmlns="">2008-06-07T05:36:41.000-07:00</SEND_DATE>
  <RECEIVE_DATE xmlns="">2008-06-07T05:38:30.000-07:00</RECEIVE_DATE>
  <CORRELATION_ID xmlns="" />
- <PARAMETER_LIST xmlns="">
  - <PARAMETER_LIST_ITEM>
    <NAME>APPLICATION_ID</NAME>
    <VALUE>201</VALUE>
  </PARAMETER_LIST_ITEM>
  - <PARAMETER_LIST_ITEM>
    <NAME>DOCUMENT_DIRECTION</NAME>
    <VALUE>OUT</VALUE>
  </PARAMETER_LIST_ITEM>
  - <PARAMETER_LIST_ITEM>
    <NAME>DOCUMENT_NO</NAME>
    <VALUE>5789</VALUE>
  </PARAMETER_LIST_ITEM>
  - <PARAMETER_LIST_ITEM>
    <NAME>ECX_DEBUG_LEVEL</NAME>
    <VALUE>0</VALUE>
  </PARAMETER_LIST_ITEM>
  - <PARAMETER_LIST_ITEM>
    <NAME>ECX_DOCUMENT_ID</NAME>
    <VALUE>5789:0:204</VALUE>
  </PARAMETER_LIST_ITEM>
  - <PARAMETER_LIST_ITEM>
    <NAME>ECX_PARAMETER1</NAME>
    <VALUE />
  </PARAMETER_LIST_ITEM>
  - <PARAMETER_LIST_ITEM>
    <NAME>ECX_PARAMETER2</NAME>
    <VALUE>0</VALUE>
  </PARAMETER_LIST_ITEM>
</PARAMETER_LIST>
</WF_EVENT_T>
```

15. Examine the Assign and Invoke activities as well for the event raised and document number.
16. 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 the same as that of the approved purchase order.

### Confirming the Output Order Number

```
<PRIORITY xmlns="">50</PRIORITY>
<SEND_DATE xmlns="">2008-06-07T05:36:41.000-07:00</SEND_DATE>
<RECEIVE_DATE xmlns="">2008-06-07T05:38:30.000-07:00</RECEIVE_DATE>
<CORRELATION_ID xmlns="">/>
<PARAMETER_LIST xmlns="">
  <PARAMETER_LIST_ITEM>
    <NAME>APPLICATION_ID</NAME>
    <VALUE>201</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>DOCUMENT_DIRECTION</NAME>
    <VALUE>OUT</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>DOCUMENT_NO</NAME>
    <VALUE>9788</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECX_DEBUG_LEVEL</NAME>
    <VALUE>0</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECX_DOCUMENT_ID</NAME>
    <VALUE>5789:0:204</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECX_PARAMETER1</NAME>
    <VALUE/>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECX_PARAMETER2</NAME>
    <VALUE>0</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECX_PARAMETER3</NAME>
    <VALUE>1318:50578:201</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECX_PARAMETER4</NAME>
    <VALUE>97094</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECX_PARAMETERS</NAME>
    <VALUE>204</VALUE>
  </PARAMETER_LIST_ITEM>
  <PARAMETER_LIST_ITEM>
    <NAME>ECX_PARTY_ID</NAME>
```



---

# Using Concurrent Programs

## Overview

A concurrent program is an instance of an execution file with associated parameters. Concurrent programs use a concurrent program executable to locate the correct execution file. The execution file can be an operating system file or database stored procedure which contains your application logic (such as PL/SQL, Java). Several concurrent programs may use the same execution file to perform their specific tasks, each having different parameter defaults.

The concurrent program can be exposed as a Web service based integration interface. An integration repository administrator can further deploy a generated service from Oracle Integration Repository to the application server.

This deployed service can be exposed to customers through service provider and invoked through any of the Web service clients.

For example, an integration developer can take a deployed Web service WSDL URL and directly use it to define a partner link for the Web service that a BPEL process connects to in order to perform tasks, or carry information between the Web service and the BPEL process.

Detailed information on how to create a BPEL process to invoke the Web service and use it to update Oracle E-Business Suite is discussed in this chapter. For the example described in the following sections, we use Oracle JDeveloper 10.1.3.3.0 as a design-time tool to create the BPEL process and use Oracle SOA Suite BPEL server 10.1.3.3.0 for the process deployment.

## Using Concurrent Program WSDLs at Design Time

### BPEL Process Scenario

This example uses Departure Shipment Notice Outbound WSHDSNO concurrent program to explain the BPEL process creation.

When a shipment notice generation request is received as an input to the BPEL process, a sales order information including header and line items are read by a File Adapter. The sales order data is then passed through to create a departure shipment notice (DSNO). The shipment notice creation document number will be passed back to the request application.

If the BPEL process is successfully executed after deployment, you should be able to validate if the generated shipment notice has correct trading partner information as described in the sales order.

### **Prerequisites to Create a BPEL Process Using a Concurrent Program Web Service**

Before performing design-time tasks for concurrent programs, you need to ensure the following tasks are in place:

- An integration repository administrator needs to successfully deploy the generated concurrent program Web service to the application server.
- An integration developer needs to locate and record the deployed WSDL URL for the concurrent program exposed as a Web service.
- SOAHeader elements should be populated in order to run the concurrent program for SOAP request

### ***Deploying the Concurrent Program WSDL URL***

An integration repository administrator must first create a Web service for the selected interface definition, and then deploy the service from Oracle Integration Repository to the application server.

For example, the administrator must perform the following steps before letting integration developers use the deployed WSDL in creating a BPEL process:

1. To generate a Web service, locate the interface definition first from the Oracle Integration Repository (such as Departure Shipment Notice Outbound WSHDSNO concurrent program ) and click **Generate WSDL** in the interface details page. The Web Service - SOA Provider region appears in the interface details page. For detailed instruction on how to generate a Web service, see *Generating Web Services, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.
2. To deploy a generated Web service, select at least one authentication type and click **Deploy** in the Web Service - SOA Provider region of the interface details page to deploy the service.

Once the service is successfully deployed, the selected authentication type(s) will be displayed along with 'Deployed' Web Service Status. For more information on securing Web services with authentication types, see *Managing Web Service Security, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

For detailed instruction on how to deploy a Web service, see *Deploying Web Services, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

## Searching and Recording WSDL URL

Apart from the required tasks performed by the administrator, an integration developer also needs to log on to the system to locate and record the deployed Web service WSDL URL for the interface (such as WSHDSNO concurrent program) that needs to be orchestrated into a meaningful business process in Oracle JDeveloper using BPEL language.

This WSDL information will be used directly for a partner link during the BPEL process creation at design time.

## Confirming and Recording a Deployed WSDL URL

The screenshot displays the Oracle Integration Repository interface. At the top, the header includes the Oracle logo and 'Integration Repository' text. Navigation links for 'Diagnostics', 'Home', 'Logout', 'Preferences', 'Help', and 'Personalize Page' are visible. The main content area is titled 'Concurrent Program : Departure Shipment Notice Outbound'. It features a metadata section with fields for Internal Name (WSHDSNO), Type (Concurrent Program), Product (Shipping Execution Common), Status (Active), and Business Entity (Trip). A 'Scope' section indicates 'Interface Source' is 'Public Oracle'. Below this is a 'Full Description' section with a paragraph about the program's function. The 'Web Service - SOA Provider' section shows 'Web Service Status' as 'Deployed' and 'WSDL' as 'View WSDL'. The 'Authentication Type' section has checkboxes for 'Username Token' and 'SAML Token (Sender Vouches)'. The 'Source Information' section lists 'Source File' as 'patch/115/import/US/wshprg.ldt', 'Source Version' as '120.22.12010000.5', and 'Source Product' as 'WSH'. The 'Parameters' section contains a table with columns 'Name', 'Type', 'Required', 'Displayed', and 'Description'. The 'Methods' section includes a table with columns 'Name', 'Internal Name', 'Status', and 'Description'. At the bottom, there are 'Browse', 'Search', and 'Printable Page' buttons, and a footer with 'About this Page', 'Privacy Statement', and 'Copyright (c) 2006, Oracle. All rights reserved'.

**Internal Name:** WSHDSNO  
**Type:** Concurrent Program  
**Product:** Shipping Execution Common  
**Status:** Active  
**Business Entity:** [Trip](#)

**Scope:** Public Oracle

**Full Description**

This concurrent program generates a Departure Ship Notice Outbound(DSNO). DSNO is a notice used by the supplier to provide advance notification of a shipment to the buyer or trading partner that the material has been shipped. The Outbound Ship Notice/Manifest transaction serves business needs of both the sender and the receiver of the transaction.

**Web Service - SOA Provider**

Web Service Status: **Deployed**  
WSDL: [View WSDL](#)

\* Authentication Type: ☐ Username Token ☒ SAML Token (Sender Vouches)

**Source Information**

Source File: patch/115/import/US/wshprg.ldt  
Source Version: 120.22.12010000.5  
Source Product: WSH

**Parameters**

Name	Type	Required	Displayed	Description
Trip	WSH_SRS_TRIPS	Yes	Yes	Trip
Trip Stop	WSH_SRS_TRIP_REL_STOPS	Yes	Yes	Trip Stop

**Methods**

Name	Internal Name	Status	Description
<a href="#">Process</a>	Process	Active	Departure Ship Notice Outbound

How to search for an interface and review the interface details, see Searching and Viewing Integration Interfaces, page 2-1.

## Setting Variables in SOAHeader for SOAP Request

You must populate certain variables in the BPEL process for SOAHeader elements to pass values that may be required to set application context during service execution. These SOAHeader elements for concurrent program interface type are *Responsibility*, *RespApplication*, *SecurityGroup*, *NLSLanguage*, and

*Org\_Id.*

**Note:** The user information is defined by the `wsseUsername` property passed within the security headers. Detailed instructions on how to pass the security headers along with the SOAP request, see *Passing Values to Security Headers*, page 6-9.

The expected values for these elements are described in the following table:

***Header Variables and Expected Values for Concurrent Program Interface Type***

Element Name	Expected Value
Responsibility	<code>responsibility_key</code> (such as "SYSTEM_ADMINISTRATOR")
RespApplication	Application Short Name (such as "FND")
SecurityGroup	Security Group Key (such as "STANDARD")
NLSLanguage	NLS Language (such as "AMERICAN")
Org_Id	Org Id (such as "202")

**Note:** NLS Language and Org\_Id are optional values to be passed.

- If the NLS Language element is specified, SOAP requests can be consumed in the language passed. All corresponding SOAP responses and error messages can also be returned in the same language. If no language is identified, then the default language of the user will be used.
- If a service execution is dependent on any particular organization, then you must pass the Org\_Id element of that SOAP request.

The context information can be specified by configuring an Assign activity before the Invoke activity in the BPEL PM.

**BPEL Process Creation Flow**

Based on the scenario, the following design-time tasks are discussed in this chapter:

1. Create a new BPEL project, page 6-5

Use this step to create a new BPEL project called `ShipNotice.bpel`. This



automatically creates two dummy activities - Receive and Reply - to receive input from a third party application and to reply output of the BPEL process back to the request application.

2. Create a Partner Link, page 6-7

Use this step to create a partner link for the Departure Shipment Notice Outbound `Shipment_Notice` concurrent service.

3. Add a Partner Link for File Adapter, page 6-10

This is to synchronous read sales order details received from the trading partner.

4. Add Invoke activities, page 6-15

Use this step to create two Invoke activities in order to:

1. Point to the File Adapter - Synchronous Read operation to read the order details from the Assign activity.
2. Point to the `Shipment_Notice` Web service to create the shipment notice with header and line details.

5. Add Assign activities, page 6-19

Use this step to create three Assign activities in order to:

1. To pass the `SOAHeader` variables for the invocation of the `DSNO` concurrent program service.
2. To pass the order details from the output of the Synchronous Read - File Adapter service to the input of the `DSNO` creation.
3. To set the SOAP response to output.

For general information and basic concept of a BPEL process, see Understanding BPEL Business Processes, page D-1 and *Oracle BPEL Process Manager Developer's Guide* for details.

## Creating a New BPEL Project

Use this step to create a new BPEL project that will contain various BPEL process activities.

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

#### *Entering BPEL Project Information*

**BPEL Project Creation Wizard - Project Settings**

The BPEL Project Creation Wizard allows you to create a project in which you can design a business process based on the BPEL (Business Process Execution Language) standard.

Please specify the process name and project settings below.

Name:

Namespace:

☒ **Use Default Project Settings**

Project Name:

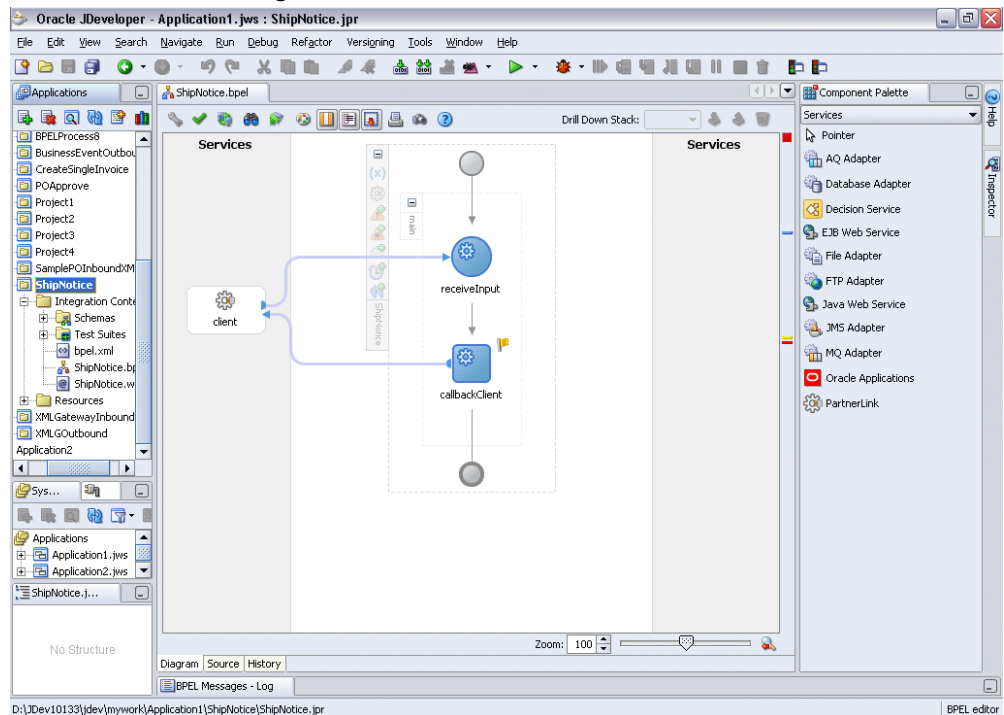
Project Directory:

Template:

7. In the **Name** field, enter a descriptive name for example `ShipNotice`.
8. From the Template list, select **Synchronous BPEL Process** and then select **Use Default Project Settings**.
9. Use the default input and output schema elements in the Input/Output Elements dialog box.
10. Click **Finish**.

A new synchronous BPEL process is created with the Receive and Reply activities. The required source files including `bpel.xml`, using the name you specified (for example, `ShipNotice.bpel`) are also generated.

## New BPEL Process Diagram



## Creating a Partner Link for the Web Service

Use this step to create a Partner Link called `Shipment_Notice` for the Web service exposed through WSHDSNO concurrent program.

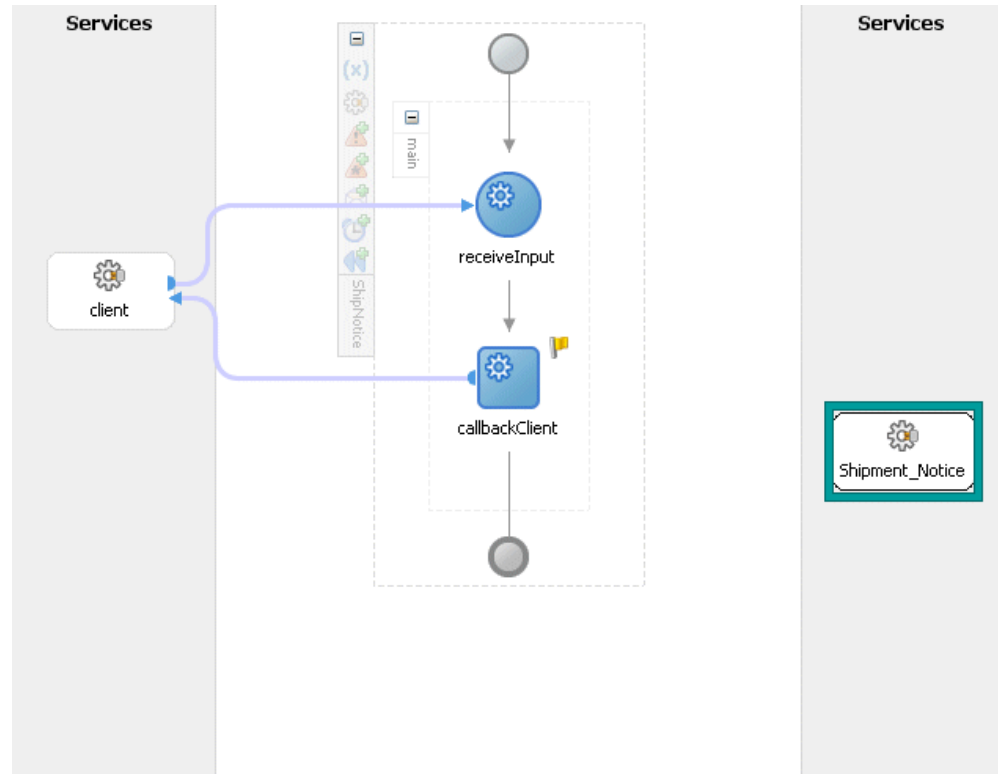
**To create a partner link for Shipment\_Notice Web service:**

1. In JDeveloper BPEL Designer, drag and drop the **PartnerLink** service from the Component Palette into the Partner Link border area of the process diagram. The Service Name dialog box appears.
2. Copy the WSDL URL corresponding to the Departure Shipment Notice Outbound WSHDSNO service that you recorded earlier in the WSDL File field. Click **OK**.
3. A Partner Link Type message dialog box appears asking whether you want the system to create a new WSDL file that will by default create partner link types for you.

Click **Yes** to have the Partner Name value populated automatically.

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.

#### Adding the Partner Link



4. You can optionally change the default partner link name by double-clicking the icon to open the Edit Partner Link window. For example, change it from `WSHDSNO` to `Shipment_Notice`.

Select the Partner Role value from the drop-down list. Click **Apply**.

### Editing Partner Link Parameters

**Edit Partner Link**

Errors: 1

General Image Property

Name: Shipment\_Notice

Process: ShipNotice

**WSDL Settings**

WSDL File: file:/D:/JDev10133/jdev/mywork/Application1/ShipNot

Partner Link Type: WSHDSNO\_PortType\_PL

Partner Role: WSHDSNO\_PortType\_Role

My Role: ---- Not Specified ----

Help Apply OK Cancel

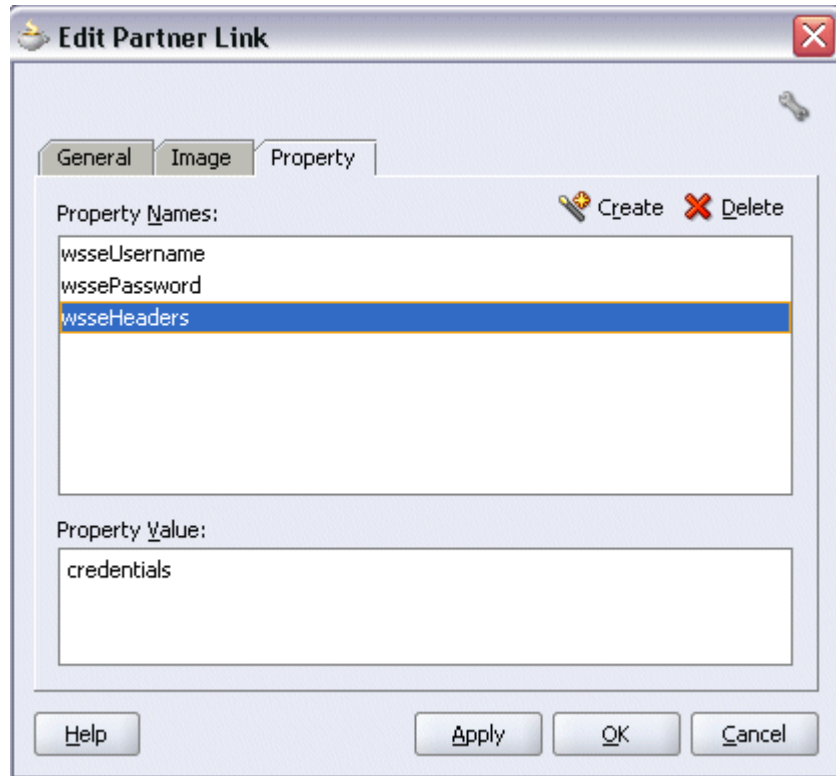
#### 5. Passing Values to Security Headers

Select the Property tab and click the **Create Property** icon to select the following properties from the property name drop-down list in order to pass the security headers along with the SOAP request:

- wsseUsername  
Specify the username to be passed in the Property Value box.
- wssePassword  
Specify the corresponding password for the username to be passed in the Property Value box.
- wsseHeaders  
Enter `credentials` as the property value.

Click **Apply** to save the selected property values.

### Adding Properties



6. Click **OK** to complete the partner link configuration.

## Adding a Partner Link for File Adapter

Use this step to configure a BPEL process by synchronously reading a sales order to obtain the order details.

### To add a Partner Link for File Adapter to read order details:

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 welcome page appears.
2. Click **Next**. The Service Name dialog box appears.
3. Enter a name for the file adapter service, for example `ReadOrder`. You can add an optional description of the service.
4. Click **Next** and the Operation dialog box appears.

### Specifying the Operation

**Adapter Configuration Wizard - Step 2 of 5: Operation**

The File Adapter supports three operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, and a Synchronous Read File operation that reads the current contents of a file. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type: ☐ Read File ☐ Write File ☒ Synchronous Read File

Operation Name:

Help < Back Next > Finish Cancel

5. Specify the operation type, for example **Synchronous Read File**. This automatically populates the **Operation Name** field.

Click **Next** to access the File Directories dialog box.

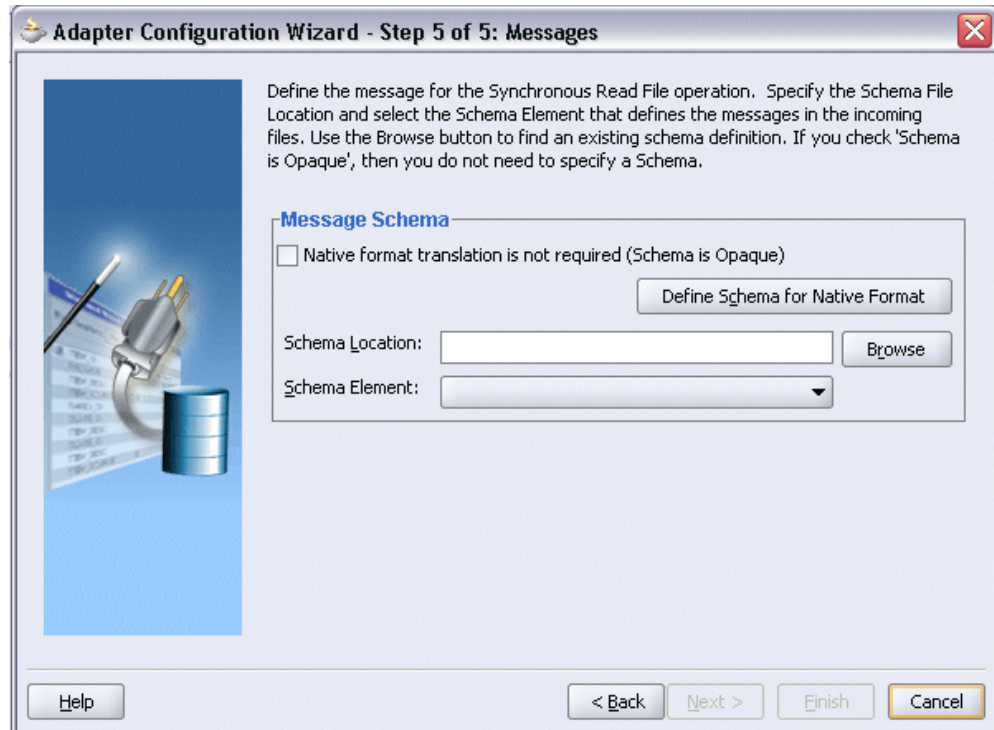
6. Select **Physical Path** radio button and enter the input payload file directory information. For example, enter `/usr/tmp/` as the directory name.

Uncheck the **Delete Files after successful retrieval** check box. Click **Next** to open the File Name dialog box.

7. Enter the name of the file for the synchronous read file operation. For example, enter `'order_data.xml'`. Click **Next**. The Messages dialog box appears.



## Specifying Message Schema



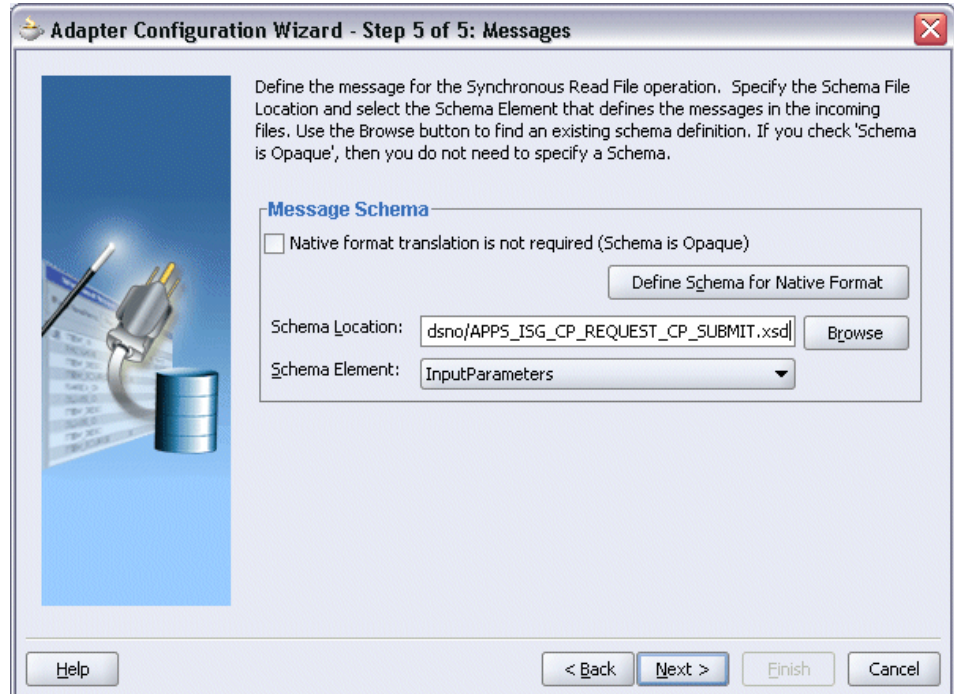
8. Select **Browse** for schema file in Schema Location. The Type Chooser window is displayed.
  1. Click **Import Schema Files** button on the top right corner of the Type Chooser window.
  2. Enter the schema location for the service. Such as  
`http://<myhost>:<port>/webservicess/SOAPProvider/concurrentprogram/wshdsno/APPS_ISG_CP_REQUEST_CP_SUBMIT.xsd`.  
  
Schema location for your service can be found from the service WSDL URL (for example,  
`http://<myhost>:<port>/webservicess/SOAPProvider/concurrentprogram/wshdsno/?wsdl`).
  3. Select the **Add to Project** check box and click **OK**.
  4. Click **OK** for Import schema prompt.  
  
The Imported Schemas folder is automatically added to the Type Chooser window.
  5. Select schema element by expanding the Imported Schemas folder > `APPS_ISG_CP_REQUEST_CP_SUBMIT.xsd` > `InputParameters`.



Click **OK**.

The selected schema location and element values are displayed.

#### **Viewing Selected Message Schema and Element**



9. Click **Next** and 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 `ReadOrder.wsdl`.

### Completing the Partner Link Configuration

**Edit Partner Link**

General Image Property

Name: ReadOrder

Process: ShipNotice

**WSDL Settings**

WSDL File: file:/D:/JDev10133/jdev/mywork/Application1/ShipNoI

Partner Link Type: SynchRead\_plt

Partner Role: SynchRead\_role

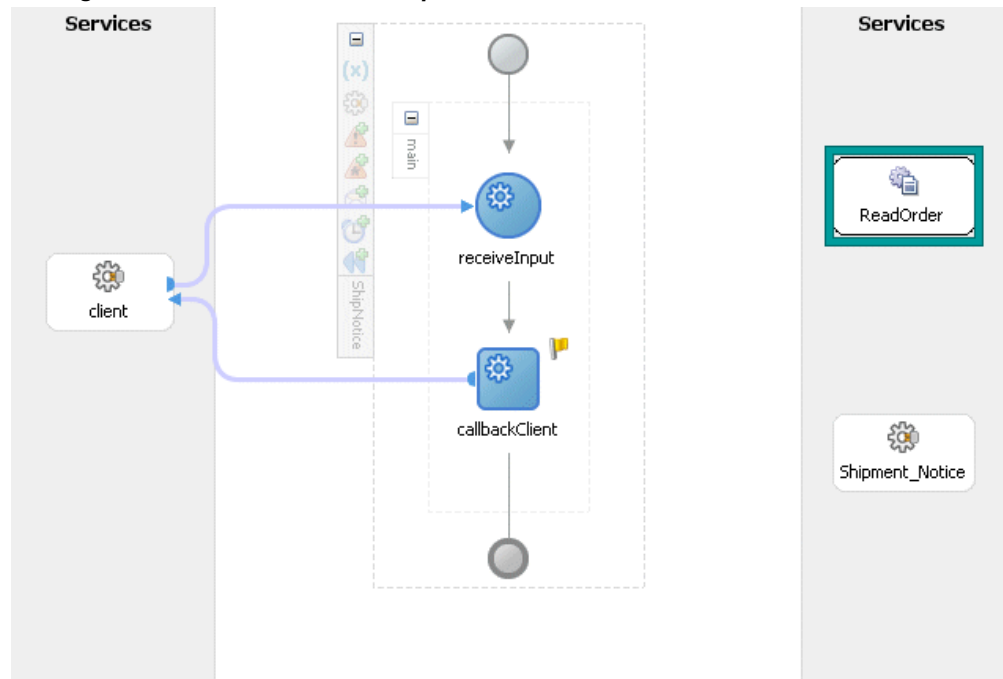
My Role: ----- Not Specified -----

Help Apply OK Cancel

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

The ReadOrder Partner Link appears in the BPEL process diagram.

### Adding the Partner Link for File Adapter



## Adding Invoke Activities

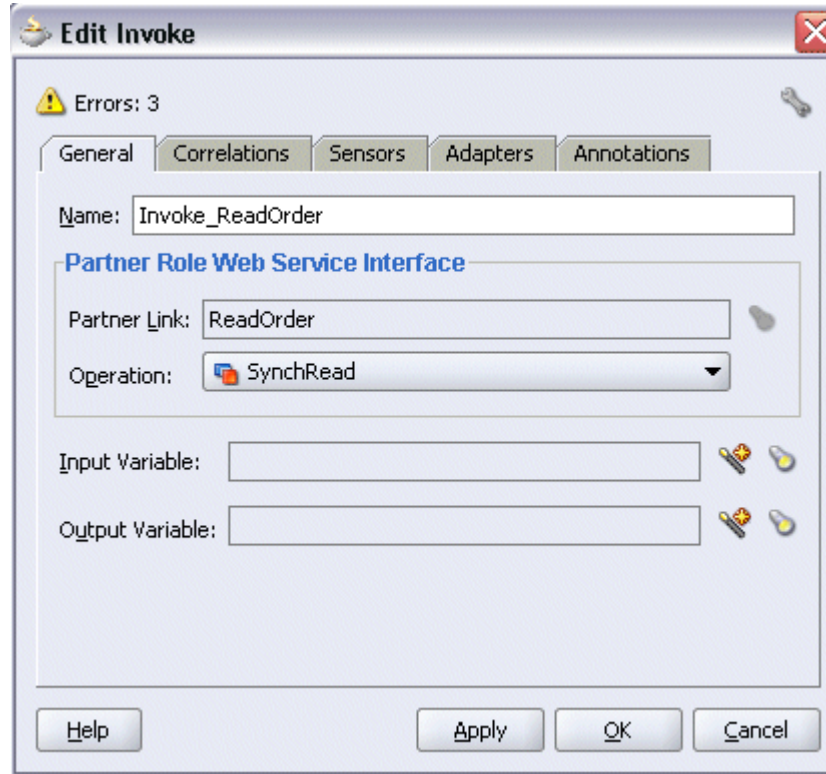
This step is to configure two Invoke activities:

- Read order details that is passed from the first Assign activity through the `ReadOrder` partner link for File Adapter.
- Send the order header and line details received from the Assign activities to generate an outbound shipment notice (DSNO) by using the `Shipment_Notice` partner link.

### To add an Invoke activity for ReadOrder Partner Link:

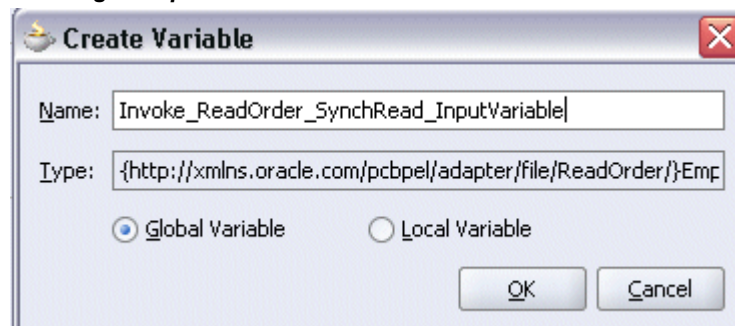
1. In JDeveloper BPEL Designer, drag and drop the **Invoke** activity from the Component Palette into the Activity box of the process diagram, between the **Receive** and **Reply** activities.
2. Link the Invoke activity to the `ReadOrder` service. The Invoke activity will send order data to the partner link. The Edit Invoke dialog box appears.

### Editing the Invoke Activity



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

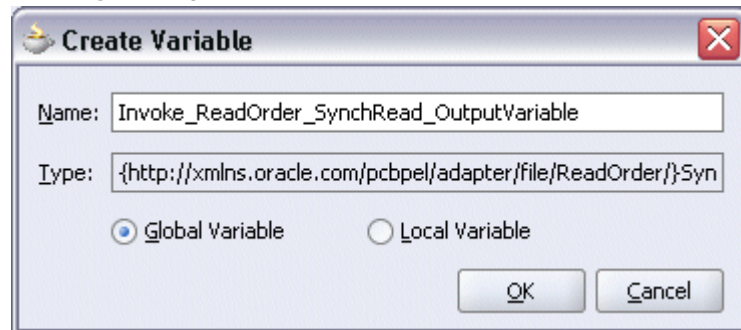
### Creating an Input Variable



4. Select **Global Variable** and then enter a name for the variable. You can also accept the default name. Click **OK**.
5. Click the **Create** icon next to the **Output Variable** field. Select **Global Variable** and

then enter a name for the variable. You can also accept the default name. Click **OK**.

#### **Creating an Output Variable**



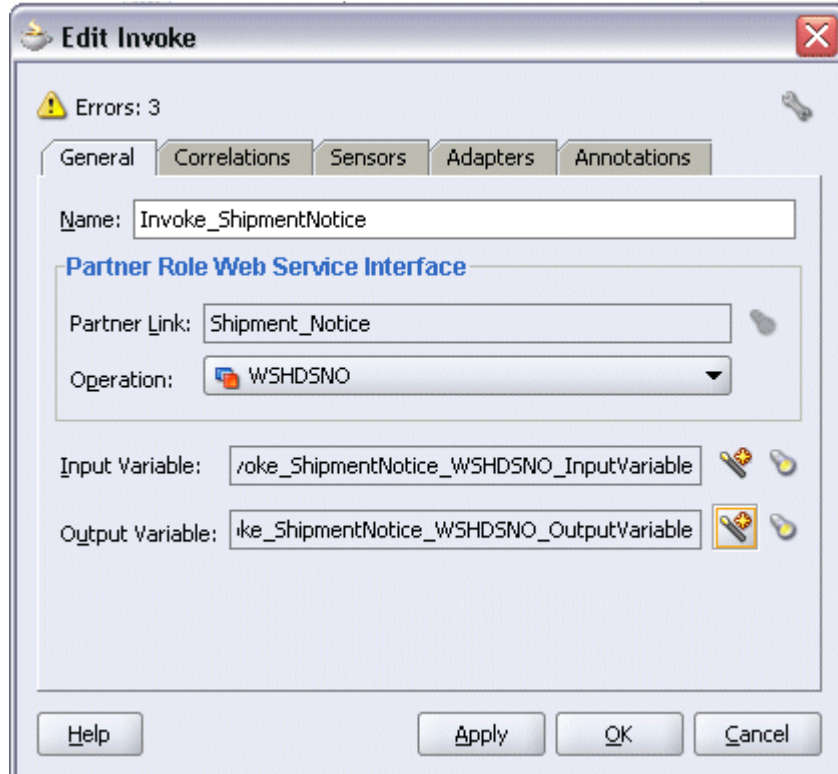
6. Click **Apply** and then **OK** in the Edit Invoke dialog box to finish configuring the Invoke activity.

The Invoke activity appears in the process diagram.

#### **To add the second Invoke activity for Shipment\_Notice Partner Link:**

1. In JDeveloper BPEL Designer, drag and drop the **Invoke** activity from the Component Palette into the Activity box of the process diagram, after the **Invoke** and **Reply** activities.
2. Link the Invoke activity to the `Shipment_Notice` service. The Invoke activity will send event data to the partner link. The Edit Invoke dialog box appears.
3. Enter a name for the Invoke activity such as 'Invoke\_ShipmentNotice'. Select input and output global variables as described in the first Invoke activity creation procedure.

### Editing Invoke Variables



The **Edit Invoke** dialog box is shown with the **General** tab selected. It features a warning icon and the text "Errors: 3" in the top left corner. The dialog has five tabs: **General**, **Correlations**, **Sensors**, **Adapters**, and **Annotations**. The **Name** field is set to "Invoke\_ShipmentNotice". Below this, the **Partner Role Web Service Interface** section contains a **Partner Link** field set to "Shipment\_Notice" and an **Operation** dropdown menu set to "WSHDSNO". The **Input Variable** field is set to "Invoke\_ShipmentNotice\_WSHDSNO\_InputVariable" and the **Output Variable** field is set to "Invoke\_ShipmentNotice\_WSHDSNO\_OutputVariable". Both variable fields have edit and delete icons to their right. At the bottom, there are four buttons: **Help**, **Apply**, **OK**, and **Cancel**.

**Edit Invoke**

Errors: 3

General Correlations Sensors Adapters Annotations

Name: Invoke\_ShipmentNotice

**Partner Role Web Service Interface**

Partner Link: Shipment\_Notice

Operation: WSHDSNO

Input Variable: Invoke\_ShipmentNotice\_WSHDSNO\_InputVariable

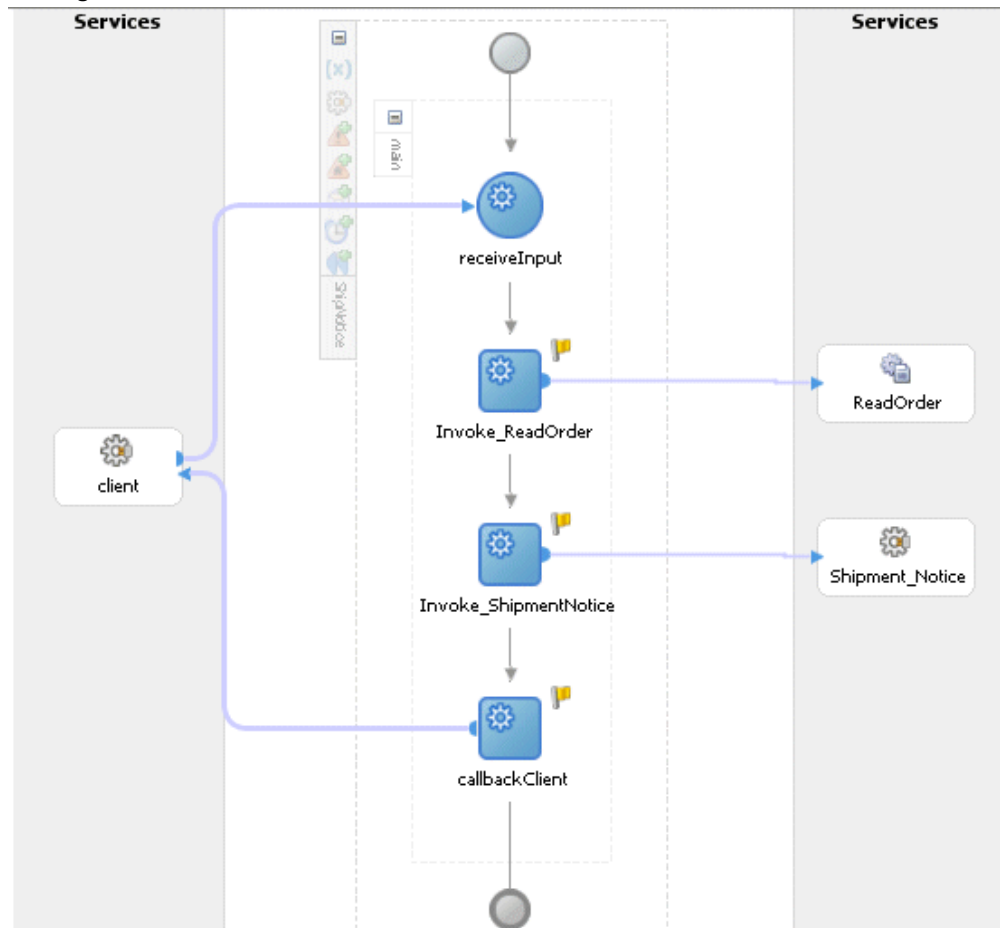
Output Variable: Invoke\_ShipmentNotice\_WSHDSNO\_OutputVariable

Help Apply OK Cancel

Click **Apply** and then **OK** in the Edit Invoke dialog box to finish configuring the Invoke activity.

The second Invoke activity appears in the process diagram.

### Adding Invoke Activities



### Adding Assign Activities

This step is to configure three Assign activities:

1. To pass the application context for SOAHeader in the invocation of the DSNO concurrent program service.
2. To pass the order details from the output of the Synchronous Read - File Adapter service to the input of the DSNO creation through the Invoke\_ShipmentNotice Invoke activity.
3. To set the SOAP response to output.

#### Assigning SOAHeader Parameters:

To add the first Assign activity to pass SOAHeader variables used in the invocation of the DSNO concurrent program service:

1. Add the first Assign activity by dragging and dropping the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the two **Invoke** activities.
2. Enter 'SOAHeader' as the Assign name in the Edit Assign dialog box. Click **OK**.
3. Enter the first pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
  - In the From navigation tree, select type Expression and then enter 'ORDER\_MGMT\_SUPER\_USER' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Invoke\_Shipment\_Notice\_WSHDSNO\_InputVariable > header > ns2:SOAHeader** and select **ns2:Responsibility**. The XPath field should contain your selected entry.
  - Click **OK**.
4. Enter the second pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
  - In the From navigation tree, select type Expression and then enter 'ONT' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Invoke\_Shipment\_Notice\_WSHDSNO\_InputVariable > header > ns2:SOAHeader** and select **ns2:RespApplication**. The XPath field should contain your selected entry.
  - Click **OK**.
5. Enter the third pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
  - In the From navigation tree, select type Expression and then enter 'STANDARD' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Invoke\_Shipment\_Notice\_WSHDSNO\_InputVariable > header > ns2:SOAHeader** and select **ns2:SecurityGroup**. The XPath field should contain your selected entry.
  - Click **OK**.
6. Enter the fourth pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:



- In the From navigation tree, select type Expression and then enter 'AMERICAN' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Invoke\_Shipment\_Notice\_WSHDSNO\_InputVariable > header > ns2:SOAHeader** and select **ns2:NLSLanguage**. The XPath field should contain your selected entry.
  - Click **OK**.
7. Enter the fifth pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
- In the From navigation tree, select type Expression and then enter '202' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Invoke\_Shipment\_Notice\_WSHDSNO\_InputVariable > header > ns2:SOAHeader** and select **ns2:Org\_Id**. The XPath field should contain your selected entry.
  - Click **OK**.
8. The Edit Assign dialog box appears.
- Click **Apply** and then **OK** to complete the configuration of the Assign activity.

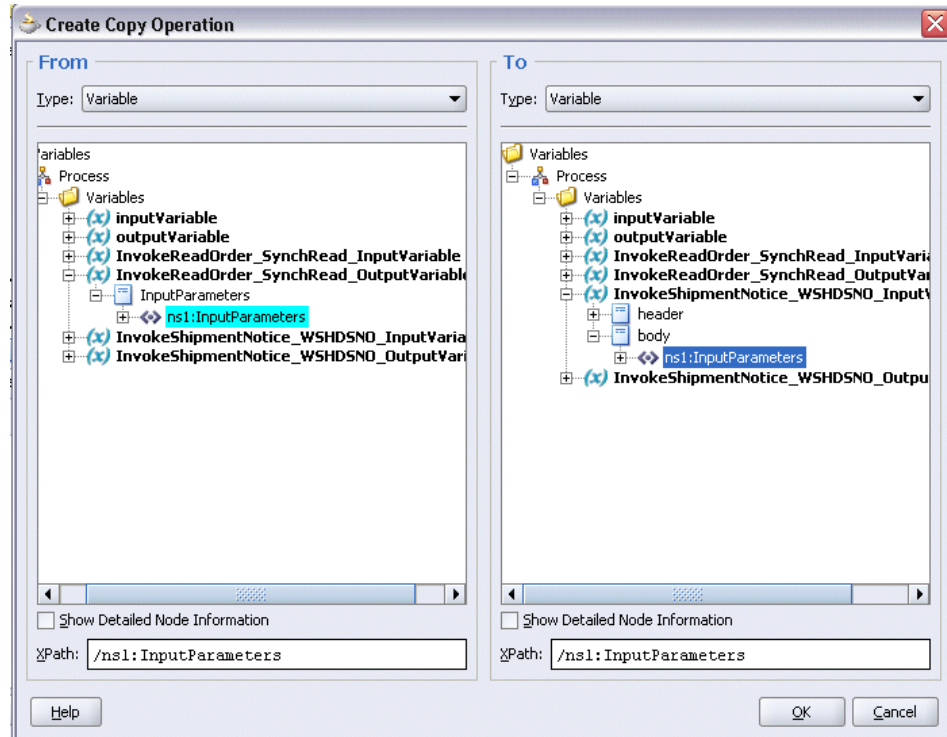
**To add the second Assign activity to set order details to the Invoke\_ShipmentNotice Invoke activity:**

1. In JDeveloper BPEL Designer, drag and drop the **Assign** activity from the Component Palette into the Activity box of the process diagram, between **Assign** and **Invoke** activities.
2. Double-click the **Assign** activity to access the Edit Assign dialog box.
3. Click the General tab to enter the name for the Assign activity, such as 'SetOrderDetails'.
4. On the Copy Operation tab, click **Create** and then select **Copy Operation** from the menu. The Create Copy Operation window appears.
5. Enter the first pair of parameters:
  - In the From navigation tree, select type elect type Variable. Navigate to **Variable > Process > Variables > InvokeReadOrder\_SynchRead\_OutputVariable > InputParameters** and select **ns1:InputParametersr**.

The XPath field should contain your selected entry.

- In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > InvokeShipmentNotice\_WSHDSNO\_InputVariable > body** and select **ns1:InputParameters**. The XPath field should contain your selected entry.
- Click **OK**.

### Assigning Parameters



6. The Edit Assign dialog box appears.

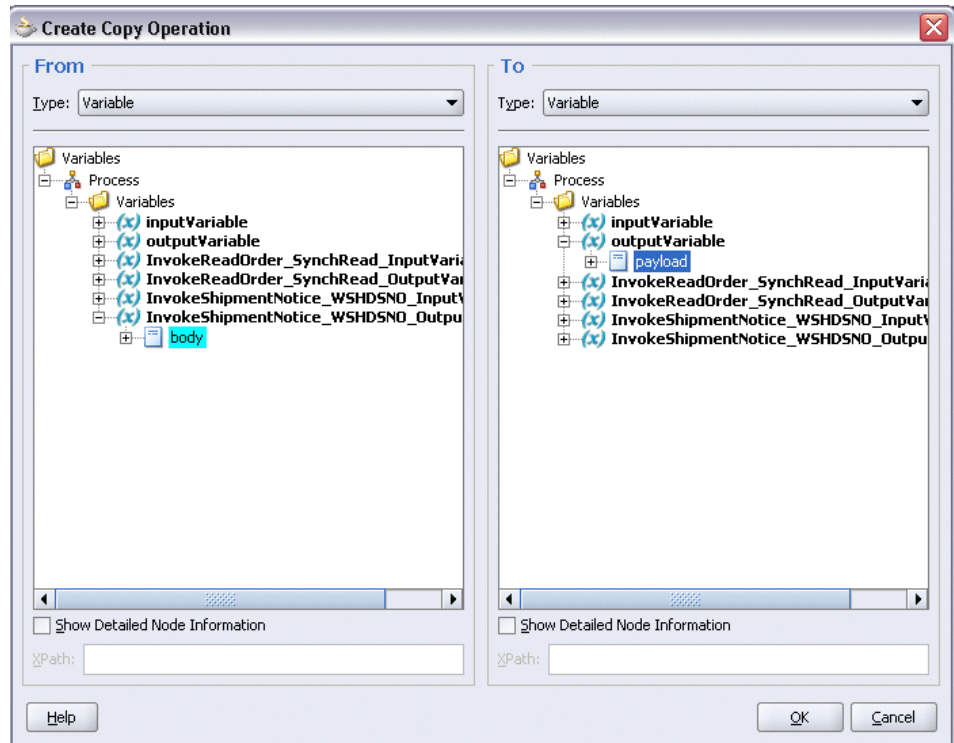
Click **Apply** and then **OK** to complete the configuration of the Assign activity.

### To add the third Assign activity to set SOAP response to output:

1. Add the third Assign activity by dragging and dropping the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the **Invoke** and the **Reply** activities.
2. Repeat Step 2 to Step 4 described in creating the first Assign activity to add the third Assign activity called 'SetCPdetails'.
3. Enter the following information:

- In the From navigation tree, select type Variable. Navigate to **Variable > Process > Variables > InvokeShipmentNotice\_WSHDSNO\_OutputVariable** and select **body**.
- In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > OutputVariable** and select **payload**.

### Assigning Parameters



- Click **OK**.

4. Click **Apply** and then **OK** to complete the configuration of the Assign activity.

## Deploying and Testing the BPEL Process at Run Time

After creating a BPEL process using the WSDL URL generated from the concurrent program interface definition, you can deploy it to a BPEL server if needed. To ensure that this process is modified or orchestrated appropriately, you can also manually test the BPEL process by initiating the business process contained in the BPEL process to test the interface integration.

### Prerequisites

Before deploying the BPEL process using Oracle JDeveloper, you must ensure that you

have established the connectivity between the design-time environment and the run-time servers including the application server and the integration server.

How to configure the necessary server connection, see *Configuring Server Connection*, page B-1.

To validate your BPEL process, perform the following run-time tasks:

1. Deploy the BPEL process, page 6-24

Once you deploy the process to a BPEL server, it becomes available so that you can run the process manually to test it for validation.

2. Test the BPEL process, page 6-25

After deploying a BPEL process, you can manage the process from the BPEL console to validate the interface integration contained in your BPEL process.

## Deploying the BPEL Process

You must deploy the BPEL process (*ShipNotice.bpel*) that you created earlier before you can run it.

**To deploy the BPEL process:**

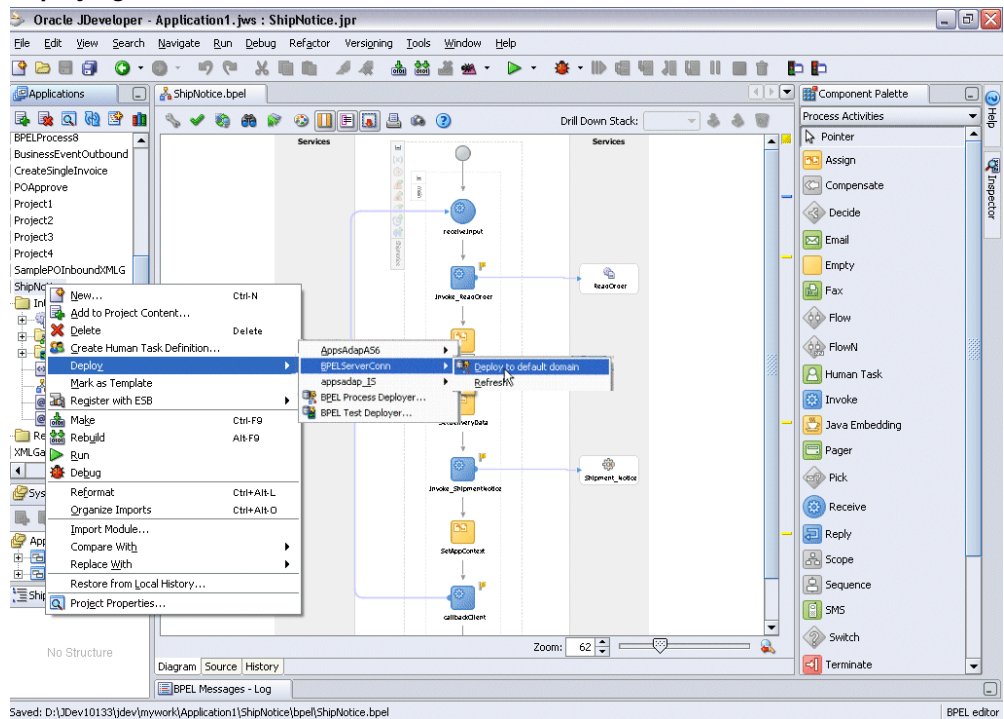
1. In the Applications Navigator of JDeveloper BPEL Designer, select the **ShipNotice** project.
2. Right-click the project and click **Make** action from the menu.

Look for any compilation error messages in Message Log.

Right-click the project and select **Deploy > Integration Server Connection name > Deploy to Default Domain** action from the menu.

For example, you can select **Deploy > BPELServerConn > Deploy to Default Domain** to deploy the process if you have the BPEL Process Manager setup appropriately.

## Deploying the BPEL Process



3. Look for 'Build successful' message in Apache Ant – Log. The BPEL project is compiled and successfully deployed.

## Testing the BPEL Process

To validate whether the BPEL process that you created works or not, you need to manually initiate the process after it has been successfully deployed to the BPEL server. Therefore, the validation starts with the BPEL console to ensure that you can find the deployed BPEL process listed in the console. Then, you can log on to Oracle E-Business Suite to manually initiate the processes and to confirm that the departure shipment notice outbound (DSNO) is generated in the XML file.

### To test the BPEL process:

1. Log into Oracle Application Server 10g BPEL Console (<http://<soaSuiteServerHostName>:<port>/BPELConsole>). The BPEL Console login page appears.
2. Enter the username and password and click **Login**. The Oracle Enterprise Manager 10g BPEL Console appears with a list of deployed BPEL processes.

## Deployed BPEL Processes

Oracle BPEL Console v10.1.2.0.0 - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

Getting Started Latest Headlines

**ORACLE** BPEL Console Manage BPEL Domain | Logout | Support

**Dashboard** **BPEL Processes** **Instances** **Activities**

Deployed BPEL Processes		In-Flight BPEL Process Instances		
Name	Instance	BPEL Process	Last Modified ↑	
InsertPurchaseOrder				
<a href="#">ShipNotice</a>				
TaskActionHandler				
TaskManager				
Recently Completed BPEL Process Instances ( <a href="#">More...</a> )				
	9 : Instance #9 of InsertPurchaseOrder	InsertPurchaseOrder (v. 1.0)	11/3/05 5:16:08 PM	
	8 : Instance #8 of InsertPurchaseOrder	InsertPurchaseOrder (v. 1.0)	11/3/05 12:32:59 PM	
	7 : Instance #7 of InsertPurchaseOrder	InsertPurchaseOrder (v. 1.0)	11/2/05 3:17:24 PM	

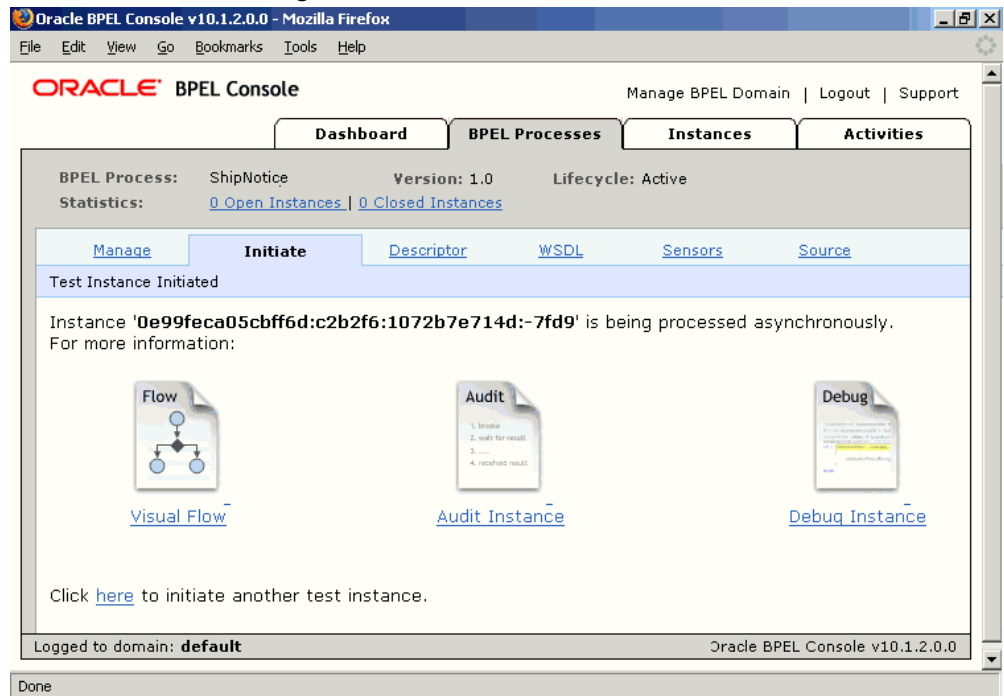
Deploy New Process

Logged to domain: **default** Oracle BPEL Console v10.1.2.0.0

<http://localhost:9700/BPELConsole/default/displayProcess.jsp?processId=ShipNotice&revisionTag=1.0>

3. In the BPEL Console, confirm that `ShipNotice` has been deployed.
4. Click the `ShipNotice` link to open the Initiate tab.
5. Enter the payload input field and 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. Double-click the Invoke\_ShipmentNotice icon from the process flow chart and click **View XML document** link to open the XML file. This file records the Request ID that is returned for the transaction.

### Verifying Records in Oracle E-Business Suite

Before verifying the records in Oracle E-Business Suite, you must first ensure that the concurrent request is completed successfully.

1. Log on to Oracle E-Business Suite with the System Administrator responsibility. Select **View > Requests** to open the Find Requests window.
2. Search for the concurrent request by entering the Request Id that you got from the audit trail and then click **Find**.
3. The request details page is displayed. You can check the Phase and Status of the request to see if the Status of the request is **Complete**.

Once the concurrent request is completed successfully, you can validate it in Oracle E-Business Suite.

Since DSNO (departure shipment notice outbound) is an outbound XML message, relevant XML Gateway setup tasks must be configured appropriately in order for the shipment notice to be delivered to the right recipient.

See *Oracle XML Gateway User's Guide* for details.

You can validate if the ship-to address, purchase order, and requested ship date addressed in the DSNO XML file are the same in your sales order.



---

# Using Business Service Objects

## Overview

A business service object, formerly known as Service Bean, is a high-level service component that allows OA Framework or BC4J components to be deployed as Web services. It is the tool by which Oracle E-Business Suite employs service oriented architecture (SOA) and Web services to facilitate integration with each other and with third party trading partners.

Business service object interfaces provide access to SOA services to facilitate integration between Oracle E-Business Suite and trading partners. They often employ service data objects as parameters to pass complex data.

To better utilize these business service objects for broader customers, integration repository administrators can first generate Web services and then deploy them to the application server. Integration developers can orchestrate those deployed services into a meaning business process with service endpoints using a BPEL language. This process can take the data from a business partner and then insert or update Oracle E-Business Suite if necessary.

To better understand how to use business service object interfaces in updating Oracle E-Business Suite, detailed design-time and run-time tasks are included in this chapter. For the example described in the following sections, we use Oracle JDeveloper 10.1.3.3.0 as a design-time tool to create the BPEL process and use Oracle SOA Suite BPEL server 10.1.3.3.0 for the process deployment.

## Using Business Service Object WSDLs at Design Time

### BPEL Process Scenario

This example uses PurchaseOrderService  
`/oracle/apps/fnd/framework/toolbox/tutorial/PurchaseOrderService`  
business service object interface to explain the BPEL process creation.

When a purchase order approval request is received, the purchase order details is read

by a File Adapter. The order data is then passed to the `approvePurchaseOrder` method within the `PurchaseOrderService` to initiate the single PO approval process. The approval information is then replied back to the requestor.

If the BPEL process is successfully executed after deployment, you should notice the purchase order status changed from `Incomplete` to `Approved`.

### **Prerequisites to Create a BPEL Process Using a Business Service Object Web Service**

- An integration repository administrator needs to successfully generate and deploy a Web service to the application server.
- An integration developer needs to locate and record the deployed WSDL URL for the BSO interface exposed as a Web service.
- Header variables need to be populated for Web service authorization.

### ***Generating and Deploying WSDL URL***

An integration repository administrator must first create a Web service for a selected interface definition, and then deploy the service from Oracle Integration Repository to the application server.

For example, the administrator must perform the following steps before letting the integration developers use the deployed WSDL in creating a BPEL process:

1. To generate a Web service, locate the business service object interface definition first from the Oracle Integration Repository (such as `PurchaseOrderService` /`oracle/apps/fnd/framework/toolbox/tutorial/PurchaseOrderService`) and click **Generate WSDL** in the interface details page.

Since business service object interfaces are service enabled by using Web Service Provider, once the service is successfully generated, the Web Service - Web Service Provider region appears in the interface details page. For detailed instruction on how to generate a Web service, see *Generating Web Services, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

2. To deploy a generated Web service, select at least one authentication type and click **Deploy** in the Web Service - Web Service Provider region of the interface details page to deploy the service. Once the service is successfully deployed, the selected authentication type(s) will be displayed along with 'Deployed' Web Service Status. For more information on securing Web services with authentication types, see *Managing Web Service Security, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

For detailed instruction on how to deploy a Web service, see *Deploying Web Services, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

### ***Searching and Recording WSDL URL***

Apart from the required tasks performed by the administrator, an integration developer also needs to log on to the system to locate and record the deployed Web service WSDL

URL for the interface that needs to be orchestrated into a meaningful business process in Oracle JDeveloper using BPEL language.

This WSDL information will be used later in creating a partner link for the interface exposed as a Web service during the BPEL process creation at design time.

**Confirming and Recording a Deployed WSDL URL**

ORACLE<sup>®</sup> Integration Repository

Home Logout Preferences Help Diagnostics

Integration Repository

View By: Interface Type

Business Event

Business Service Object

Applications Technology

Application Object Library

Discrete Manufacturing

Financial Payables Suite

Financial Receivables Suite

Financials

Order Management Suite

Product Lifecycle Management

Public Sector

Service Suite

Supply Chain Management

Concurrent Program

EDI

Interface View

Java

Open Interface

PL/SQL

Service Data Object

XML Gateway Map

Integration Repository >

Business Service Object : Purchase Order Service

Search Printable Page

Qualified Name

/oracle/apps/fnd/framework/svctoolbox/tutorial/PurchaseOrderService

Status

Active

Interface

oracle.apps.fnd.framework.svctoolbox.tutorial.PurchaseOrderService

Scope

Public

Extends

oracle.svc.dataProcessorService

Interface Source

Oracle

Product

Application Object Library

XML Schema

PurchaseOrderService

Approves a list of purchase orders.

Full Description

Approves a list of purchase orders.

Web Service - Web Service Provider

Web Service Status

Deployed

WSDL

View WSDL

\* Authentication Type

☒ Username Token

☒ SAML Token (Sender Vouches)

Source Information

Source File

java/framework/svctoolbox/tutorial/server/PurchaseOrderSAM.xml

Source Version

120.23.12010000.4

Source Product

FND

Implementation

oracle.apps.fnd.framework.svctoolbox.tutorial.server.PurchaseOrderSAM

Methods

Internal Name	Status	Description
getItemPrice	Active	Returns the price of a specific item.
getPurchaseOrder	Active	Gets a PurchaseOrder based on its primary key attributes.
mergePurchaseOrder	Active	Updates or creates a PurchaseOrder.
processDataSource	Active	Processes a root data object containing a list of data objects applied to a data source.
queryDataSource	Active	Gets root data object containing a list of data objects from the data source based on criteria.
receiveItem	Active	Receives a single purchaser order.
receiveItems	Active	Receives a list of items.
updatePurchaseOrder	Active	Updates a PurchaseOrder.
deletePurchaseOrder	Active	Deletes a PurchaseOrder.
acknowledgePurchaseOrder	Active	Acknowledges a purchase order.
approvePurchaseOrder	Active	approves a single purchase order.
approvePurchaseOrders	Active	Approves a list of purchase orders.
createPurchaseOrder	Active	Creates a new PurchaseOrder.

How to search for an interface and review the interface details, see Searching and Viewing Integration Interfaces, page 2-1.

**Setting Header Variables for SOAP Request**

You need to populate certain variables in the BPEL process for header elements to pass values that may be required to set application context during service execution. These header elements for Business Service Object interface type are *RESPONSIBILITY\_NAME*, *RESPONSIBILITY\_APPL\_NAME*, *SECURITY\_GROUP\_NAME*, *NLS\_LANGUAGE*, and *ORG\_ID*.

**Note:** The user information is defined by the *wsseUsername* property passed within the security headers. Detailed instructions on how to pass the security headers along with the SOAP request, see *Passing Values to Security Headers*, page 7-8.

The expected values for these elements are described in the following table:

### **Header Variables and Expected Values for Business Service Object Interfaces**

Element Name	Expected Value
RESPONSIBILITY_NAME	responsibility_name (such as "System Administrator") or {key}responsibility_key (such as "{key}SYSTEM_ADMINISTRATOR")
RESPONSIBILITY_APPL_NAME	Application Short Name (such as "FND")
SECURITY_GROUP_NAME	Security Group Key (such as "STANDARD")
NLS_LANGUAGE	NLS Language (such as "AMERICAN")
ORG_ID	Org ID (such as "202")

**Note:** NLS\_Language and ORG\_ID are optional values to be passed.

- If the NLS Language element is specified, SOAP requests can be consumed in the language passed. All corresponding SOAP responses and error messages can also be returned in the same language. If no language is identified, then the default language of the user will be used.
- If a service execution is dependent on any particular organization, then you must pass the ORG\_ID element in the ServiceBean\_Header of that SOAP request.

The context information can be specified by configuring an Assign activity before the Invoke activity in the BPEL PM.

Detailed information on how to set header variables for the SOAP request, see Assigning ServiceBean\_Header Parameters, page 7-19.

#### **BPEL Process Creation Flow**

Based on the scenario, the following design-time tasks are discussed in this chapter:

1. Create a new BPEL project, page 7-5

Use this step to create a new BPEL project called `ApprovePO.bpel`. This automatically creates two dummy activities - Receive and Reply - to receive input from a third party application and to reply output of the BPEL process back to the request application.

2. Create a Partner Link, page 7-7

Use this step to create a partner link for PurchaseOrderService Web services.

3. Add a Partner Link for File Adapter, page 7-9

This is to synchronous read purchase order details received from the requestor.

4. Add Invoke activities, page 7-15

Use this step to create two Invoke activities in order to:

1. Point to the File Adapter - Synchronous Read operation to read the purchase order from the input file.
2. Point to the PurchaseOrderService Web service to initiate the single purchase order approval process.

5. Add Assign activities, page 7-18

Use this step to create three Assign activities in order to:

1. To set the SOAHeader details.
2. Pass the purchase order details read from the File Adapter as an input to the Invoke activity for PurchaseOrderService Web service.
3. Pass single purchase order approval information to the requestor through the Reply activity.

For general information and basic concept of a BPEL process, see Understanding BPEL Business Processes, page D-1 and *Oracle BPEL Process Manager Developer's Guide* for details.

## Creating a New BPEL Project

Use this step to create a new BPEL project that will contain various BPEL process activities.

**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 produces a list of available categories.
4. Expand the **General** node and then select **Projects**.
5. Select **BPEL Process Project** from the **Items** group.

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

#### **Entering BPEL Project Information**

**BPEL Project Creation Wizard - Project Settings**

The BPEL Project Creation Wizard allows you to create a project in which you can design a business process based on the BPEL (Business Process Execution Language) standard.

Please specify the process name and project settings below.

Name:

Namespace:

☒ **Use Default Project Settings**

Project Name:

Project Directory:

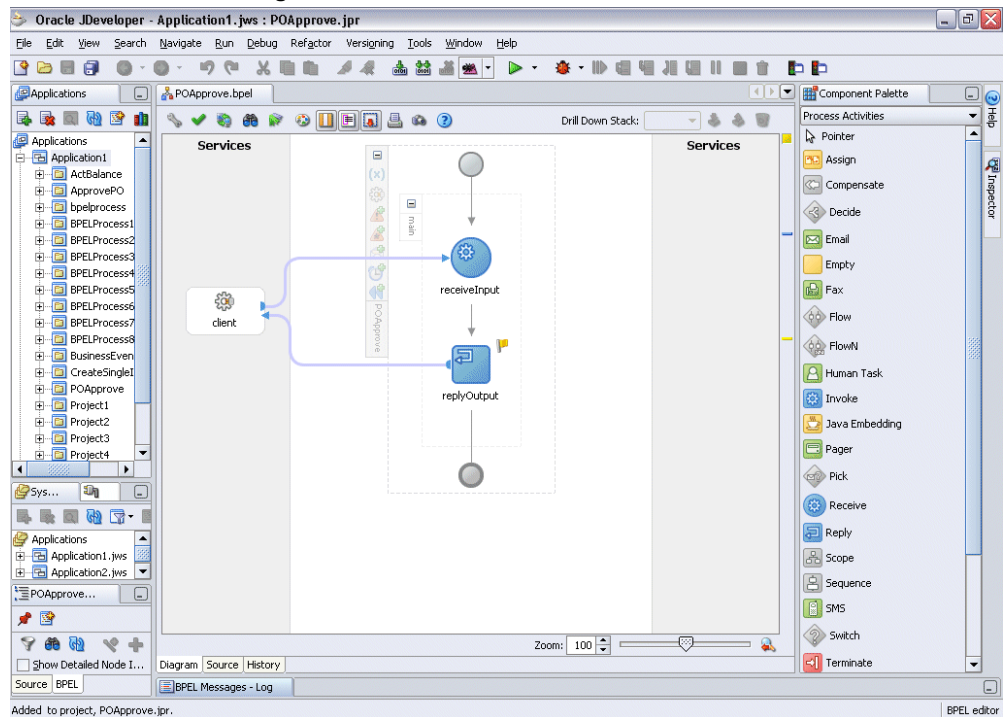
Template:

Help < Back Next > Finish Cancel

7. In the **Name** field, enter a descriptive name for example `ApprovePO`.
8. From the Template list, select **Synchronous BPEL Process** and then select **Use Default Project Settings**.
9. Use the default input and output schema elements in the Input/Output Elements dialog box.
10. Click **Finish**.

A new synchronous BPEL process is created with the Receive and Reply activities. The required source files including `bpel.xml`, using the name you specified (for example, `ApprovePO.bpel`) are also generated.

## New BPEL Process Diagram



## Creating a Partner Link

Use this step to configure a Partner Link called PurchaseOrderService.

**To create a partner link for PurchaseOrderService:**

1. In JDeveloper BPEL Designer, drag and drop the **PartnerLink** service from the Component Palette into the Partner Link border area of the process diagram. The Service Name dialog box appears.
2. Copy the WSDL URL corresponding to the PurchaseOrderService /oracle/apps/fnd/framework/toolbox/tutorial/PurchaseOrderService that you recorded earlier in the WSDL File field.

A Partner Link Type message dialog box appears asking whether you want the system to create a new WSDL file that will by default create partner link types for you.

Click **Yes** to have the PurchaseOrderService partner link created in the process diagram.

The Partner Name and Partner Link Type values populated automatically. You can select the Partner Role value from the drop-down list.

Click **Apply**.

### Editing the Partner Link

**Edit Partner Link**

Errors: 1

General Image Property

Name: PurchaseOrderService

Process: ApprovePO

**WSDL Settings**

WSDL File: ation1/ApprovePO/bpel/PurchaseOrderService1.wsdl

Partner Link Type: PurchaseOrderService\_PortType\_PL

Partner Role: PurchaseOrderService\_PortType\_Role

My Role: ---- Not Specified ----

Help Apply OK Cancel

### 3. Passing Values to Security Headers

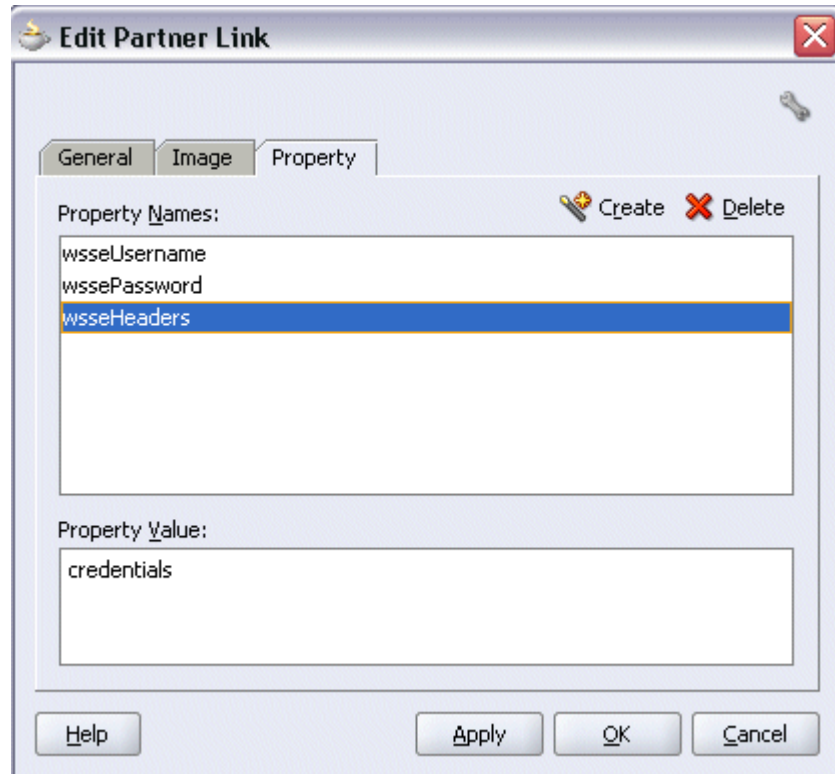
Select the Property tab and click the **Create Property** icon to select the following properties from the property name drop-down list in order to pass the security headers along with the SOAP request:

- wsseUsername  
Specify the username to be passed in the Property Value box.
- wssePassword  
Specify the corresponding password for the username to be passed in the Property Value box.
- wsseHeaders  
Enter `credentials` as the property value.

Click **Apply** to save the selected property values.



### Adding Properties



4. 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.

## Adding a Partner Link for File Adapter

Use this step to synchronously read the purchase order details received from the third party application.

### To add a Partner Link for 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 welcome page appears.
2. Click **Next**. The Service Name dialog box appears.
3. Enter a name for the file adapter service for example ReadPO. You can add an optional description of the service.
4. Click **Next** and the Operation dialog box appears.

### Specifying the Operation

The File Adapter supports three operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, and a Synchronous Read File operation that reads the current contents of a file. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type: ☐ Read File ☐ Write File ☒ Synchronous Read File

Operation Name:

Help < Back Next > Finish Cancel

5. Specify the operation type, for example **Synchronous Read File**. This automatically populates the **Operation Name** field.

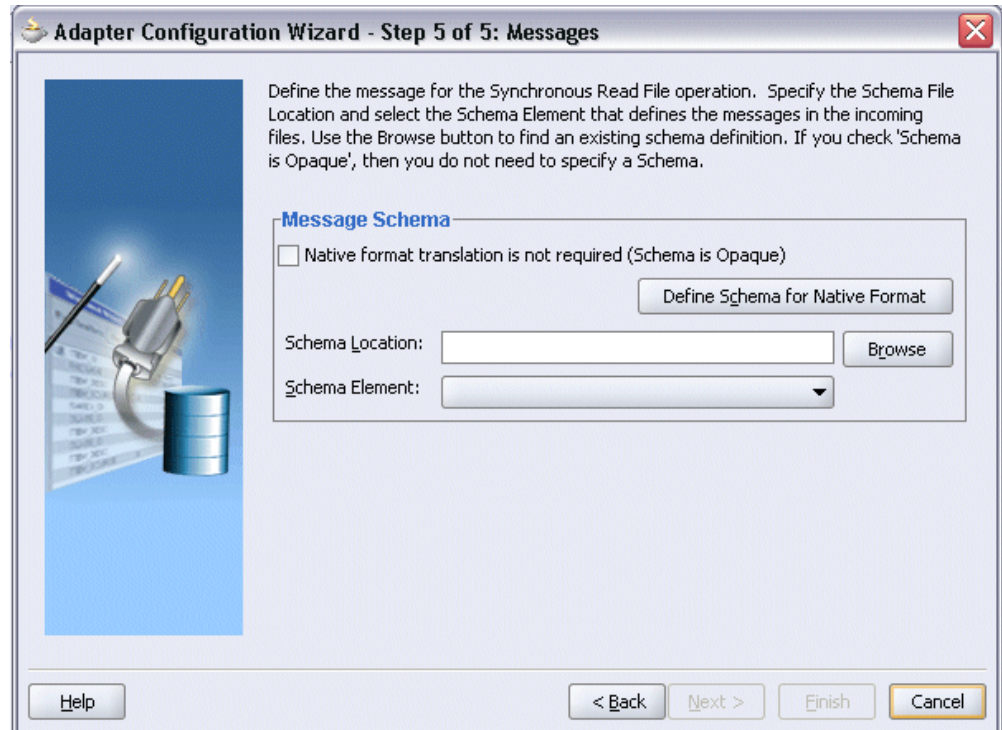
Click **Next** to access the File Directories dialog box.

6. Select **Physical Path** radio button and enter the input payload file directory information. For example, enter `/usr/tmp/` as the directory name.

Uncheck the **Delete Files after successful retrieval** check box. Click **Next** to open the File Name dialog box.

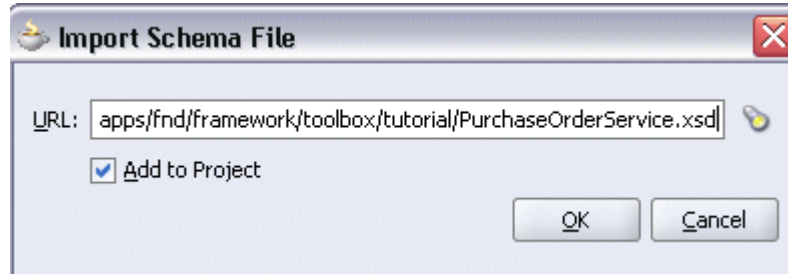
7. Enter the name of the file for the synchronous read file operation. For example, enter 'Input.xml'. Click **Next**. The Messages dialog box appears.

### Specifying Message Schema



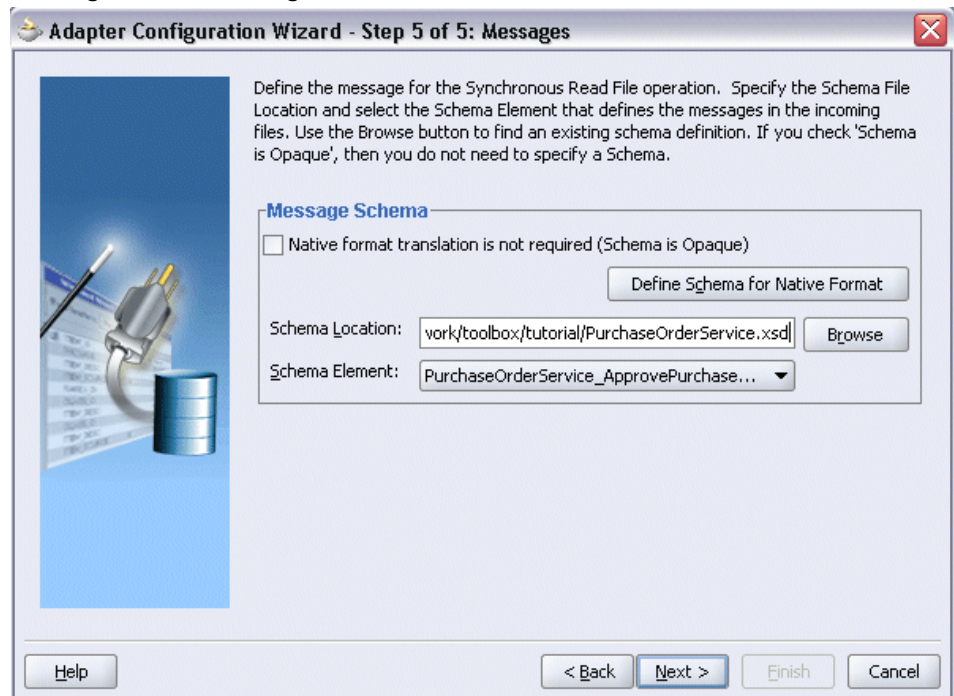
8. Select **Browse** for schema file in Schema Location. The Type Chooser window is displayed.
  1. Click **Import Schema Files** button on the top right corner of the Type Chooser window.
  2. Enter the schema location for the service. Such as  
`http://<myhost>:<port>/webservices/AppsWSProvider/oracle/apps/fnd/framework/toolbox/tutorial/PurchaseOrderService.xsd`.  
  
Schema location for your service can be found from the service WSDL URL (for example,  
`http://<myhost>:<port>/AppsWSProvider/oracle/apps/fnd/framework/toolbox/tutorial/PurchaseOrderService?wsdl`).

### Importing Schema Location



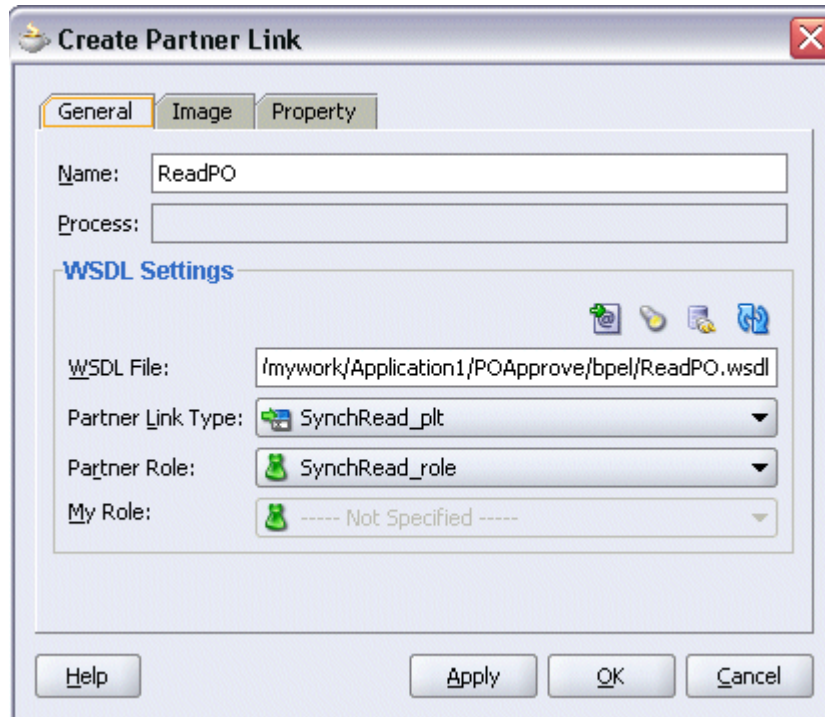
3. Select the **Add to Project** check box and click **OK**.
4. Click **OK** for Import schema prompt.  
The Imported Schemas folder is automatically added to the Type Chooser window.
5. Select schema element by expanding the Imported Schemas folder > PurchaseOrderService.xsd > PurchaseOrderService\_ApprovePurchaseOrder.  
Click **OK**.  
The selected schema location and element values are displayed.

### Viewing Selected Message Schema and Element



9. Click **Next** and 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 `ReadPO.wsdl`.

### Completing the Partner Link Configuration



The image shows a 'Create Partner Link' dialog box with three tabs: 'General', 'Image', and 'Property'. The 'General' tab is active. It contains the following fields and controls:

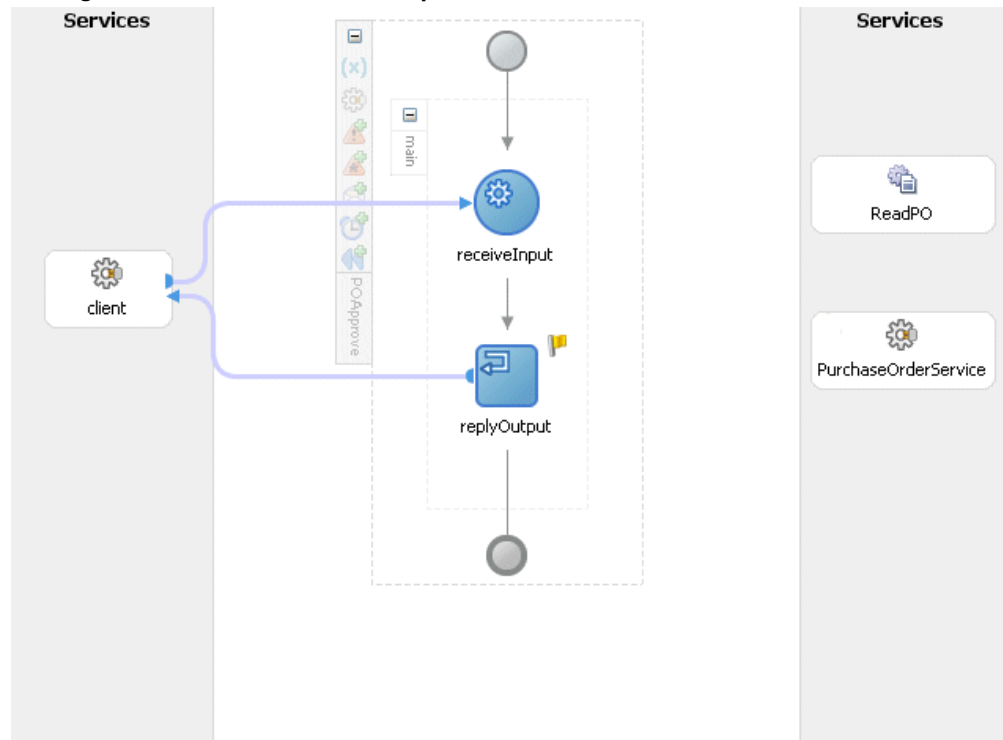
- Name:** ReadPO
- Process:** (empty text box)
- WSDL Settings:**
  - WSDL File:** /mywork/Application1/POApprove/bpel/ReadPO.wsdl
  - Partner Link Type:** SynchRead\_plt
  - Partner Role:** SynchRead\_role
  - My Role:** ---- Not Specified ----

At the bottom of the dialog are four buttons: 'Help', 'Apply', 'OK', and 'Cancel'.

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

The ReadPO Partner Link appears in the BPEL process diagram.

### Adding the Partner Link for File Adapter



### Adding an Invoke activity

This step is to configure two Invoke activities:

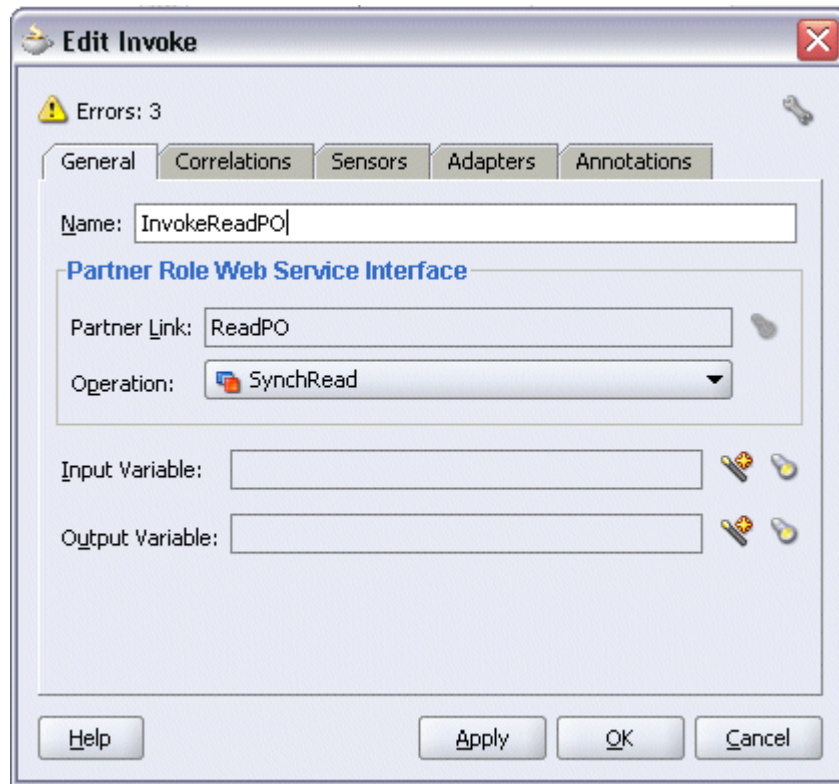
1. Point to the File Adapter ReadPO to synchronous read the purchase order from the Receive activity.
2. Point to the PurchaseOrderService partner link to send the transaction information that is received from the Assign activities to initiate the single purchase order approval process.

#### To add an Invoke activity for ReadPO Partner Link:

1. In JDeveloper BPEL Designer, drag and drop the **Invoke** activity from the Component Palette into the Activity box of the process diagram, between the **Receive** and **Reply** activities.
2. Link the Invoke activity to the ReadPO service. The Invoke activity will send event data to the partner link. The Edit Invoke dialog box appears.

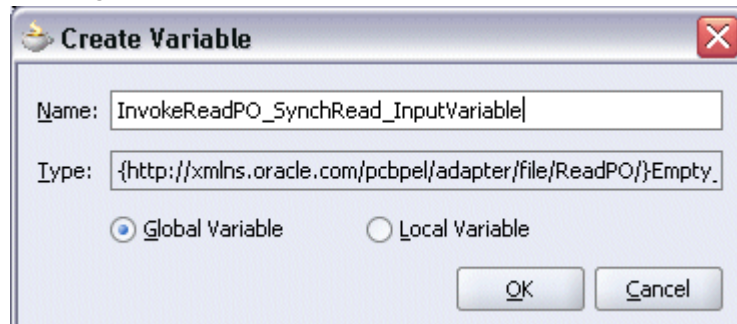


### Editing the Invoke Activity



3. Enter a name for the Invoke activity, and 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** and then enter a name for the variable. You can also accept the default name. Click **OK**.
5. Enter a name for the Invoke activity and then click the **Create** icon next to the

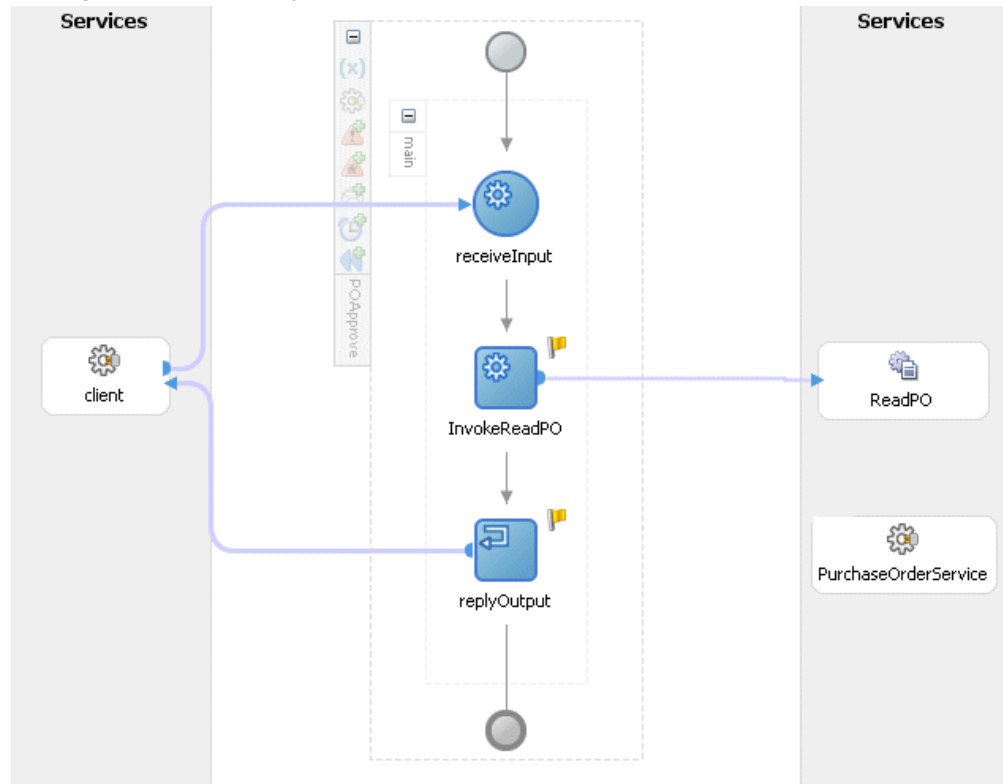


**Output Variable** field to create a new variable. The Create Variable dialog box appears.

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

The Invoke activity appears in the process diagram.

#### **Adding an Invoke Activity**



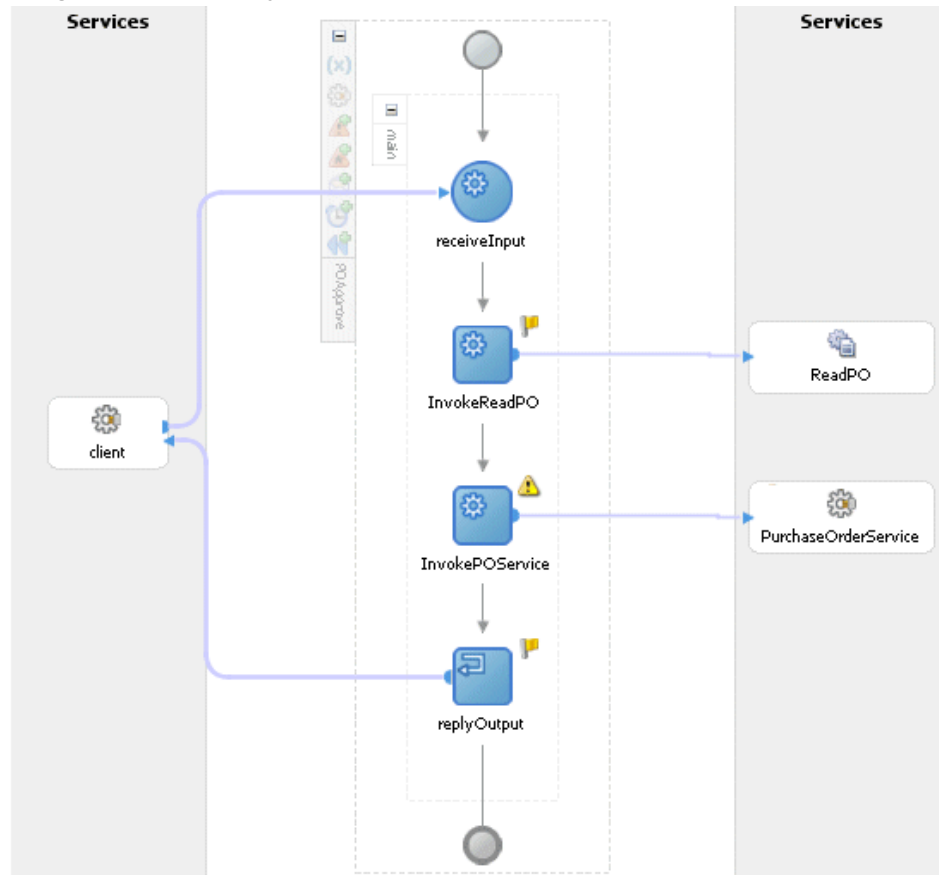
#### **To add an Invoke activity for PurchaseOrderService Partner Link:**

1. In JDeveloper BPEL Designer, drag and drop the **Invoke** activity from the Component Palette into the Activity box of the process diagram, after the **Invoke** and **Reply** activities.
2. Link the Invoke activity to the `PurchaseOrderService` service. The Invoke activity will send event data to the partner link. The Edit Invoke dialog box appears.
3. Enter a name for the Invoke activity such as 'InvokePOService'. Create input and

output variables described in the first Invoke activity. Click **OK** to close the Create Variable dialog box.

4. Click **Apply** and then **OK** to finish configuring the Invoke activity.  
The Invoke activity appears in the process diagram.

#### **Adding an Invoke Activity**



#### **Adding an Assign activity**

This step is to configure three Assign activities:

1. To set the header details.

**Note:** You need to populate certain variables in the BPEL process for ServiceBean\_Header elements to pass values that may be required to embed application context into SOAP envelopes for Web service authorization. These ServiceBean\_Header elements for Business Service Object interface type are

*RESPONSIBILITY\_NAME, RESPONSIBILITY\_APPL\_NAME,  
SECURITY\_GROUP\_NAME, NLS\_LANGUAGE, and ORG\_ID.*

Detailed information on how to set ServiceBean\_Header for the SOAP request, see Assigning ServiceBean\_Header Parameters, page 7-19.

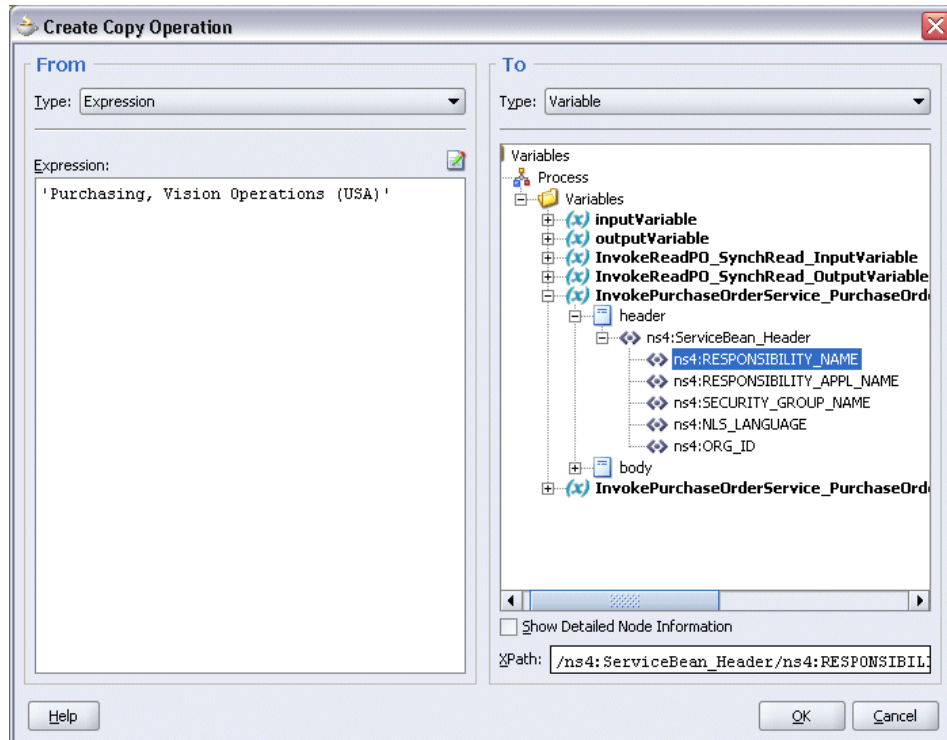
2. To pass the purchase order details read from the File Adapter as an input to the second Invoke activity for PurchaseOrderService partner link.
3. To pass single purchase order approval information to the requestor through the dummy Reply activity.

**To add the first Assign activity to pass header details:**

*Assigning ServiceBean\_Header Parameters:*

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 two **Invoke** activities.
2. Double-click the **Assign** activity to access the Edit Assign dialog box.
3. Click the General tab to enter the name for the Assign activity, such as 'SetServiceBeanHeader'.
4. On the Copy Operation tab, click **Create** and then select **Copy Operation** from the menu. The Create Copy Operation window appears.
5. Enter the first pair of parameters:
  - In the From navigation tree, select type Expression and then enter 'Purchasing, Vision Operations (USA)' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > InvokePurchaseOrderService\_PurchaseOrderService\_AcknowledgePurchaseOrder\_InputVariable > header > ns4:ServiceBean\_Header** and select **ns4:RESPONSIBILITY\_NAME**. The XPath field should contain your selected entry.

### Assign Responsibility Parameter



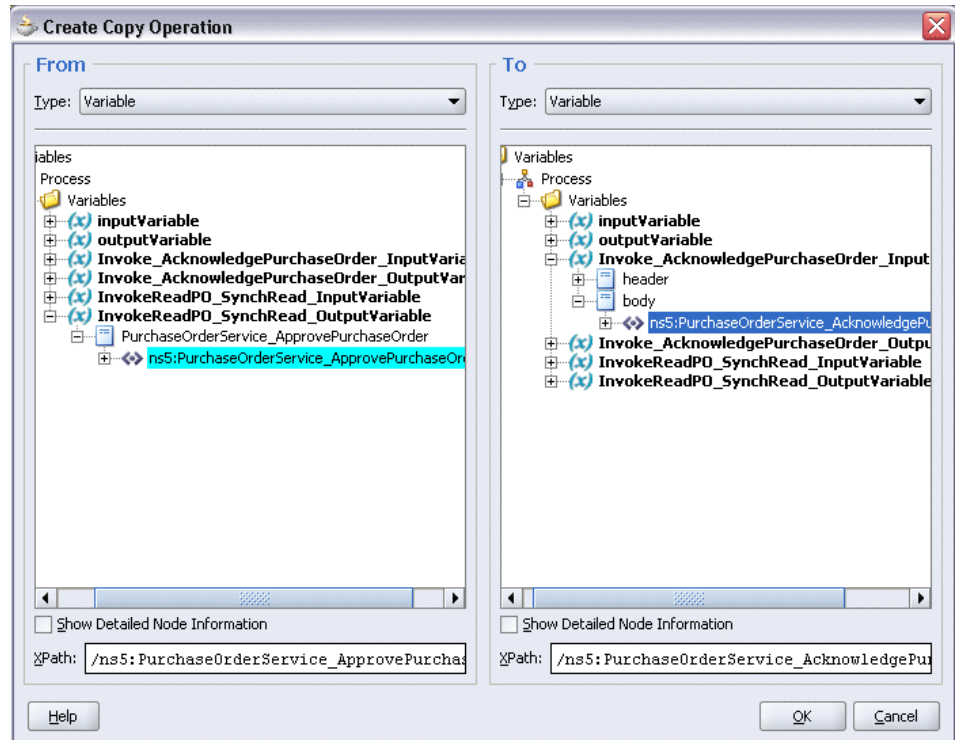
- Click **OK**.
6. Enter the second pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
- In the From navigation tree, select type Expression and then enter 'PUR' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > InvokePurchaseOrderService\_PurchaseOrderService\_AcknowledgePurchaseOrder\_InputVariable > header > ns4:ServiceBean\_Header** and select **ns4:RESPONSIBILITY\_APPL\_NAME**. The XPath field should contain your selected entry.
  - Click **OK**.
7. Enter the third pair of parameters:
- In the From navigation tree, select type Expression and then enter 'STANDARD' in the Expression box.

- In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > InvokePurchaseOrderService\_PurchaseOrderService\_AcknowledgePurchaseOrder\_InputVariable > header> ns4:ServiceBean\_Header** and select **ns4:SECURITY\_GROUP\_NAME**. The XPath field should contain your selected entry.
  - Click **OK**.
8. Enter the fourth pair of parameters:
- In the From navigation tree, select type Expression and then enter 'AMERICAN' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > InvokePurchaseOrderService\_PurchaseOrderService\_AcknowledgePurchaseOrder\_InputVariable > header> ns4:ServiceBean\_Header** and select **ns4:NLS\_LANGUAGE**. The XPath field should contain your selected entry.
  - Click **OK**.
9. Enter the fifth pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
- In the From navigation tree, select type Expression and then enter '204' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > InvokePurchaseOrderService\_PurchaseOrderService\_AcknowledgePurchaseOrder\_InputVariable > header> ns4:ServiceBean\_Header** and select **ns4:ORG\_ID**. The XPath field should contain your selected entry.
  - Click **OK**.
10. The Edit Assign dialog box appears.
11. Click **Apply** and then **OK** to complete the configuration of the Assign activity.

**To enter the second Assign activity to pass PO details to the InvokePOService Invoke 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 **Assign** and **Invoke** activities.

2. Double-click the **Assign** activity to access the Edit Assign dialog box.
3. Click the General tab to enter the name for the Assign activity, such as 'SetPOApproval'.
4. On the Copy Operation tab, click **Create** and then select **Copy Operation** from the menu. The Create Copy Operation window appears.
5. Enter the following information:
  - In the From navigation tree, navigate to **Variable > Process > Variables > InvokeReadPO\_SynchRead\_OutputVariable > PurchaseOrderService\_ApprovePurchaseOrder** > and select **ns5:PurchaseOrderService\_ApprovePurchaseOrder**. The XPath field should contain your selected entry.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > InvokePurchaseOrderService\_PurchaseOrderService\_AcknowledgePurchaseOrder\_InputVariable > body** and select **ns5:PurchaseOrderService\_AcknowledgePurchaseOrder**. The XPath field should contain your selected entry.



- Click **OK**.
6. Click **Apply** and then **OK** to complete the configuration of the Assign activity.
- To enter the third Assign activity to set the SOAP response to output:**
1. Add the second Assign activity by dragging and dropping the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the **InvokePOService Invoke** and the **Reply** activities.
  2. Repeat Step 2 to Step 4 described in creating the first Assign activity to add the second Assign activity called 'SetPOStatus'.
  3. Enter the following information:
    - In the From navigation tree, select type Variable. Navigate to **Variable > Process > Variables > InvokePurchaseOrderService\_PurchaseOrderService\_AcknowledgePurchaseOrder\_OutputVariable** and select **body**.
    - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > outputVariable** and select **payload**.
    - Click **OK**.

The Edit Assign dialog box appears.

4. Click **Apply** and then **OK** to complete the configuration of the Assign activity.

## Deploying and Testing the BPEL Process at Run Time

After creating a BPEL process using the WSDL URL generated from the business service object interface definition, you can deploy it to a BPEL server if needed. To ensure that this process is modified or orchestrated appropriately, you can also manually test the BPEL process by initiating the business process contained in the BPEL process to test the interface integration.

### Prerequisites

Before deploying the BPEL process using Oracle JDeveloper, you must ensure that you have established the connectivity between the design-time environment and the run-time servers including the application server and the integration server.

How to configure the necessary server connection, see *Configuring Server Connection*, page B-1.

To validate your BPEL process, perform the following run-time tasks:

1. Deploy the BPEL process, page 7-24

Once you deploy the process to a BPEL server, it becomes available so that you can run the process manually to test it for validation.

2. Test the BPEL process, page 7-25

After deploying a BPEL process, you can manage the process from the BPEL console to manually initiate the business process and test the interface integration contained in your BPEL process.

## Deploying the BPEL Process

You must deploy the Approve Purchase Order BPEL process (*POApprove.bpel*) that you created earlier before you can run it.

### To deploy the BPEL process:

1. In the Applications Navigator of JDeveloper BPEL Designer, select the **POApprove** project.
2. Right-click the project and click **Make** action from the menu.

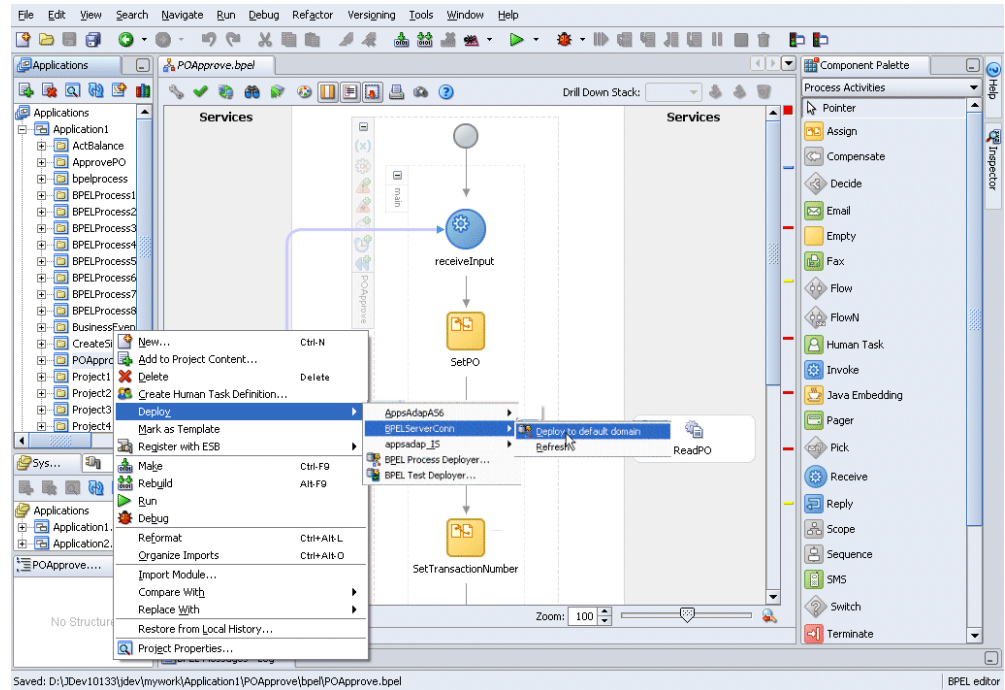
Look for any compilation error messages in Message Log.

Right-click the project and select **Deploy >Integration Server Connection name > Deploy to Default Domain** action from the menu.



For example, you can select **Deploy > BPELServerConn > Deploy to Default Domain** to deploy the process if you have the BPEL Process Manager setup appropriately.

### Deploying the BPEL Process



3. Look for 'Build successful' message in Apache Ant – Log to ensure that the BPEL project is compiled and successfully deployed.

## 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 logging on to Oracle E-Business Suite to manually initiate the processes.

### To test the BPEL process:

1. Log into Oracle Application Server 10g BPEL Console (<http://<soaSuiteServerHostName>:<port>/BPELConsole>). The BPEL Console login page appears.
2. Enter the username and password and click **Login**.  
The Oracle Enterprise Manager 10g BPEL Control appears.
3. In the BPEL Console, confirm that 'POApprove' has been deployed.

4. Click the ApprovePO link to open the Initiate tab
5. Enter the payload input field and click **Post XML Message** to initiate the process.
6. The audit trail provides information about the steps that have been executed. You can check the audit trail by clicking the **Audit Instance** icon.

This is to verify that a purchase order is approved successfully.

#### **Validating the Process in Oracle E-Business Suite**

Additionally, you can validate the BPEL process in Oracle E-Business Suite. Log on to Oracle E-Business Suite with the Purchasing responsibility. Open up the Purchase Orders form and search for the supplier to bring up the purchase order details. Notice that the Status field is 'Approved'.

---

## Using Java APIs for Forms Services

### Overview

Java APIs are business interfaces based on Java classes. Some specialized Java classes are XML document-based integration points wrapped in Java classes for executing business logic in Oracle Forms. These Java classes are categorized with subtype 'Java APIs for Forms'. You can locate them through performing a search with 'subtype of an interface' category and 'Java APIs for Forms' as category value.

Similar to other service enabled integration interface types, Oracle E-Business Suite Integrated SOA Gateway allows these specialized Java classes to be service enabled through SOA Provider.

Once this XML document-based integration point becomes a Web service, the generated Web service can be deployed from the Oracle Integration Repository to Oracle Application Server. Services can then be exposed to customers through service provider and invoked through any of the Web service clients or orchestration tool including Oracle JDeveloper, Apache Axis, .NET Web Service Client, Oracle BPEL Process Manager, and Oracle Enterprise Service Bus (ESB).

To better understand how each individual Web service represented in WSDL URL can be used in inserting or updating application data, detailed design-time and run-time tasks in creating and deploying a BPEL process are discussed in this chapter. For the example described in the following sections, we use Oracle JDeveloper 10.1.3.3.0 as a design-time tool to create the BPEL process and use Oracle SOA Suite BPEL server 10.1.3.3.0 for the process deployment.

### Using Java APIs for Forms Services at Design Time

#### BPEL Process Scenario

This example uses CreateAndMaintainSalesOrders  
`oracle.apps.ont.services.oexoeord.OEXOEORDServices_DocStyle` Java  
API for Forms interface to explain the BPEL process creation.

When a create order request is received, the order information including header and line items will be read and passed to create a sales order. Once the order is created, the order number will then be returned to the requestor.

If the BPEL process is successfully executed after deployment, you should find a sales order is created in the Oracle E-Business Suite. The order number should be the same as the payload input value.

### **Prerequisites to Create a BPEL Process Using a Java API for Forms Web Service**

Before performing the design-time tasks for the Web service, you need to ensure the following tasks are in place:

- An integration repository administrator needs to successfully generate and deploy a Web service to the application server.
- An integration developer needs to locate and record the deployed WSDL URL for the document-based Java interface exposed as a Web service.
- SOAHeader variables need to be populated for Web service authorization.

### ***Deploying a Java API for Forms WSDL URL***

An integration repository administrators must first create a Web service for a selected document-based Java interface definition, and then deploy the service from Oracle Integration Repository to the application server.

For example, the administrator must perform the following steps before letting the integration developers use the deployed WSDL in creating a BPEL process:

1. To generate a Web service, locate the Java API for Forms interface definition first (such as `CreateAndMaintainSalesOrders` `oracle.apps.ont.services.oexoord.OEXOEORDServices_DocStyle`) through a search.

## Searching Java APIs for Forms Interfaces

**ORACLE** Integration Repository Home Logout Preferences Help Personalize Page Diagnostics

Integration Repository

Search

Interface Name  Internal Name  Browse

Product Family  Interface Type

Product  Business Entity

[Hide More Search Options](#)

☒ **TIP** Select Category before a Category Value

Category  Web Service Status

Category Value  Standard

Interface Source  Standard Specification

Status  Scope

Name	Internal Name	Product	Type	Source	Status	Description
<a href="#">CreateAndMaintainSalesOrders</a>	oracle.apps.ont.services.oexoeord.OEXOEORDServices_DocStyle Order Management	Order Management	Java	Oracle	Active	The service end-point provides capabilities to Create, Delete, Query and Update a Sales Order.

From the main Search page, click **Show More Search Options** to display more fields for search. Enter the following criteria:

- Category: Interface Subtype
- Category Value: Java API for Forms

Click **Go** to execute the search. This retrieves all Java APIs for Forms interfaces from the Oracle Integration Repository.

Click the [CreateAndMaintainSalesOrders](#) Java API for Forms link to display the interface details page and click **Generate WSDL**.

Once the service is successfully generated, the Web Service - SOA Provider region appears in the interface details page. For detailed instruction on how to generate a Web service, see *Generating Web Services, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

2. To deploy a generated Web service, select at least one authentication type and click **Deploy** in the Web Service - SOA Provider region of the interface details page to deploy the service.

Once the service is successfully deployed, the selected authentication type(s) will be displayed along with 'Deployed' Web Service Status. For more information on securing Web services with authentication types, see *Managing Web Service Security, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

For detailed instruction on how to deploy a Web service, see *Deploying, Undeploying, and Redeploying Web Services, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

## Searching and Recording WSDL URL

Apart from the required tasks performed by the administrators, an integration developer also needs to log on to the system to locate and record the deployed Web service WSDL URL for the interface that needs to be orchestrated into a meaningful business process in Oracle JDeveloper using BPEL language.

This WSDL information will be used later in creating a partner link for the interface exposed as a Web service during the BPEL process creation at design time.

### Viewing and Recording a Deployed WSDL URL

The screenshot shows the Oracle Integration Repository interface. At the top, there's a navigation bar with links like Home, Logout, Preferences, Help, Personalize Page, and Diagnostics. Below this, the main content area displays details for a specific interface. The interface name is 'oracle.apps.ont.services.oexoeord.OEXOEORDServices\_DocStyle'. It is categorized as a 'Java' type, 'Order Management' product, and 'Active' status. The business entity is 'Sales Order', and the derived interface is 'Generated SOA Provider Document Interface'. The scope is 'Public', the interface source is 'Oracle', and the interface subtype is 'Java APIs for Forms'. A 'Full Description' section explains that the service end-point provides capabilities to Create, Delete, Query, and Update a Sales Order. Below this, a 'Web Service - SOA Provider' section shows the web service status as 'Deployed' and provides a 'View WSDL' link. The authentication type is set to 'SAML Token (Sender Vouches)'.

How to search for an interface and review the interface details, see Searching and Viewing Integration Interfaces, page 2-1.

### Setting Variables in SOAHeader for SOAP Request

You must populate certain variables in the BPEL process for SOAHeader elements to pass values that would be used to set applications context during service execution. These SOAHeader elements for Java APIs for Forms interfaces are *Responsibility*, *RespApplication*, *SecurityGroup*, *NLSLanguage*, and *Org\_Id*.

**Note:** The user information is defined by the `wsseUsername` property passed within the security headers. Detailed instructions on how to pass the security headers along with the SOAP request, see Passing Values to Security Headers, page 8-10.

The expected values for these elements are described in the following table:

### **Header Variables and Expected Values for Java APIs for Forms Interfaces**

Element Name	Expected Value
Responsibility	responsibility_key (such as "SYSTEM_ADMINISTRATOR")
RespApplication	Application Short Name (such as "FND")
SecurityGroup	Security Group Key (such as "STANDARD")
NLSLanguage (optional)	NLS Language (such as "AMERICAN")
Org_Id	Org Id (such as "202")

#### **Note:**

- Unlike SOAHeader used in other interface types to set context header, Org\_Id is a mandatory value that must be passed when using a Java APIs for Forms service.

- NLS Language is an optional value to be passed.

If the NLS Language element is specified, SOAP requests can be consumed in the language passed. All corresponding SOAP responses and error messages can also be returned in the same language. If no language is identified, then the default language of the user will be used.

The context information can be specified by configuring an Assign activity before the Invoke activity in the BPEL PM.

Detailed information on how to set SOAHeader for the SOAP request, see Assigning SOAHeader Parameters, page 8-20.

#### **BPEL Process Creation Flow**

Based on the sales order creation scenario, the following design-time tasks are discussed in this chapter:

1. Create a new BPEL project, page 8-6

Use this step to create a new BPEL project called `CreateSalesOrder.bpel` using an Synchronous BPEL Process template. This automatically creates two dummy activities - Receive and Reply - to receive input from a third party application and to reply output of the BPEL process back to the request application.

2. Create a Partner Link, page 8-7

Use this step to create an order in Oracle E-Business Suite by using the CreateAndMaintainSalesOrders (oracle.apps.ont.services.oexoeord.OEXOEORDServices\_DocStyle) API exposed as a Web service.

3. Add a Partner Link for File Adapter, page 8-11

Use this step to synchronous read order header details passed from the first Assign activity.

4. Add Invoke activities, page 8-15

Use this step to configure two Invoke activities in order to:

- Point to the File Adapter to synchronous read order header details that is passed from the first Assign activity.
- Point to the OEXOEORDServices\_DocStyle partner link to initiate the order creation with payload and transaction details received from the Assign activities.

5. Add Assign activities, page 8-19

Use this step to configure Assign activities in order to pass order header details, payload information and order number to appropriate Invoke activities to facilitate order creation. At the end, pass the order number to the request application through the dummy Reply activity.

For general information and basic concept of a BPEL process, see Understanding BPEL Business Processes, page D-1 and *Oracle BPEL Process Manager Developer's Guide* for details.

## Creating a New BPEL Project

Use this step to create a new BPEL project that will contain various BPEL process activities.

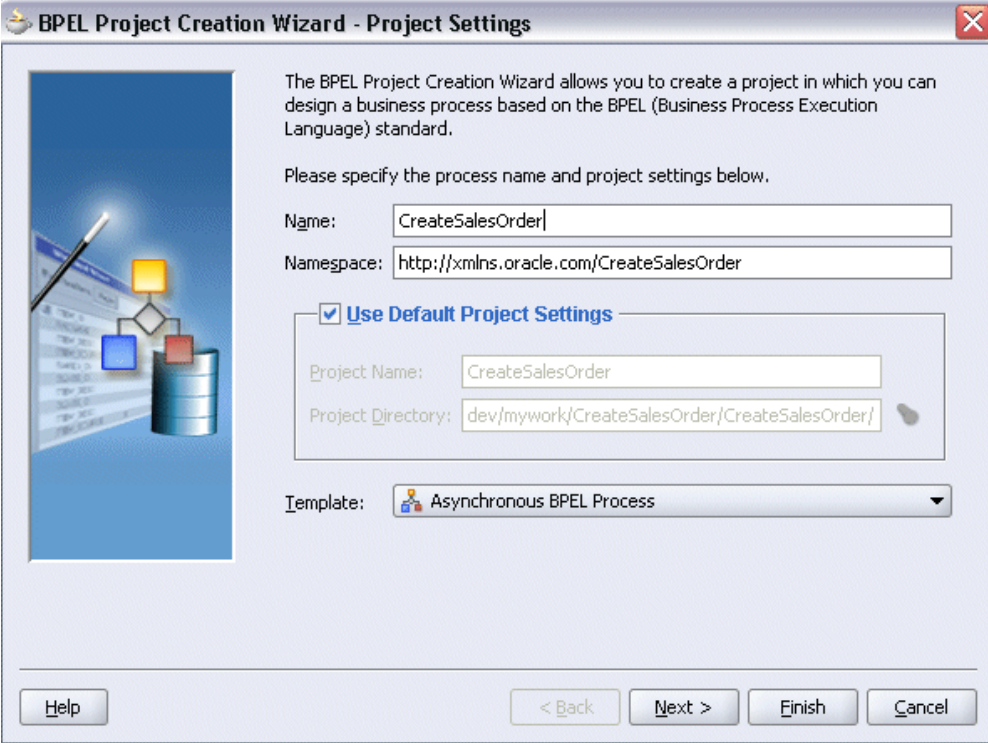
**To create a new BPEL project:**

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 produces a list of available categories.
4. Expand the **General** node and then select **Projects**.
5. Select **BPEL Process Project** from the **Items** group.



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

### Entering BPEL Project Information



The BPEL Project Creation Wizard allows you to create a project in which you can design a business process based on the BPEL (Business Process Execution Language) standard.

Please specify the process name and project settings below.

Name:

Namespace:

☒ **Use Default Project Settings**

Project Name:

Project Directory:

Template:

Buttons:

7. In the **Name** field, enter a descriptive name such as `CreateSalesOrder`.
8. From the Template list, select **Synchronous BPEL Process**. Select **Use Default Project Settings**.
9. Use the default input and output schema elements in the Input/Output Elements dialog box.
10. Click **Finish**.

A new synchronous BPEL process is created with the Receive and Reply activities. The required source files including `bpel.xml`, using the name you specified (for example `CreateSalesOrder.bpel`) are also generated.

## Creating a Partner Link

Use this step to create a Partner Link called `OEXOEORDServices_DocStyle`.

**To create a partner link for CreateOrder Web service:**

1. In JDeveloper BPEL Designer, drag and drop the **PartnerLink** service from the

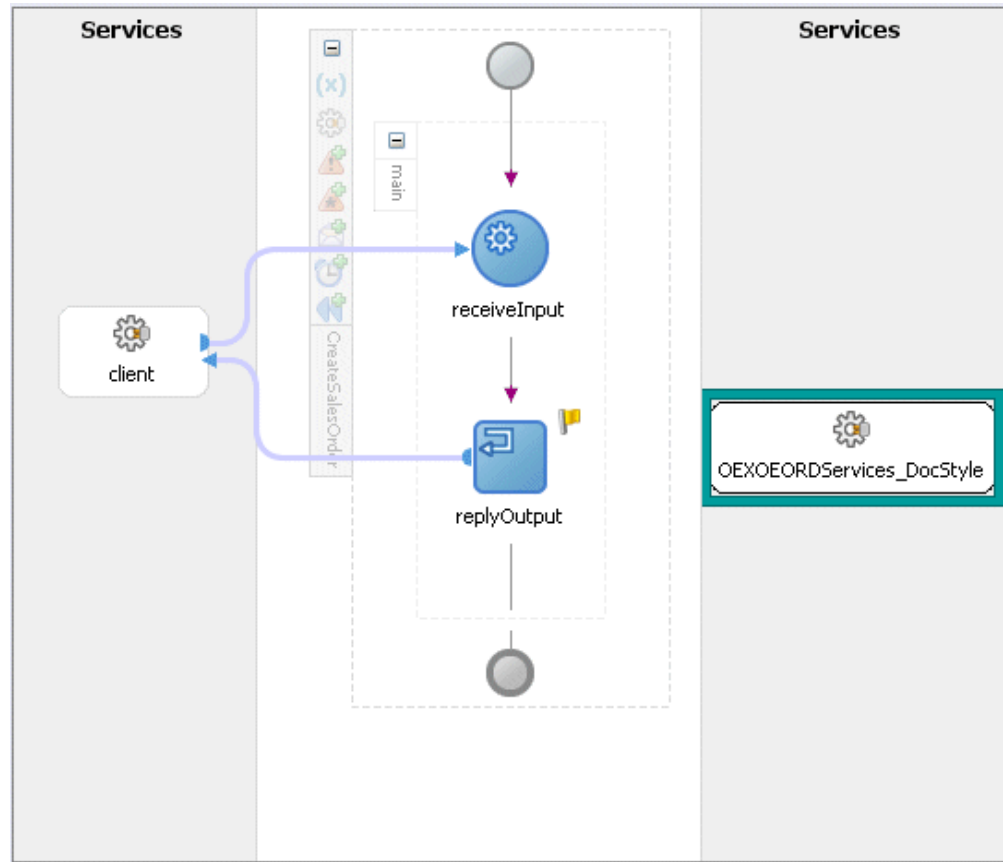
Component Palette into the Partner Link border area of the process diagram. The Service Name dialog box appears.

2. Copy the deployed WSDL URL corresponding to the CreateAndMaintainSalesOrders service that you recorded earlier in the WSDL File field.
3. A Partner Link Type message dialog box appears asking whether you want the system to create a new WSDL file that will by default create partner link types for you.

Click **Yes** to have the Partner Name, Partner Link Type values populated automatically.

You can select appropriate values for the Partner Role and My Role fields.

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.



4. You can optionally change the default partner link name by double-clicking the icon to open the Edit Partner Link window if you like.  
Select the Partner Role value from the drop-down list.  
Click **Apply**.

### Editing the Partner Link

**Edit Partner Link**

General Image Property

Name: OEXOEORDServices\_DocStyle

Process: CreateSalesOrder

**WSDL Settings**

WSDL File: file:/D:/JDev10133/jdev/mywork/CreateSalesOrder/C

Partner Link Type: OEXOEORDServices\_DocStyle\_PortType\_PL

Partner Role: OEXOEORDServices\_DocStyle\_PortType\_Role

My Role: ---- Not Specified ----

Help Apply OK Cancel

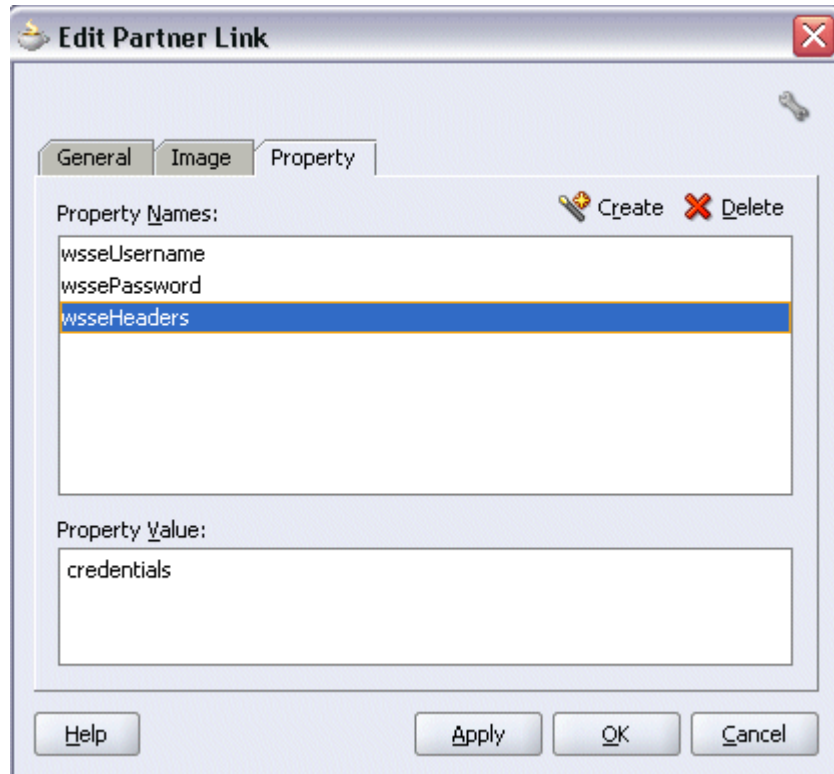
### 5. Passing Values to Security Headers

Select the Property tab and click the **Create Property** icon to select the following properties from the property name drop-down list in order to pass the security headers along with the SOAP request:

- wsseUsername  
Specify the username to be passed in the Property Value box.
- wssePassword  
Specify the corresponding password for the username to be passed in the Property Value box.
- wsseHeaders  
Enter `credentials` as the property value.

Click **Apply** to save the selected property values.

### Adding Properties



6. Click **OK** to complete the partner link configuration.

## Adding a Partner Link for File Adapter

Use this step to configure a BPEL process by reading current contents of a file.

### To add a Partner Link for File Adapter to Read Payload:

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 welcome page appears.
2. Click **Next**. The Service Name dialog box appears.
3. Enter a name for the file adapter service such as `ReadPayload`. You can add an optional description of the service.
4. Click **Next**. The Operation dialog box appears.

### Specifying the Operation

The File Adapter supports three operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, and a Synchronous Read File operation that reads the current contents of a file. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type:

- ☐ Read File
- ☐ Write File
- ☒ Synchronous Read File

Operation Name:

Help    < Back    Next >    Finish    Cancel

5. Specify the operation type, for example **Synchronous Read File**. This automatically populates the **Operation Name** field.

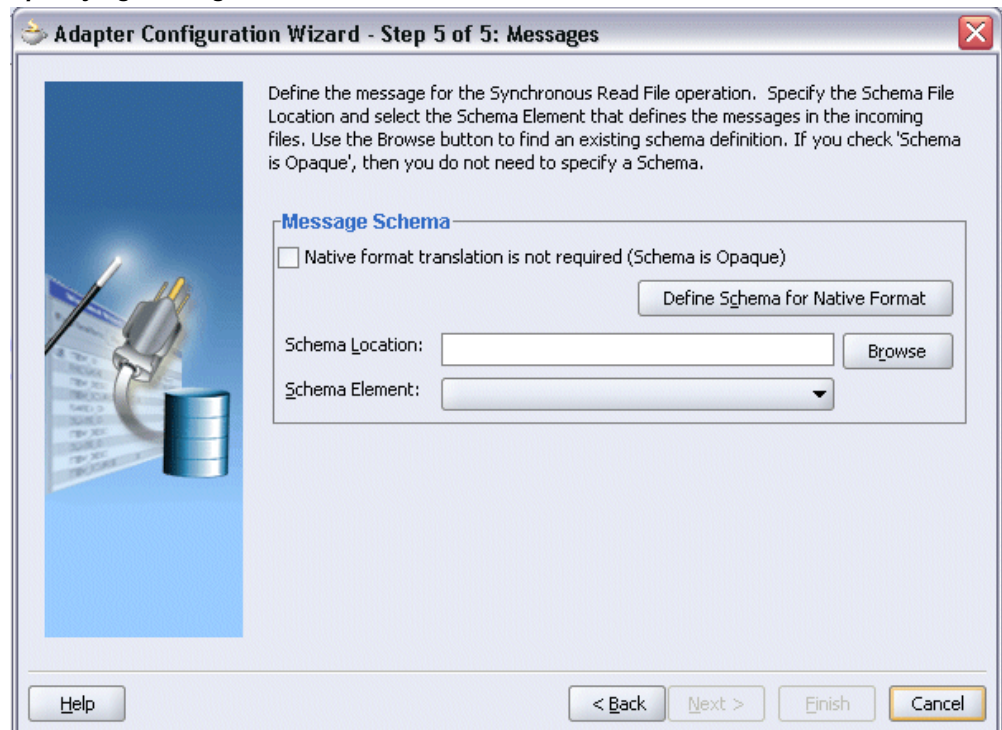
Click **Next** to access the File Directories dialog box.

6. Select **Physical Path** radio button and enter the input payload file directory information. For example, enter `/usr/tmp/` as the directory name.

Uncheck the **Delete Files after successful retrieval** check box. Click **Next** to open the File Name dialog box.

7. Enter the name of the file for the synchronous read file operation. For example, enter 'Input.xml'. Click **Next**. The Messages dialog box appears.

### Specifying Message Schema



8. Select **Browse** for schema file in Schema Location. The Type Chooser window is displayed.
  1. Click **Import Schema Files** button on the top right corner of the Type Chooser window.
  2. Enter the schema location for the service in the Import Schema File pop-up window.

Schema location for your service can be found from the service WSDL URL (for example,  
`http://<myhost>:<port>/webservices/SOAPProvider/java/oracle/apps/ont/services/oexoeord/OEXOEORDServices_DocStyle/?wsdl`).
  3. Select the **Add to Project** check box and click **OK**.
  4. Click **OK** for Import schema prompt.

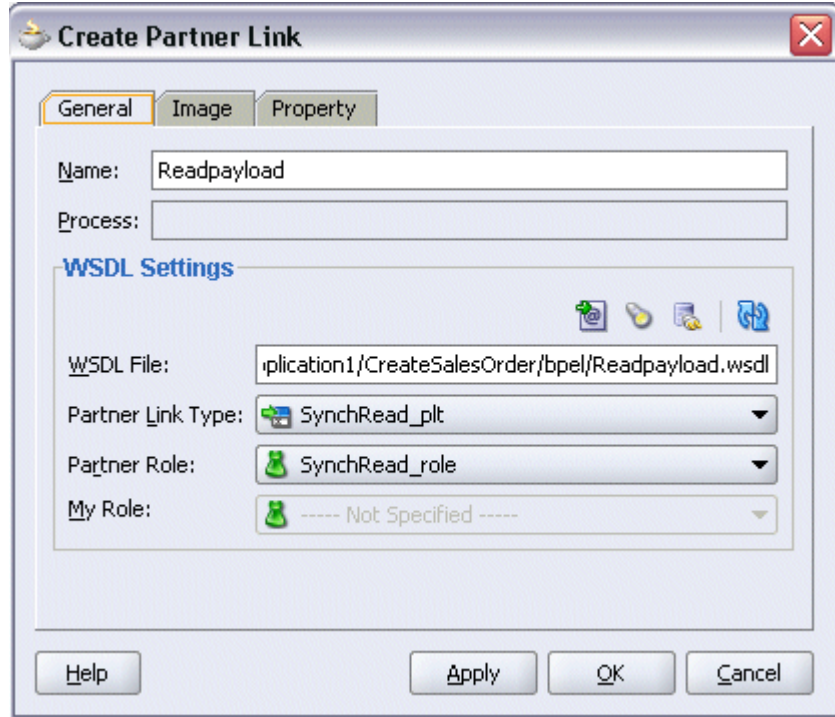
The Imported Schemas folder is automatically added to the Type Chooser window.
  5. Select an appropriate schema element by expanding the Imported Schemas folder. Click **OK**.



The selected schema location and element values are displayed.

9. Click **Next** and 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 `ReadPayload.wsdl`.

#### *Completing the Partner Link Configuration*

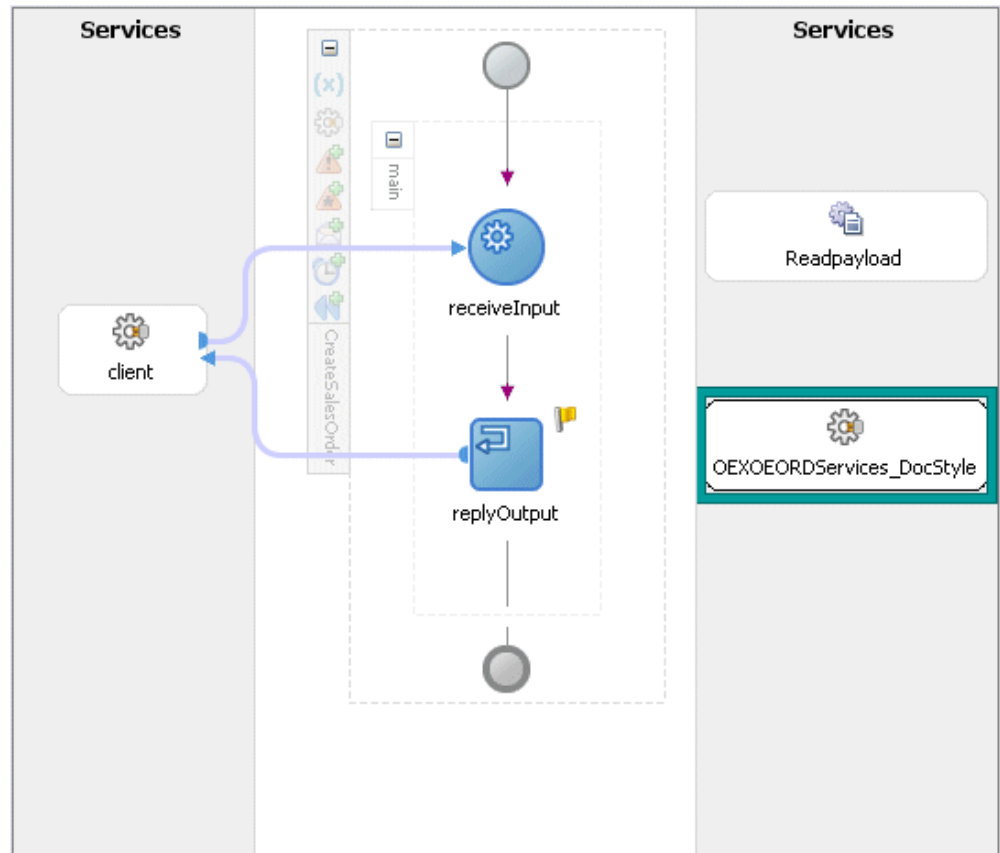
The image shows a 'Create Partner Link' dialog box with three tabs: 'General', 'Image', and 'Property'. The 'General' tab is selected. It contains a 'Name' field with the value 'Readpayload', an empty 'Process' field, and a 'WSDL Settings' section. The 'WSDL Settings' section includes a 'WSDL File' field with the path 'application1/CreateSalesOrder/bpel/Readpayload.wsdl', a 'Partner Link Type' dropdown menu set to 'SynchRead\_plt', a 'Partner Role' dropdown menu set to 'SynchRead\_role', and a 'My Role' dropdown menu set to '---- Not Specified ----'. At the bottom of the dialog are buttons for 'Help', 'Apply', 'OK', and 'Cancel'.

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

The `ReadPayload` Partner Link appears in the BPEL process diagram.



### Adding the Partner Link for File Adapter



## Adding Invoke Activities

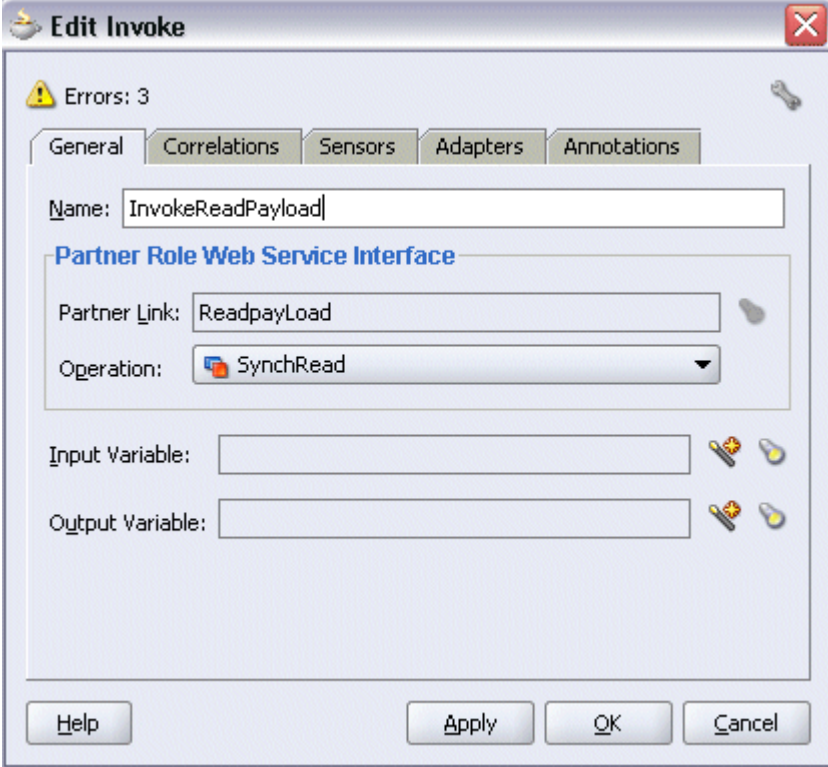
This step is to configure two Invoke activities:

- Read sales order creation details that is passed from the first Assign activity using ReadPayload partner link for File Adapter.
- Send the payload and order details received from the Assign activities to create an sales order by using the CreateSalesOrder partner link.

### To add an Invoke activity for ReadPayload Partner Link:

1. In JDeveloper BPEL Designer, drag and drop the **Invoke** activity from the Component Palette into the Activity box of the process diagram, between the **Receive** and **Reply** activities.
2. Link the Invoke activity to the ReadPayload service. The Invoke activity will send invoice data to the partner link. The Edit Invoke dialog box appears.

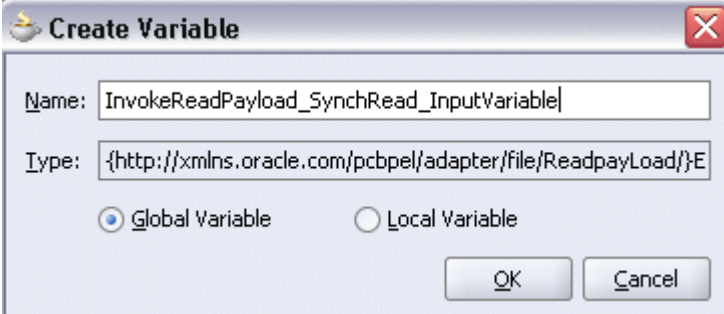
### Editing the Invoke Activity



The **Edit Invoke** dialog box is shown with the **General** tab selected. It features a **Name** field containing "InvokeReadPayload". Below this is a section titled **Partner Role Web Service Interface** with a **Partner Link** field containing "ReadpayLoad" and an **Operation** dropdown menu set to "SynchRead". At the bottom of this section are **Input Variable** and **Output Variable** fields, each with a **Create** icon (a lightbulb with a plus sign) to its right. The dialog also includes a **Help** button and **Apply**, **OK**, and **Cancel** buttons at the bottom.

3. Enter a name for the Invoke activity, and 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



The **Create Variable** dialog box is shown. It has a **Name** field containing "InvokeReadPayload\_SynchRead\_InputVariable" and a **Type** field containing "{http://xmlns.oracle.com/pcbpel/adapter/file/ReadpayLoad/}E". Below these fields are two radio buttons: **Global Variable** (which is selected) and **Local Variable**. At the bottom are **OK** and **Cancel** buttons.

4. Select **Global Variable**, and then enter a name for the variable. You can also accept the default name. Click **OK**.
5. click the **Create** icon next to the **Output Variable** field to create a new variable. The

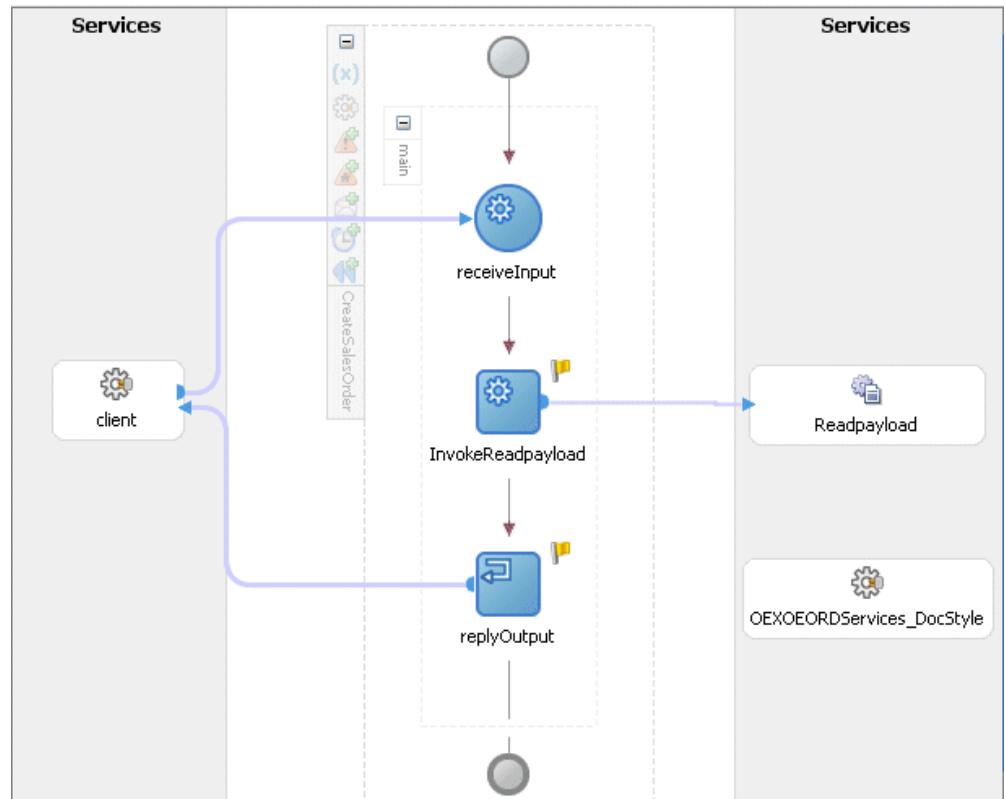
Create Variable dialog box appears.

Select **Global Variable**, and then enter a name for the variable. You can also accept the default name. Click **OK**.

6. Click **Apply** and then **OK** in the Edit Invoke dialog box to finish configuring the Invoke activity.

The Invoke activity appears in the process diagram.

#### **Adding an Invoke Activity**



#### **To add an Invoke activity for OEXOEORDServices\_DocStyle Partner Link:**

1. In JDeveloper BPEL Designer, drag and drop the **Invoke** activity from the Component Palette into the Activity box of the process diagram, after the **Invoke** and **Reply** activities.
2. Link the Invoke activity to the `OEXOEORDServices_DocStyle` service. The Invoke activity will send event data to the partner link. The Edit Invoke dialog box appears.
3. Enter a name for the Invoke activity such as 'InvokeCreateOrder'. Click the **Create** icon next to the **Input Variable** field to create a new variable. The Create Variable

dialog box appears.

#### **Creating a Variable**



Select **Global Variable**, and then enter a name for the variable. You can also accept the default name. Click **OK**.

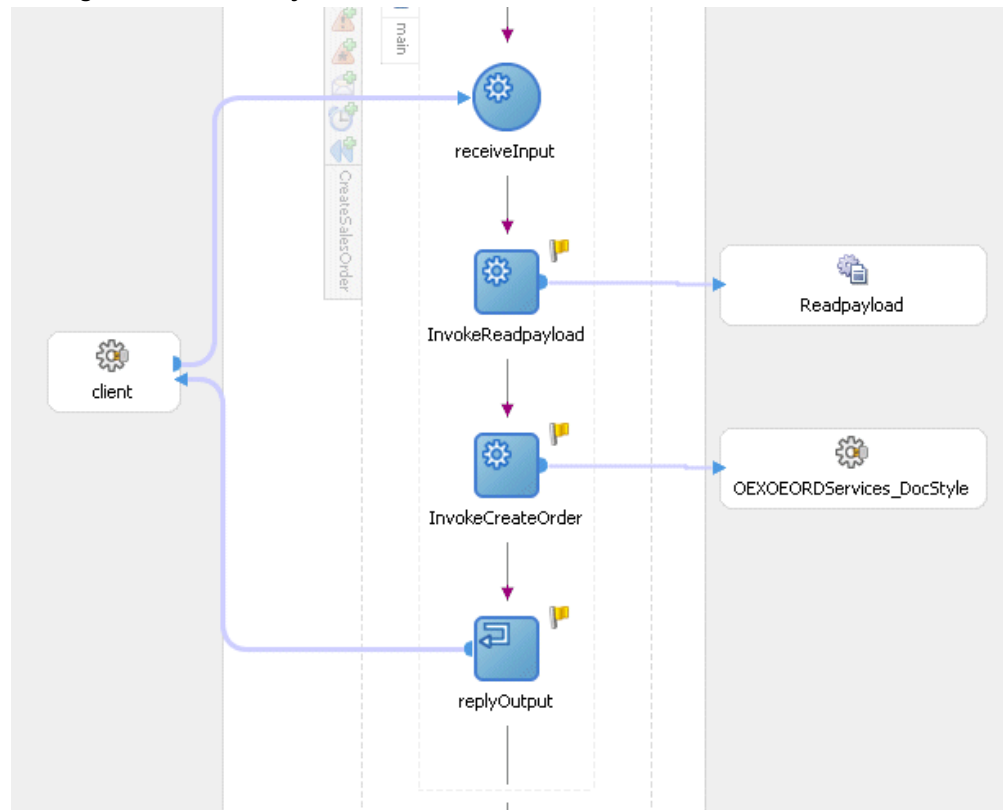
4. click the **Create** icon next to the **Output Variable** field to create a new variable. The Create Variable dialog box appears.

Select **Global Variable**, and then enter a name for the variable. You can also accept the default name. Click **OK**.

5. Click **Apply** and then **OK** in the Edit Invoke dialog box to finish configuring the Invoke activity.

The Invoke activity appears in the process diagram.

### Adding an Invoke Activity



### Adding Assign Activities

This step is to configure four Assign activities:

1. To set the SOAHeader for SOAP request.

**Note:** You need to populate certain variables in the BPEL process for SOAHeader elements to pass values that may be required to set application context during service execution. These SOAHeader elements are *Responsibility*, *RespApplication*, *SecurityGroup*, *NLSLanguage*, and *Org\_Id*.

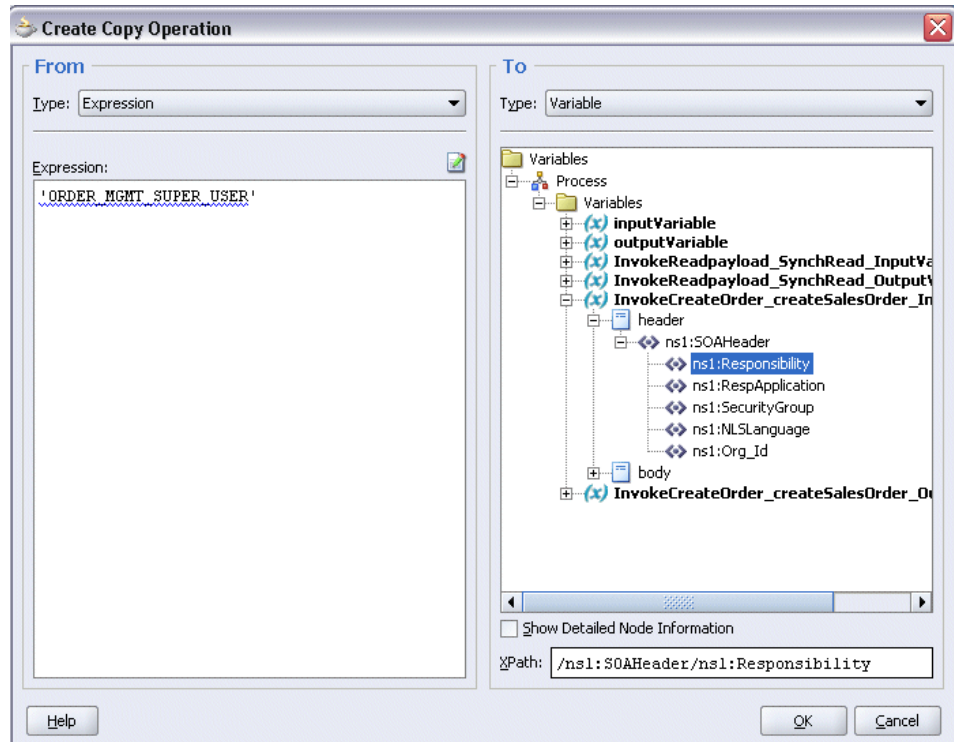
2. To pass the payload information to the `InvokeCreateOrder` Invoke activity.
3. To pass the order number information to the `InvokeCreateOrder` Invoke activity.
4. To pass the order number information back to the dummy Reply activity.

**To add the first Assign activity to set SOAHeader for SOAP request:**

*Assigning SOAHeader Parameters:*

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 two **Invoke** activities.
2. Double-click the **Assign** activity to access the Edit Assign dialog box.
3. Click the General tab to enter the name for the Assign activity, such as 'SetHeader'.
4. On the Copy Operation tab, click **Create** and then select **Copy Operation** from the menu. The Create Copy Operation window appears.
5. Enter the first pair of parameters:
  - In the From navigation tree, select type Expression and then enter 'ORDER\_MGMT\_SUPER\_USER' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > InvokeCreateOrder\_createSalesOrder\_InputVariable > header > ns1:SOAHeader** and select **ns1:Responsibility**. The XPath field should contain your selected entry.

### Assigning the Responsibility Parameter



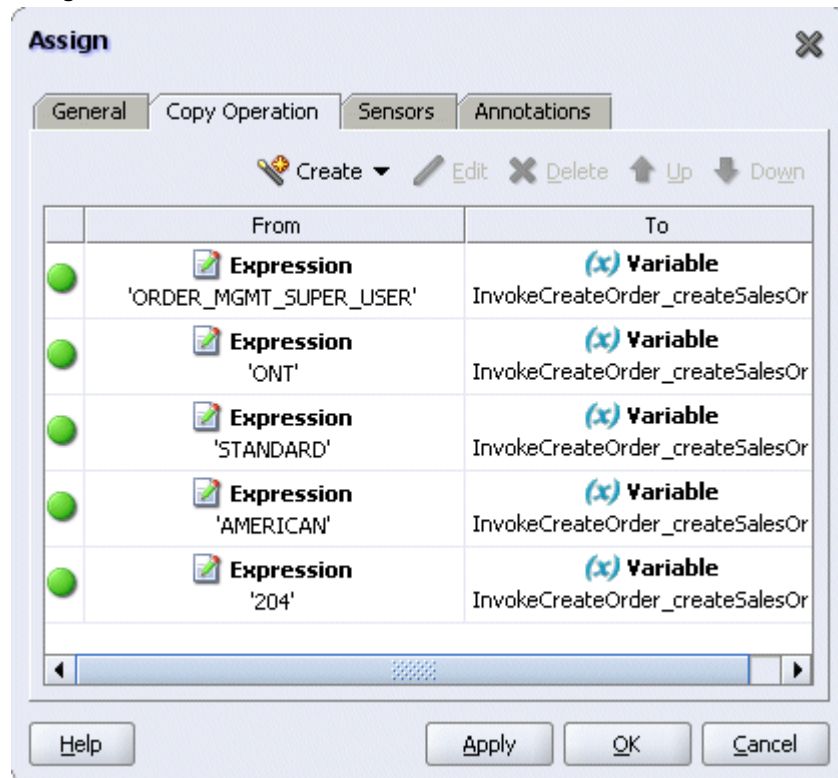
- Click **OK**.
6. Enter the second pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
- In the From navigation tree, select type Expression and then enter 'ONT' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > InvokeCreateOrder\_createSalesOrder\_InputVariable > header > ns1:SOAHeader** and select **ns1:RespApplication**. The XPath field should contain your selected entry.
  - Click **OK**.
7. Enter the third pair of parameters:
- In the From navigation tree, select type Expression and then enter 'STANDARD' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process >**

**Variables > InvokeCreateOrder\_createSalesOrder\_InputVariable > header> ns1:SOAHeader** and select **ns1:SecurityGroup**. The XPath field should contain your selected entry.

- Click **OK**.
8. Enter the fourth pair of parameters:
- In the From navigation tree, select type Expression and then enter 'AMERICAN' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > InvokeCreateOrder\_createSalesOrder\_InputVariable > header> ns1:SOAHeader** and select **ns1:NLSLanguage**. The XPath field should contain your selected entry.
  - Click **OK**.
9. Enter the fifth pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
- In the From navigation tree, select type Expression and then enter '204' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > InvokeCreateOrder\_createSalesOrder\_InputVariable > header> ns1:SOAHeader** and select **ns1:Org\_Id**. The XPath field should contain your selected entry.
  - Click **OK**.
10. The Edit Assign dialog box appears.



### Assign Parameters



11. Click **Apply** and then **OK** to complete the configuration of the Assign activity.

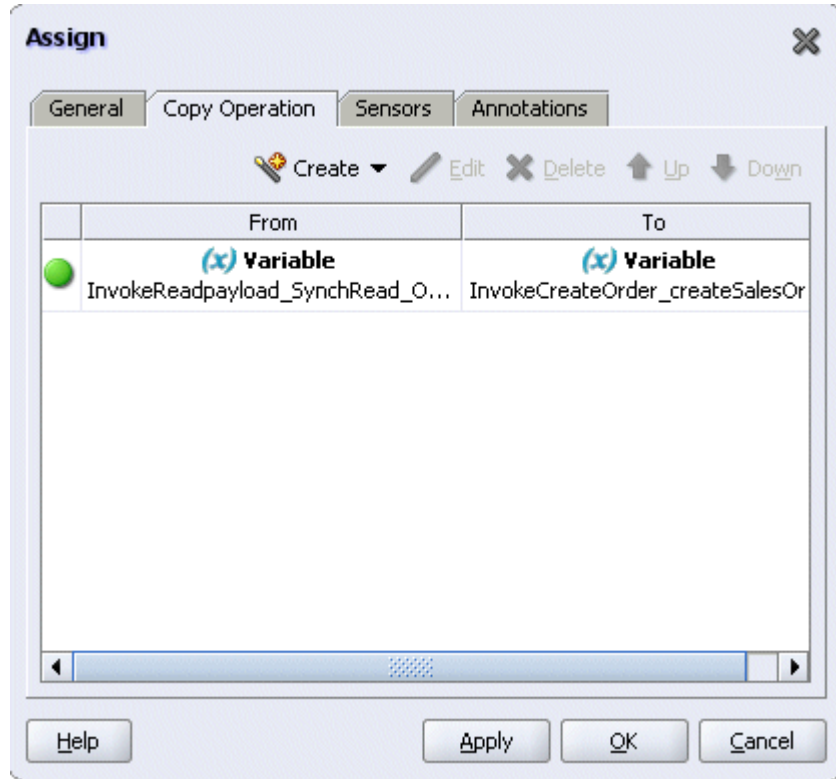
### To enter the second Assign activity to pass payload information to the InvokeCreateOrder Invoke activity:

1. Add the second Assign activity by dragging and dropping the **Assign** activity from the Component Palette into the Activity box of the process diagram, between **Assign** and **Invoke** activities.
2. Repeat Step 2 to Step 4 described in creating the first Assign activity to add the second Assign activity called 'SetPayload'.
3. Enter the following information:
  - In the From navigation tree, navigate to **Variable > Process > Variables > InvokeReadPayload\_SynchRead\_OutputVariable** and select an appropriate variable.
  - In the To navigation tree, select type Variable and then navigate to **Variable > Process > Variables > InvokeCreateOrder\_createSalesOrder\_InputVariable > Body** and select **ns1:createSalesOrder\_Request**. The XPath field should contain

your selected entry.

- Click **OK**.
4. The Edit Assign dialog box appears.

#### Assign Parameters



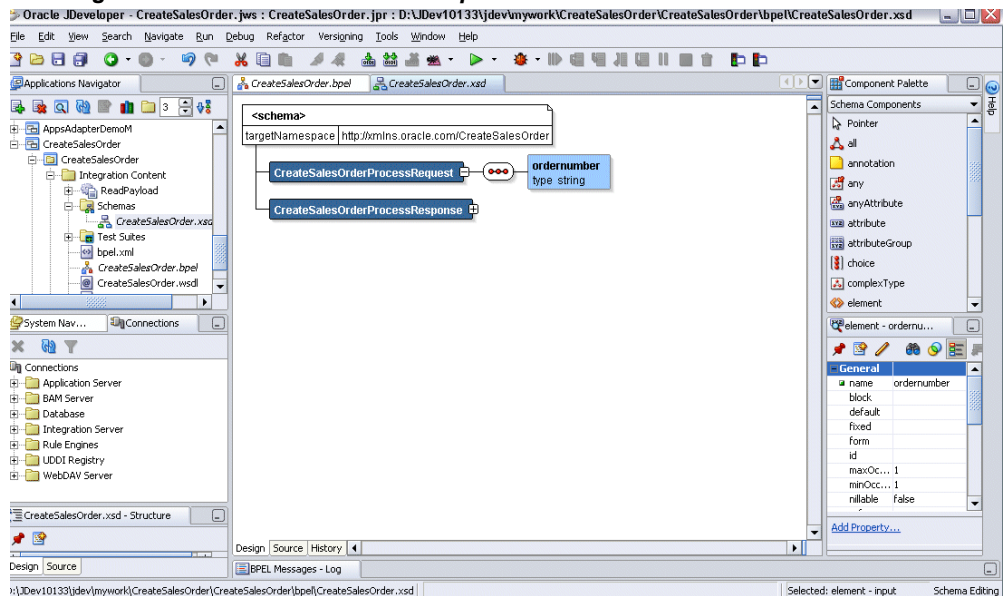
5. Click **Apply** and then **OK** to complete the configuration of the Assign activity.

#### Defining Schema for BPEL Process Input Request

Before setting the input request for the SOAP request, you need to define necessary schema for BPEL process request.

1. From the Applications Navigator window, expand the **CreateSalesOrder > Integration Content > Schemas** folder to open the `CreateSalesOrder.xsd` file.
2. In the Design mode, expand 'CreateSalesOrderProcessRequest' to view elements within process request.

### Defining Schema for BPEL Process Request



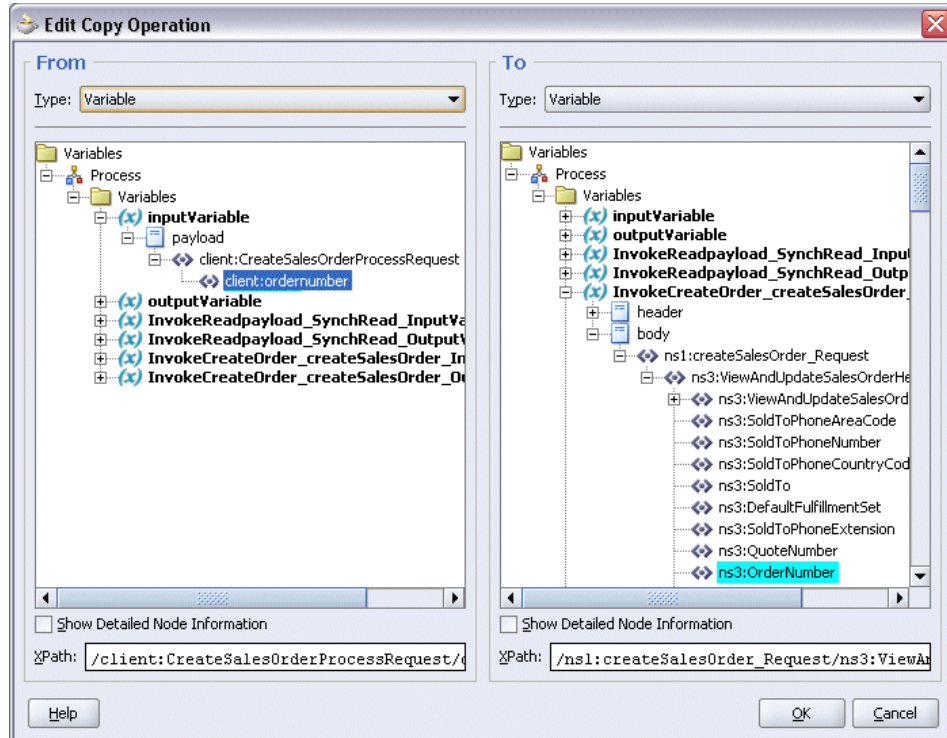
3. From element properties, change the name from 'input' to 'ordernumber'.

4. Right-click on mouse and select **Rebuild** option.

Look for compilation messages in Log to ensure the successful compilation.

**To enter the third Assign activity to pass the order number to the InvokeCreateOrder Invoke activity:**

1. Add the third Assign activity by dragging and dropping the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the second **Assign** activity and the **InvokeCreateOrder Invoke** activity.
2. Repeat Step 2 to Step 4 described in creating the first Assign activity to add the third Assign activity called 'SetOrderNumber'.
3. Enter the following information:
  - In the From navigation tree, navigate to **Variable > Process > Variables > inputVariable > Payload > client:CreateSalesOrderProcessRequest** and select **client:ordernumber**. The XPath field should contain your selected entry.
  - In the To navigation tree, select type Variable and then navigate to **Variable > Process > Variables > InvokeCreateOrder\_createSalesOrder\_inputVariable > Body > ns1:createSalesOrder\_Request > ns3:ViewAndUpdateSalesOrderHeaderDetails** and select **ns3:OrderNumber**. The XPath field should contain your selected entry.



- Click **OK**.

4. The Edit Assign dialog box appears.

Click **Apply** and then **OK** to complete the configuration of the Assign activity.

**To add the fourth Assign activity to reply back customer order number:**

1. Add the third Assign activity by dragging and dropping the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the **InvokeCreateOrder** **Invoke** and the **Reply** activities.
2. Repeat Step 2 to Step 4 described in creating the first Assign activity to add the fourth Assign activity called 'SetCustomNumber'.
3. Enter the following information:
  - In the From navigation tree, select type Variable. Navigate to **Variable** > **Process** > **Variables** > **InvokeCreateOrder\_createSalesOrder\_OutputVariable** and select **body**.
  - In the To navigation tree, select type Variable. Navigate to **Variable** > **Process** > **Variables** > **outputVariable** and select **payload**.
  - Click **OK**.

4. The Edit Assign dialog box appears.
5. Click **Apply** and then **OK** to complete the configuration of the Assign activity.

## Deploying and Testing the BPEL Process

After creating a BPEL process using the WSDL URL generated from the Java API for Forms interface definition, you can deploy it to a BPEL server if needed. To ensure that this process is modified or orchestrated appropriately, you can also manually test the BPEL process by initiating the business process contained in the BPEL process to test the interface integration.

### Prerequisites

Before deploying the BPEL process using Oracle JDeveloper, you must ensure that you have established the connectivity between the design-time environment and the run-time servers including the application server and the integration server.

How to configure the necessary server connection, see *Configuring Server Connection*, page B-1.

To validate your BPEL process, perform the following run-time tasks:

1. Deploy the BPEL process, page 8-27

Once you deploy the process to a BPEL server, it becomes available so that you can run the process manually to test it for validation.

2. Test the BPEL process, page 8-28

After deploying a BPEL process, you can manage the process from the BPEL console to manually initiate the business process and test the interface integration contained in your BPEL process.

## Deploying the BPEL Process

You must deploy the Create Sales Order BPEL process (`CreateSalesOrder.bpel`) that you created earlier before you can run it.

### To deploy the BPEL process:

1. In the Applications Navigator of JDeveloper BPEL Designer, select the **CreateSalesOrder** project.
2. Right-click the project and click **Make** action from the menu.

Look for any compilation error messages in Message Log.

Right-click the project and select **Deploy >Integration Server Connection name > Deploy to Default Domain** action from the menu.

For example, you can select **Deploy > BPELServerConn > Deploy to Default Domain** to deploy the process if you have the BPEL Process Manager setup appropriately.

3. Look for 'Build successful' message in Apache Ant – Log to ensure that the BPEL project is compiled and successfully deployed.

## Testing the BPEL Process

To validate whether the BPEL process that you created works or not, you need to manually initiate the process after it has been successfully deployed to the BPEL server. Therefore, the validation starts with the BPEL console to ensure that you can find the deployed BPEL process listed in the console. Then, you can log on to Oracle E-Business Suite to validate the process creation.

### To test the BPEL process:

1. Log into Oracle Application Server 10g BPEL Console (<http://<soaSuiteServerHostName>:<port>/BPELConsole>). The BPEL Console login page appears.



You can also manage the list of domains using BPEL Admin:

➔ [Goto BPEL Admin](#)

2. Enter the username and password and click **Login**.

The Oracle Enterprise Manager 10g BPEL Control appears. The list of deployed processes is shown under Deployed BPEL Processes. You can confirm that CreateSalesOrder has been deployed.

3. Click the CreateSalesOrder link to open the Initiate tab
4. In the payload section, enter an unique number in the ordernumber field, such as BPEL\_1, and click **Post XML Message** to initiate the process.
5. **Verifying Records in Oracle E-Business Suite**  
Log on to the Forms-based Oracle E-Business Suite with the Oracle Management Super User, Vision Operation (USA) responsibility.
6. Select **Order Returns > Sales Order**. Sales Order Forms would open up.
7. Search for an order by entering the order number (such as BPEL\_1) in the Customer PO field. This would bring up the details of a newly created order.





---

# Using Composite Services - BPEL

## Overview

Composite services use native services as building blocks to orchestrate the business invocation sequence from discrete Web services into a meaningful end-to-end business flow through a Web service composition language BPEL. Strictly speaking, this type of interface is comparatively service enabled without additional service generation process as required by native interface types.

At design time, based on business needs, an integration developer can create a composite service - BPEL type by using any of the Web service WSDL URL that has been successfully generated and deployed to Oracle Application Server.

At run time, the developer can also view each composite service details by selecting an appropriate composite service from the Oracle Integration Repository browser, download the selected composite service from the repository to their local directories, open them in Oracle JDeveloper to modify the BPEL project if necessary before deploying it to a BPEL server in Oracle SOA Suite or a third party BPEL PM server.

This chapter discusses each run-time task listed below for using BPEL composite services. Detailed design-time tasks on how to create a BPEL composite service are included in each individual interface described earlier in this book.

- Viewing composite services, page 9-2
- Downloading composite services, page 9-2
- Modifying and deploying BPEL processes, page 9-4

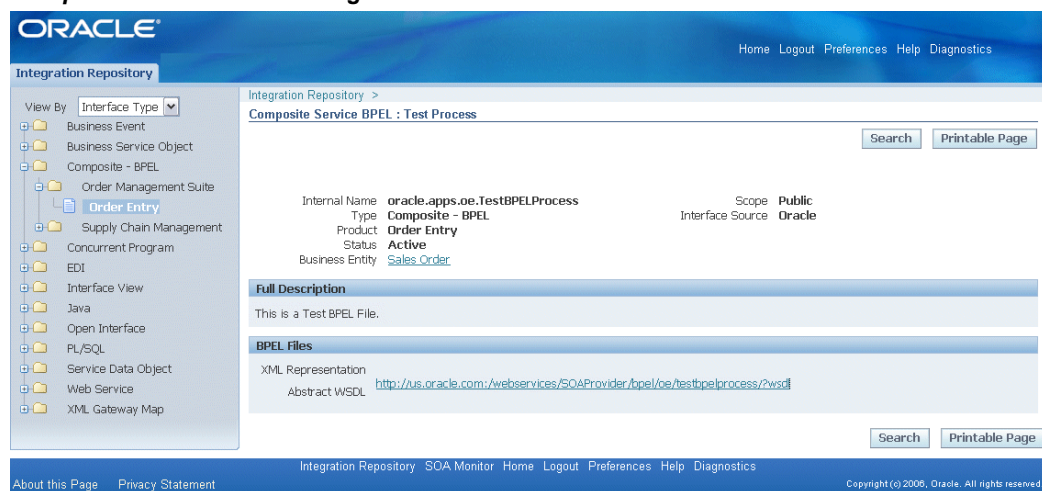
For general information and basic concept of a BPEL process, see Understanding BPEL Business Processes, page D-1.

## Viewing Composite Services

Similar to all other users, system integration developers can view a composite service by navigating to the Composite Service interface type directly from the Oracle Integration Repository Browser window or by performing a search by selecting Composite Service interface type in the Search page.

Clicking on a composite service name link from the navigation tree or search results, you will find the composite service interface details page where displays composite service name, description, BPEL file, and other annotated information.

### Composite Service Details Page



The composite service details page allows you to perform the following tasks in the BPEL Files region:

- View an abstract WSDL file by clicking the URL link
- Review XML representation file by clicking the URL link

You can also download a corresponding composite service project file, such as BPEL file, to your local machine. See: Downloading Composite Services, page 9-2.

## Downloading Composite Services

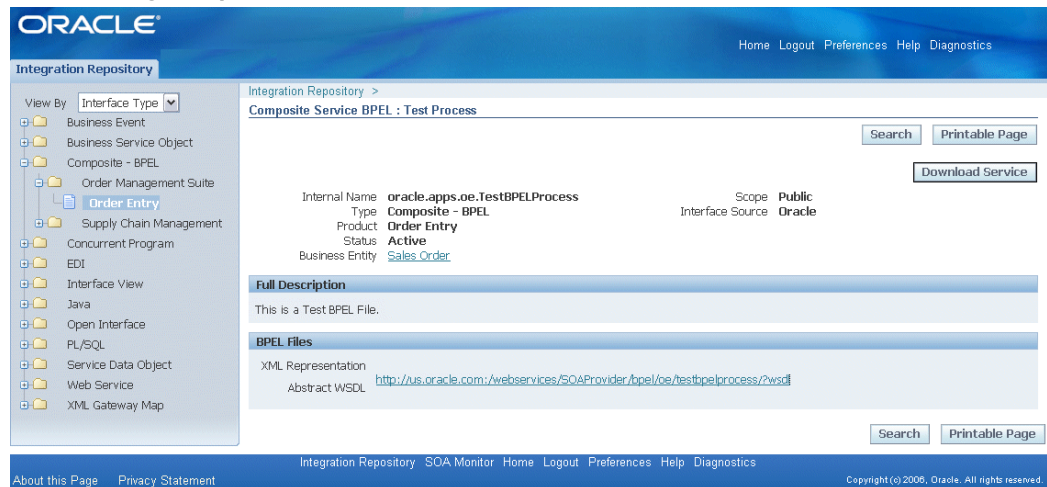
In addition to viewing composite service details and reviewing a WSDL abstract, the developers can download the composite service relevant files aggregated in a .JAR file to your local machine.

**Important:** In general, only system integration developers and integration repository administrators can download the composite

services. However, general users (system integration analysts) who are granted with the download privilege, an Integration Repository Download Composite Service permission set FND\_REP\_DOWNLOAD\_PERM\_SET, can also perform the download action.

For more information on how to grant Download Composite Service privilege, see Role-Based Access Control (RBAC) Security, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

### Downloading Composite Services



To download the .ZIP file for a composite service, navigate to the composite service details page for a service that you want to download, and then click **Download Service** to download the file to your local machine.

After you download the file, you can unzip the BPEL .JAR file and open the BPEL process in Oracle JDeveloper for further modification on service endpoints if needed. Additionally, You can deploy the BPEL process to a BPEL server through Oracle BPEL Process Manager. How to modify and deploy BPEL projects, see Modifying and deploying BPEL processes, page 9-4.

#### To download a composite service:

1. Log on to Oracle Integration Repository with the system integration developer role. Select the Integrated SOA Gateway responsibility from the navigation menu. Select the Integration Repository link.
2. In the Integration Repository tab, select 'Interface Type' from the View By drop-down list.
3. Expand the Composite Service interface type node to locate your desired composite

service.

4. Click the composite service that you want to download it to open the Composite Service Interface Details page.
5. Click **Download Service** to download the selected composite file to your local machine.

## Modifying and Deploying BPEL Processes

After downloading a composite service BPEL project, an integration developer can optionally modify the BPEL project. This can be done by first unzipping the BPEL .JAR file and then opening the BPEL file in Oracle JDeveloper to modify the BPEL process endpoints if necessary.

Additionally, the BPEL process can be further deployed to a BPEL server in Oracle SOA Suite BPEL PM or a third party BPEL PM in a J2EE environment. To ensure that this process is modified or orchestrated appropriately, you can manually test the BPEL process by initiating the business process contained in the BPEL process to test the interface integration.

The modification of a BPEL process uses the similar logic during the BPEL process creation. See Understanding BPEL Business Processes, page D-1 and design-time tasks for each interface type discussed earlier in this book.

How to test and validate the BPEL process that contains an interface exposes as a Web service, refer to the run-time tasks of the interface type described in this book.

For BPEL process modification and deployment described in this section, we use Oracle JDeveloper 10.1.3.3.0 to modify the BPEL process and use Oracle SOA Suite BPEL server 10.1.3.3.0 for the process deployment.

### To modify a BPEL process:

1. Open a BPEL file in Oracle JDeveloper BPEL Designer.
2. From the **File** menu, select **Open**.
3. Locate your BPEL file from the directory that you want to modify. Click **Open** in the Open window.
4. The selected BPEL process diagram appears.
5. Modify the BPEL process endpoints if necessary.
6. Save your work.

### To deploy a BPEL process:

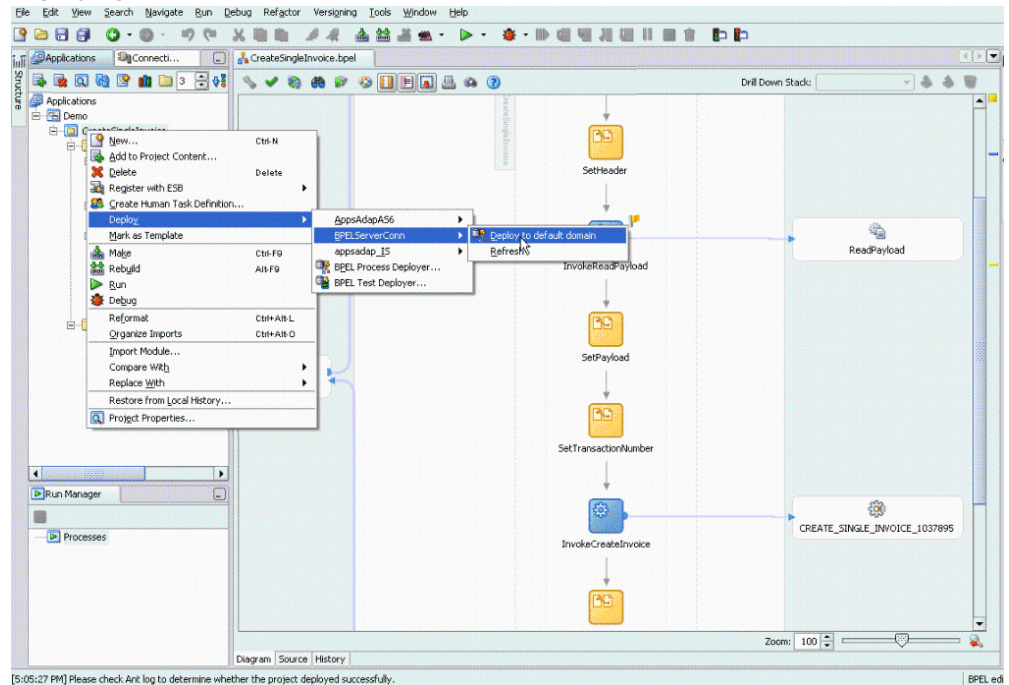
1. In the Applications Navigator of JDeveloper BPEL Designer, select the BPEL project

that you want to deploy.

2. Right-click the project and select **Deploy** action from the menu. Click on **Invoke Deployment Tool** and enter your BPEL Process Manager information.

For example, you can select **Deploy > BPELServerConn > Deploy to Default Domain** to deploy the process if you have the BPEL Process Manager setup appropriately.

### Deploying the BPEL Process



3. The Password Prompt dialog box appears.

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

The BPEL project is compiled and successfully deployed.



---

# Creating and Using Custom Integration Interfaces

## Overview

To support custom integration interfaces, Oracle E-Business Suite Integrated SOA Gateway provides a mechanism allowing these custom interfaces to be displayed through the Oracle Integration Repository browser along with Oracle packaged interfaces. This enables Oracle Integration Repository a single source of truth in centrally displaying all integration interfaces regardless of custom or Oracle packaged ones within the entire Oracle E-Business Suite.

Custom interface definitions can be created for various interface types including custom interface definitions for XML Gateway Map, Business Event, PL/SQL, Concurrent Program, Business Service Object, Java (except for Java APIs for Forms subtype) and Composite Service for BPEL type.

**Note:** Please note that custom interface types of EDI, Open Interface Tables, Interface Views, and Java APIs for Forms interfaces are not supported in this release.

Oracle Integration Repository currently does not support the creation of custom Product Family and custom Business Entity.

Depending on your business needs, integration developers can create custom integration interfaces first, and then annotate the custom interfaces based on the Integration Repository annotation standards. Once these custom interfaces are annotated, appropriate validation on the annotated custom interfaces is required before they are uploaded to the Integration Repository by an integration repository administrator.

### For custom integration interfaces of interface types

If a custom interface created for a supported interface type has been uploaded to Oracle

Integration Repository, to use this custom interface, an integration repository administrator should first create necessary security grants, and then generate and deploy the Web service WSDL file to the application server if the custom interface type can be service enabled. Thus, the deployed service can be exposed to customers through a service provider and invoked through any of the Web service clients or orchestration tools.

#### **For custom composite service - BPEL type**

If a custom interface is needed for a composite service - BPEL type, the integration developer will first create a composite service by orchestrating discrete native services into a meaningful process flow using BPEL. Based on the annotation standards specifically for composite service, the developer will then annotate the composite service, and create and unzip the JAR file of the BPEL project. Like custom interfaces of other interface types, appropriate validation on the BPEL project JAR file is required to ensure its compliance with the annotation standards before it is uploaded to the Integration Repository.

To have a better understanding of how to create custom interfaces as well as how to use custom interfaces as Web services, the following topics are discussed in this chapter:

- Creating Custom Integration Interfaces, page 10-2
- Using Custom Integration Interfaces as Web Services, page 10-22

## **Creating Custom Integration Interfaces**

The following topics are discussed in this section:

- Creating Custom Integration Interfaces of Interface Types, page 10-2
- Creating Custom Integration Interfaces of Composite Services, page 10-8
- Creating Custom Business Events Using Workflow XML Loader, page 10-14

### **Creating Custom Integration Interfaces of Interface Types**

Custom interface definitions can be created and annotated for almost all interface types. After appropriate validation, these custom interfaces will be uploaded to Oracle Integration Repository and embedded into the interface categories where they belong.

**Note:** Please note that custom interface types of EDI, Open Interface Tables and Interface Views are not supported in this release.

Oracle Integration Repository currently does not support the creation of custom Product Family and custom Business Entity.



## Enabling Custom Integration Interfaces

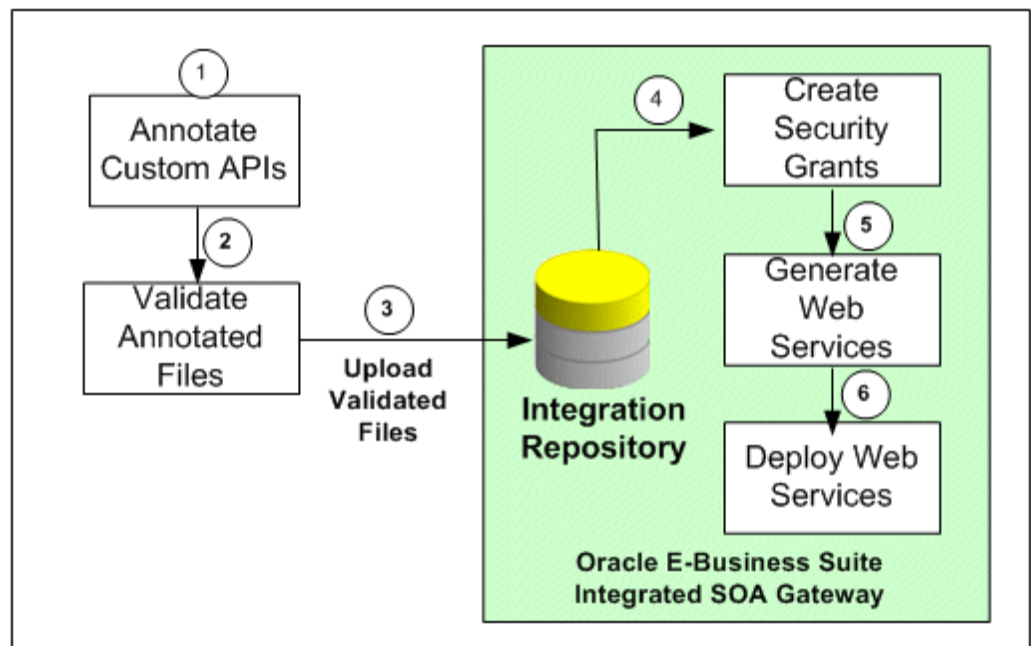
The custom interface design and service enablement process flow can be illustrated in the following diagram:

**Note:** Not all integration interface definitions can be service enabled. Oracle Integration Repository supports service enablement only for the following interface types:

- PL/SQL
- XML Gateway Map (inbound)
- Concurrent Program
- Business Service Object (Service Beans)

Please note that the Business Event and XML Gateway Map (outbound) interface types are supported through subscription model.

### Enabling Custom Integration Interfaces and Services



1. A system integration developer annotates a custom integration interface definition based on the Integration Repository annotation standards for the supported interface types.

See: Creating and Annotating Custom Integration Interfaces, page 10-4.

2. The integration repository administrator validates the annotated custom interface definitions against the annotation standards. This validation is performed by executing the Integration Repository Parser to read the annotated file and then generate an Integration Repository loader file (iLDT) if no error occurred.
3. If no error occurs during the validation, an integration repository administrator will then upload the generated iLDT file to Oracle Integration Repository through backend processing.

How to generate and upload the iLDT files, see *Generating and Uploading iLDT Files*, page 10-7.

After the upload, you can search and view the uploaded custom interface from the Integration Repository user interface for verification.

4. The administrator then creates necessary security grants for the custom interfaces.
5. The administrator generates Web services for selected custom interfaces if the interface types that the custom interfaces belong to can be service enabled.
6. The administrator deploys the WSDL Web services from Oracle Integration Repository to the application server.

See: *Viewing Custom Interfaces and Performing Administrative Tasks*, page 10-7.

### **Custom Integration Interface Annotation Example**

The key essence of successfully creating custom integration interfaces relies on properly explanation of the new interface feature or definition. Once a custom interface definition of a specific interface type is created, an integration developer must properly annotate the custom file based on the Integration Repository annotation standards so that the interface file of a specific interface type can be displayed with appropriate description from the browser interface.

For example, the integration developer can create a Supplier Ship and Debit Request custom interface using PL/SQL API. This custom PL/SQL API package specification file (zz\_sdrequest\_s.pls) can be as follows:

```

set verify off
whenever sqlerror exit failure rollback;
WHENEVER OSERROR EXIT FAILURE ROLLBACK;

create or replace package ZZ_SDREQUEST as
/* $Header: zz_sdrequest_s.pls $ */

-- Custom procedure to create single supplier ship and debit request

procedure ZZ_CREATE_SDREQUEST (
  CP_API_VERSION_NUMBER IN      NUMBER,
  CP_INIT_MSG_LIST      IN      VARCHAR2 := FND_API.G_FALSE,
  CP_COMMIT              IN      VARCHAR2 := FND_API.G_FALSE,
  CP_VALIDATION_LEVEL    IN      NUMBER := FND_API.G_VALID_LEVEL_FULL,
  CX_RETURN_STATUS      OUT     VARCHAR2,
  CX_MSG_COUNT          OUT     NUMBER,
  CX_MSG_DATA           OUT     VARCHAR2,
  CP_SDR_HDR_REC        IN      OZF_SD_REQUEST_PUB.SDR_HDR_REC_TYPE,
  CP_SDR_LINES_REC      IN      OZF_SD_REQUEST_PUB.SDR_lines_rec_type,
  CP_SDR_CUST_REC        IN      OZF_SD_REQUEST_PUB.SDR_cust_rec_type,
  CP_SDR_BILLTO_REC     IN      OZF_SD_REQUEST_PUB.SDR_cust_rec_type,
  CX_REQUEST_HEADER_ID  OUT     NUMBER
)
;
end ZZ_SDREQUEST;

/
commit;
exit;

```

Based on the PL/SQL API annotation standards, the integration developer must annotate the Supplier Ship and Debit Request custom package specification file by adding the annotation information specifically in the following places:

- Annotate the PL/SQL API package specification
- Annotate the PL/SQL procedure

The annotations for the procedure should be placed between the definition and '; '.

Please note that you only need to annotate the custom package specification file, but not the package body file.

How to annotate custom interfaces for the interface types supported by Oracle Integration Repository, see Integration Repository Annotation Standards, page A-1.

```

set verify off
whenever sqlerror exit failure rollback;
WHENEVER OSERROR EXIT FAILURE ROLLBACK;

create or replace package ZZ_SDREQUEST as
/* $Header: zz_sdrequest_s.pls $ */
/**
 * This custom PL/SQL package can be used to create supplier ship and
 * debit request for single product.
 * @rep:scope public
 * @rep:product OZF
 * @rep:displayname Single ship and debit request
 * @rep:category BUSINESS_ENTITY OZF_SSD_REQUEST
 */

-- Custom procedure to create single supplier ship and debit request

procedure ZZ_CREATE_SDREQUEST (
  CP_API_VERSION_NUMBER IN      NUMBER,
  CP_INIT_MSG_LIST IN      VARCHAR2 := FND_API.G_FALSE,
  CP_COMMIT IN      VARCHAR2 := FND_API.G_FALSE,
  CP_VALIDATION_LEVEL IN      NUMBER := FND_API.G_VALID_LEVEL_FULL,
  CX_RETURN_STATUS OUT      VARCHAR2,
  CX_MSG_COUNT OUT      NUMBER,
  CX_MSG_DATA OUT      VARCHAR2,
  CP_SDR_HDR_REC IN      OZF_SD_REQUEST_PUB.SDR_HDR_REC_TYPE,
  CP_SDR_LINES_REC IN      OZF_SD_REQUEST_PUB.SDR_lines_rec_type,
  CP_SDR_CUST_REC IN      OZF_SD_REQUEST_PUB.SDR_cust_rec_type,
  CP_SDR_BILLTO_REC IN      OZF_SD_REQUEST_PUB.SDR_cust_rec_type,
  CX_REQUEST_HEADER_ID OUT      NUMBER
)
/**
 * Use this procedure to create single supplier ship and debit request
 * @param CP_API_VERSION_NUMBER Version of the custom API
 * @param CP_INIT_MSG_LIST Flag to initialize the message stack
 * @param CP_COMMIT Indicates Flag to commit within the program
 * @param CP_VALIDATION_LEVEL Indicates the level of the validation
 * @param CX_RETURN_STATUS Indicates the status of the program
 * @param CX_MSG_COUNT Provides the number of the messages returned by
 * the program
 * @param CX_MSG_DATA Returns messages by the program
 * @param CP_SDR_HDR_REC Contains details of the new Ship Debit Request
 * to be created
 * @param CP_SDR_LINES_REC Contains the product line information for the
 * new Ship Debit Request
 * @param CP_SDR_CUST_REC Contains the Customer information for the new
 * Ship Debit Request
 * @param CP_SDR_BILLTO_REC Contains the Bill-to information for the new
 * Ship Debit Request
 * @param CX_REQUEST_HEADER_ID Returns the id of the new Ship Debit
 * Request created
 * @rep:displayname Create ship and debit request
 * @rep:category BUSINESS_ENTITY OZF_SSD_REQUEST
 * @rep:scope public
 * @rep:lifecycle active
 */
;
end ZZ_SDREQUEST;

/
commit;

```

```
exit;
```

## Generating and Uploading iLDT Files

Once annotated custom integration interface definitions are created, these annotated source files need to be validated against the annotation standards before they can be uploaded to Oracle Integration Repository. This validation is performed by executing the Integration Repository Parser (IREP Parser), a design time tool, to read the annotated files and then generate an Integration Repository loader file (iLDT ) if no error occurred.

**Note:** Please note that Integration Repository Parser does not support the integration interfaces registered under custom applications.

It is currently tested and certified for Linux, Unix, Oracle Solaris on SPARC, HP-UX Itanium, HP-UX PA-RISC, IBM AIX on Power Systems and Windows.

Once an iLDT file is generated, an integration repository administrator can upload the generated file to Oracle Integration Repository where the custom interfaces can be exposed to all users.

How to set up the Integration Repository Parser, and use it to generate and upload the iLDT file, refer to:

- Setting Up and Using Integration Repository Parser, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*
- Generating ILDT Files, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*
- Uploading ILDT Files to Integration Repository, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*

## Viewing Custom Interfaces and Performing Administrative Tasks

### Searching and Viewing Custom Interfaces

Once annotated custom interface definitions are uploaded successfully, they are merged into the interface types they belong to and displayed together with Oracle interfaces from the Integration Repository browser window. To easily distinguish annotated custom interface definitions from Oracle interfaces, the Interface Source "Custom" is used to categorize those custom integration interfaces in contrast to Interface Source "Oracle" for Oracle interfaces.

To search for custom integration interfaces, you can use either one of the following ways:

- From the Interface List page, select 'Custom' from the Interface Source drop-down

list along with a value for the Scope field to restrict the custom integration interfaces display.

- From the Search page, click **Show More Search Options** to select 'Custom' from the Interface Source drop-down list along with any interface type, product family, or scope if needed as the search criteria.

After executing the search, all matched custom integration interfaces will be displayed. For more information on how to search and view custom integration interfaces, see *Searching Custom Integration Interfaces, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide* and *Viewing Custom Integration Interfaces, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

### Performing Administrative Tasks

Once custom integration interfaces are uploaded and displayed from the Integration Repository browser interface types, all the administrative tasks are the same for the native integration interfaces. These administrative tasks including creating security grants for newly created custom interfaces if needed, generating Web services, and deploying Web services. See *Administering Custom Integration Interfaces and Services, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

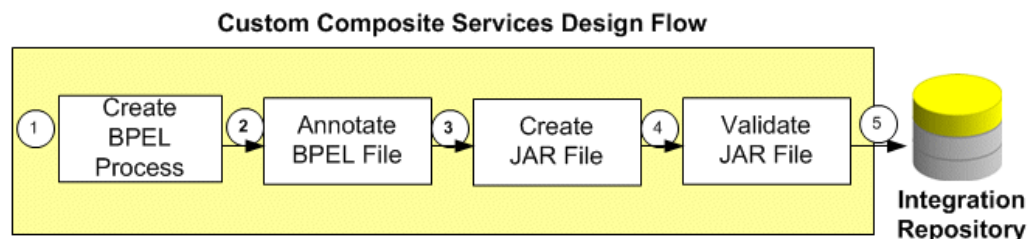
How to use custom integration interfaces as Web services to perform necessary transactions for your business needs, see a custom interface example described in the *Using Custom Integration Interfaces as Web Services*, page 10-22.

## Creating Custom Integration Interfaces of Composite Services

Integration developers can create new composite services by orchestrating discrete Web services into meaningful business processes using BPEL language. With appropriate annotation specifically for the composite services and validation against the annotation standards, the validated JAR files of the composite service BPEL projects can be uploaded to the Integration Repository.

### Creating Custom Composite Services

The following diagram illustrates the custom integration interface design flow for composite service - BPEL type:



1. A system integration developer orchestrates a composite service using a Web

service composition language BPEL.

2. The integration developer annotates the composite service based on the Integration Repository annotation standards specifically for the composite service interface type.

See: Creating and Annotating Custom Composite Services, page 10-9.

3. The integration developer creates a JAR file of the composite service BPEL project.
4. The integration repository administrator unzips the JAR file first and then validates the annotated custom interface definitions against the annotation standards specifically for composite services. This validation is performed by executing the Integration Repository Parser to read the annotated files and then generate an Integration Repository loader file (iLDT) if no error occurs.

5. An integration repository administrator uploads the generated iLDT file to Oracle Integration Repository through backend processing.

See: Generating and Uploading iLDT Files, page 10-13.

After the upload, you can search and view the uploaded custom interface from the Integration Repository user interface for verification.

Once custom integration interface definitions are uploaded and displayed from the Integration Repository browser, integration repository administrators and the integration developers can download the composite services for modification if needed. For information on how to download composite services, see Viewing and Downloading Custom Composite Services, page 10-13.

### **Custom Composite Service Annotation Example**

As mentioned earlier that the key essence of successfully creating custom integration interfaces relies on properly explanation of the new interface feature or definition. Once a custom interface definition of a specific interface type is created, an integration developer must properly annotate the custom source file based on the Integration Repository annotation standards so that the interface file of a specific interface type can be displayed with appropriate description from the browser interface.

For example, a create invoice composite service - BPEL project is created. To annotate the composite service \*.bpel file, you open the \*.bpel file in text editor and place the annotation within the comments section in the beginning of the file as highlighted below:

```

////////////////////////////////////
Oracle JDeveloper BPEL Designer

Created: Tue Oct 30 17:10:13 IST 2007
Author:  jdole
Purpose: Synchronous BPEL Process
/*#
 * This is a bpel file for creating invoice.
 * @rep:scope public
 * @rep:displayname Create Invoice
 * @rep:lifecycle active
 * @rep:product PO
 * @rep:compatibility S
 * @rep:interface oracle.apps.po.CreateInvoice
 * @rep:category BUSINESS_ENTITY INVOICE
 */

////////////////////////////////////

-->
<process name="CreateInvoice">
  targetNamespace="http://xmlns.oracle.com/CreateInvoice"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"

  xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"

  xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20"

  xmlns:ns4="http://xmlns.oracle.com/pcbpel/adapter/file/ReadPayload/"
  xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ns5="http://xmlns.oracle.com/bpel/workflow/xpath"
  xmlns:client="http://xmlns.oracle.com/CreateInvoice"

  xmlns:ns6="http://xmlns.oracle.com/bpel/services/IdentityService/xpath"
  xmlns:ora="http://schemas.oracle.com/xpath/extension"

  xmlns:ns1="http://xmlns.oracle.com/soapprovider/plsql/AR_INVOICE_API_PUB_
2108/CREATE_SINGLE_INVOICE_1037895/"

  xmlns:ns3="http://xmlns.oracle.com/soapprovider/plsql/AR_INVOICE_API_PUB_
2108/APPS/BPEL_CREATE_SINGLE_INVOICE_1037895/AR_INVOICE_API_PUB-24CREATE
_INV/"
  xmlns:ns2="http://xmlns.oracle.com/pcbpel/adapter/appscontext/"
  xmlns:bpelx="http://schemas.oracle.com/bpel/extension"

  xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc">

  <!--
  //////////////////////////////////
  PARTNERLINKS
    List of services participating in this BPEL process
  //////////////////////////////////
  -->
  <partnerLinks>
    <!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
    associated

```



with the client role are automatically set using WS-Addressing.

```
-->
<partnerLink name="client" partnerLinkType="client:CreateInvoice"
  myRole="CreateInvoiceProvider"/>
<partnerLink name="CREATE_SINGLE_INVOICE_1037895"
  partnerRole="CREATE_SINGLE_INVOICE_1037895_ptt_Role"

partnerLinkType="ns1:CREATE_SINGLE_INVOICE_1037895_ptt_PL"/>
<partnerLink name="ReadPayload" partnerRole="SynchRead_role"
  partnerLinkType="ns4:SynchRead_plt"/>
</partnerLinks>
<!--
////////////////////////////////////
VARIABLES
  List of messages and XML documents used within this BPEL process
////////////////////////////////////
-->
<variables>
<!--Reference to the message passed as input during initiation-->
  <variable name="inputVariable"
    messageType="client:CreateInvoiceRequestMessage"/>
<!--Reference to the message that will be returned to the requester-->
  <variable name="outputVariable"
    messageType="client:CreateInvoiceResponseMessage"/>
  <variable
name="Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
  messageType="ns1:Request"/>
  <variable
name="Invoke_1_CREATE_SINGLE_INVOICE_1037895_OutputVariable"
  messageType="ns1:Response"/>
  <variable name="Invoke_2_SynchRead_InputVariable"
    messageType="ns4:Empty_msg"/>
  <variable name="Invoke_2_SynchRead_OutputVariable"
    messageType="ns4:InputParameters_msg"/>
</variables>
<!--
////////////////////////////////////
ORCHESTRATION LOGIC
  Set of activities coordinating the flow of messages across the
  services integrated within this business process
////////////////////////////////////
-->
<sequence name="main">
  <!--Receive input from requestor. (Note: This maps to operation
defined in CreateInvoice.wsdl)-->
    <receive name="receiveInput" partnerLink="client"
      portType="client:CreateInvoice" operation="process"
      variable="inputVariable" createInstance="yes"/>
  <!--Generate reply to synchronous request-->
    <assign name="SetHeader">
      <copy>
        <from expression="'operations'">
          <to
variable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
          part="header"

query="/ns1:SOAHeader/ns2:ProcedureHeaderType/ns2:Username"/>
        </copy>
      <copy>
        <from expression="'Receivables, Vision Operations (USA)'">
          <to
```

```

variable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
    part="header"

query="/ns1:SOAHeader/ns2:ProcedureHeaderType/ns2:Responsibility"/>
    </copy>
    <copy>
        <from expression="'204'">
            <to
variable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
    part="header"
    query="/ns1:SOAHeader/ns2:ProcedureHeaderType/ns2:ORG_ID"/>
        </copy>
        <copy>
            <from expression="'Receivables, Vision Operations (USA)'">
                <to
variable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
    part="header"

query="/ns1:SOAHeader/ns1:SecurityHeader/ns1:ResponsibilityName"/>
    </copy>
    </assign>
    <invoke name="InvokeReadPayload" partnerLink="ReadPayload"
        portType="ns4:SynchRead_ptt" operation="SynchRead"
        inputVariable="Invoke_2_SynchRead_InputVariable"
        outputVariable="Invoke_2_SynchRead_OutputVariable"/>
    <assign name="SetPayload">
        <copy>
            <from variable="Invoke_2_SynchRead_OutputVariable"
                part="InputParameters" query="/ns3:InputParameters"/>
            Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
            part="body" query="/ns1:SOARequest/ns3:InputParameters"/>
        </copy>
    </assign>
    <assign name="SetDate">
        <copy>
            <from expression="xp20:current-date()">
                <to to
variable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
    part="body"

query="/ns1:SOARequest/ns3:InputParameters/ns3:P_TRX_HEADER_TBL/ns3:P_TR
X_HEADER_TBL_ITEM/ns3:TRX_DATE"/>
    </copy>
    </assign>
    <invoke name="Invoke_1" partnerLink="CREATE_SINGLE_INVOICE_1037895"
        portType="ns1:CREATE_SINGLE_INVOICE_1037895_ptt"
        operation="CREATE_SINGLE_INVOICE_1037895"

inputVariable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"

outputVariable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_OutputVariable"/>
    <assign name="AssignResult">
        <copy>
            <from
variable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_OutputVariable"
    part="body"

query="/ns1:SOAResponse/ns3:OutputParameters/ns3:X_MSG_DATA"/>
    <to variable="outputVariable" part="payload"
        query="/client:CreateInvoiceProcessResponse/client:result"/>
    </copy>

```

```

</assign>
  <reply name="replyOutput" partnerLink="client"
        portType="client:CreateInvoice" operation="process"
        variable="outputVariable"/>
</sequence>
</process>

```

For more information on how to annotate composite service BPEL type, see Composite Service - BPEL Annotations, page A-105.

## Generating and Uploading iLDT Files

Once annotated custom composite services are created, these annotated source files need to be validated against the annotation standards specifically for composite service - BPEL type before they can be uploaded to Oracle Integration Repository. This validation is performed by executing the Integration Repository Parser (IREP Parser), a design time tool, to read the annotated files and then generate an Integration Repository loader file (iLDT ) if no error occurred.

**Note:** Please note that Integration Repository Parser does not support the integration interfaces registered under custom applications.

It is currently tested and certified for Linux, Unix, Oracle Solaris on SPARC, HP-UX Itanium, HP-UX PA-RISC, IBM AIX on Power Systems and Windows.

Once an iLDT file is generated, an integration repository administrator can upload the generated file to Oracle Integration Repository where the custom interfaces can be exposed to all users.

How to set up the Integration Repository Parser, and use it to generate and upload the iLDT file, see:

- Setting Up and Using Integration Repository Parser, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*
- Generating ILDT Files, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*
- Uploading ILDT Files to Integration Repository, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*

## Viewing and Downloading Custom Composite Services

Once annotated custom composite service definitions are uploaded successfully, they are merged into the Composite Service BPEL type and displayed together with Oracle interfaces from the Integration Repository browser window. To easily distinguish annotated custom composite services from Oracle interfaces, the Interface Source "Custom" is used to categorize those custom interfaces in contrast to Interface Source "Oracle" for Oracle interfaces.

To search for custom composite services, from the Search page, click **Show More Search Options** to expand the search criteria. Select 'Custom' from the Interface Source drop-down list along with 'Composite Service' interface type, product family, or scope if needed as the search criteria.

After executing the search, all matched custom composite services will be displayed.

### Downloading Custom Composite Services

Similar to downloading native packaged composite services, the integration repository administrators and the integration developers can click **Download Service** in the composite service interface details page to download the relevant custom composite files aggregated in a .JAR file to your local directory.

For more information on how to search and download custom composite services, see *Downloading Composite Services, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

## Creating Custom Business Events Using Workflow XML Loader

Oracle E-Business Suite Integrated SOA Gateway allows you to create custom business events in the Business Event System and then download the events that you have created, annotate the event source codes, validate the files, and then upload the files back to the event system using Workflow XML Loader.

The Workflow XML Loader is a command line utility that lets you upload and download XML definitions for Business Event System objects between a database and a flat file. When you download Business Event System object definitions from a database, Oracle Workflow saves the definitions as an XML file. When you upload object definitions to a database, Oracle Workflow loads the definitions from the source XML file into the Business Event System tables in the database, creating new definitions or updating existing definitions as necessary.

XML files uploaded or downloaded by the Workflow XML Loader should have the extension `.wfx` to identify them as Workflow object XML definitions.

Use the following steps to create custom business events:

1. Locate and Download Business Events, page 10-14
2. Annotate the XML Definition File, page 10-17
3. Validate the Annotated Source File Using Integration Repository Parser, page 10-19
4. Upload Annotated File to the Database, page 10-20
5. Upload iLDT Files to Integration Repository, page 10-21

### Step 1: Locating and Downloading Business Events

After creating custom business events in the Oracle Workflow Business Event System,

first locate them and then download them using Workflow XML Loader.

### Locating Your Business Events

**ORACLE Administrator Workflow**

Home | Developer Studio | **Business Events** | Status Monitor | Notifications | Administration

Events | Subscriptions | Agents | Systems

Business Events: Events >

**Events**

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. An event group is a type of event composed of a set of individual member events. Event groups let you associate any events you want with each other and reference them as a group in event subscriptions.

**Search**

Enter search criteria and select the "Go" button to find your event definitions.

Name:

(Example: Entering "abc" returns "abcde" and "efgabc")

[Show More Search Options](#)

**Results: Events**

Select Event(s) and ...

[Select All](#) | [Select None](#)

Select Name	Display Name	Type	Status	Subscription	Update	Test
<input type="checkbox"/> oracle.apps.ecx.inbound.message.process	Generic Inbound Message Process Event	Event	Enabled			
<input type="checkbox"/> oracle.apps.ecx.inbound.message.receive	Generic Inbound Message Event	Event	Enabled			
<input type="checkbox"/> oracle.apps.ecx.jms.receive	Inbound Message Event for JMS queues	Event	Enabled			
<input type="checkbox"/> oracle.apps.ecx.processing.message.callback	XML Gateway Callback: Processing Event	Event	Enabled			
<input type="checkbox"/> oracle.apps.ecx.processing.message.error	XML Error Processing Event	Event	Enabled			

To download XML definitions for Business Event System objects between a database and a flat file, run the Workflow XML Loader by running Java against `oracle.apps.fnd.wf.WFXload` with the following command syntax:

```
jre oracle.apps.fnd.wf.WFXload -d{e} <user> <password> <connect string>
<protocol> <language> <xml file> <object> {<key>} {<OWNER_TAG>}
{<owner>}
```

For example, you can download either a single event or a group of events:

- Use the following command to download a single business event, such as `wfdemoe.wfx`. In the filename, first two or three chars refers to the product and the last character 'e' refers to Event.

```
java oracle.apps.fnd.wf.WFXLoad -d apps_read_only apps
hostdb:12345:sid100 thin US wfdemoe.wfx EVENTS
abc.apps.wf.bes.demo.event
```

- Use the following command to download a group of business events with wildcard:

```
java oracle.apps.fnd.wf.WFXLoad -d apps_read_only apps
hostdb:12345:sid100 thin US wfdemoe.wfx EVENTS abc.apps.wf.bes.%
```

After successfully downloading the event XML definitions, open the `.wfx` file in any text editor. You will find the content of a `wfdemoe.wfx` file, for example, containing one event shown as follows:

```

<?xml version = '1.0' encoding = 'UTF-8'?>
...

<oracle.apps.wf.event.all.sync><ExternalElement>
<OraTranslatibility>
<XlatElement Name="WF_EVENTS">
<XladID>
<Key>NAME</Key>
</XladID>
<XlatElement Name="DISPLAY_EVENTS" MaxLen="80" Expansion="50"/>
<XladID>
<Key Type="CONSTANT">DISPLAY_EVENTS</Key>
</XladID>
<XlatElement Name="DESCRIPTION" MaxLen="2000" Expansion="50"/>
<XladID>
<Key Type="CONSTANT">DESCRIPTION</Key>
</XladID>
</XlatElement>
</OraTranslatibility>
</ExternalElement>
<WF_TABLE_DATA>
  <WF_EVENTS>
    <VERSION>1.0</VERSION>
    <GUID>#NEW</GUID>
    <NAME>abc.apps.wf.demo.event</NAME>
    <TYPE>EVENT</TYPE>
    <STATUS>ENABLED</STATUS>
    <GENERATE_FUNCTION/>
    <OWNER_NAME>Oracle Workflow</OWNER_NAME>
    <OWNER_TAG>FMD</OWNER_TAG>
    <CUSTOMIZATION_LEVEL>U</CUSTOMIZATION_LEVEL>
    <LICENSED_FLAG>Y</LICENSED_FLAG>
    <JAVA_GENERATE_FUNC/>
    <DISPLAY_NAME>Demo Business Event</DISPLAY_NAME>
    <DESCRIPTION>Business event created for annotation demo.</DESCRIPTION>
    <IREP_ANNOTATION>/**
* Business event created for annotation demo.
*
* @rep:scope public
* @rep:displayname Demo Business Event
* @rep:product FND
* @rep:category BUSINESS_ENTITY
*/
</IREP_ANNOTATION>
  </WF_EVENTS>
</WF_TABLE_DATA>
</oracle.apps.wf.event.all.sync>

```

The Workflow XML Loader automatically creates a template for integration repository annotation as highlighted in bold between `<IREP_ANNOTATION>` and `</IREP_ANNOTATION>`. This is where appropriate annotations need to be placed or modified for a business event based on the business event annotation standards.

#### To download business events XML definitions:

1. Log on to Oracle Workflow page with the Workflow Administrator Web Applications responsibility. Select the Business Events link from the Navigator to open the Events page.

2. Enter search criteria in the Search region to locate your business events.
3. Change your directory to the same environment where your application is running.

For example, if your application is running on `seed100`, then change your directory to `seed100` where your business events exist.

```
/slot/ems3404/appmgr/apps/apps_st/appl  
. ./APPSseed100.env
```

4. Download the events from the database using `oracle.apps.fnd.wf.WFXload` with the following syntax:

```
jre oracle.apps.fnd.wf.WFXload -d{e} <user> <password> <connect  
string> <protocol> <language> <xml file> <object> {<key>}  
{<OWNER_TAG>} {<owner>}
```

5. Open the `.wfx` file in any text editor and notice that one business event has been placed there.

## Step 2: Annotating an XML Definition

After successfully downloading the XML definition file from a database, you should open the `.wfx` file containing one business event in any text editor and modify the annotation appropriately based on Integration Repository business event annotation standards.

The appropriate annotation includes:

- Enter meaningful description.
- Enter conditions under which the business event is raised.
- Enter UI action that invokes the business event if applicable.
- Verify scope. By default, the Workflow XML Loader annotates scope as 'public'.
- Verify display name. By default, the Workflow XML Loader uses the same display name as that mentioned in business event definition.
- Verify product. By default, the Workflow XML Loader uses Owner Tag as the Application Short Name.

Make sure that the Owner Tag corresponds to Application Short Name in `FND_APPLICATION`. Owner Name typically corresponds to Application Name but if your product is part of a larger application, you may enter an appropriate name in Owner Name.

- Enter `BUSINESS_ENTITY` code that your respective business event belongs to.
- Enter additional annotation properties if needed.

Please note that the IREP properties should not be blank. For example, the Workflow XML Loader only adds the template for Business Entity as rep:category BUSINESS\_ENTITY, page A-125, but you should add an appropriate business entity to which the event belongs. Similarly, other @rep properties cannot be left blank either.

The following is a sample business event annotation for Oracle Workflow:

```
* Business Event created to demonstrate using WFXLoad to annotate
Business Events.
*
* @rep:scope internal
* @rep:displayname Demo Business Event
* @rep:product OWF
* @rep:lifecycle active
* @rep:category BUSINESS_ENTITY WF_EVENT
*/
```

**Important:** If you decide not to annotate or publish the event in Oracle Integration Repository, you should remove the annotation only but leave the following tags unchanged. Presence of these tags is an indication that the event was reviewed for annotation.

```
<IREP_ANNOTATION/>
```

or

```
<IREP_ANNOTATION></IREP_ANNOTATION>
```

If the Loader sees these empty tags, it interprets that the business event was reviewed for annotation and it does not need to be published to the Integration Repository. Next time, when the user downloads these events, the Loader will insert empty IREP\_ANNOTATION tags as shown in the following example.

However, if you remove the entire IREP\_ANNOTATION tags for the business event and upload it, then on subsequent download the Loader will insert partially filled annotation template for the business event.



```

<WF_TABLE_DATA>
  <WF_EVENTS>
    <VERSION>1.0</VERSION>
    <GUID>#NEW</GUID>
    <NAME>oracle.apps.wf.demo.event.noannotate</NAME>
    <TYPE>EVENT</TYPE>
    <STATUS>ENABLED</STATUS>
    <GENERATE_FUNCTION/>
    <OWNER_MAME>Oracle Workflow</OWNER_MAME>
    <OWNER_TAG>FMD</OWNER_TAG>
    <CUSTOMIZATION_LEVEL>U</CUSTOMIZATION_LEVEL>
    <LICENSED_FLAG>Y</LICENSED_FLAG>
    <JAVA_GENERATE_FUNC/>
    <DISPLAY_NAME>Demo Business Event with no
annotation</DISPLAY_NAME>
    <DESCRIPTION>Business second event created for
annotation demo.</DESCRIPTION><IREP_ANNOTATION>/**
* Business event created for annotation demo.
*
* @rep:scope public
* @rep:displayname Demo Business Event
* @rep:product FND
* @rep:category BUSINESS_ENTITY
*/
</IREP_ANNOTATION>
  </WF_EVENTS>
</WF_TABLE_DATA>

```

For more information on Integration Repository Business Event Annotation Standards, see Business Event Annotations, page A-31.

### Step 3: Validating the Annotated Source File Using Integration Repository Parser

Integration Repository Parser is a standalone design time tool. It can be executed to validate the annotated custom interface definitions against the annotation standards and to generate an iLDT file if no error occurs.

After annotating the XML definition for a business event, execute the standalone Integration Repository Parser (IREP Parser) using the following command syntax to validate whether the annotation in .wfx file is valid:

Command Syntax:

```

$IAS_ORACLE_HOME/perl/bin/perl $FND_TOP/bin/irep_parser.pl -g -v
-username=<a fnd username> <product>:<relative path from product
top>:<fileName>:<version>=<Complete File Path, if not in current
directory>

```

For example:

```

$IAS_ORACLE_HOME/perl/bin/perl $FND_TOP/bin/irep_parser.pl -g -v
-username=sysadmin
owf:patch/115/xml/US:wfdemoe.wfx:12.0=../wfdemoe.wfx

```

While executing the parser, pay attention to any error messages on the console. Typically these errors would be due to incorrect annotation or some syntax errors in the annotated file. Ensure that the annotations are correct and the file has proper syntax.

If no error occurs in the annotated interface file, an iLDT (\*.ildt) file would be generated. An integration repository administrator needs to upload the generated iLDT file to the Integration Repository where the custom business events can be exposed to all users. See Step 5: Uploading iLDT Files to Integration Repository, page 10-21.

### Integration Repository Parser (irep\_parser.pl)

The `irep_parser` is a design time tool. It reads interface annotation documentation in program source files and validates it according to its file type. If the `-generate` flag is supplied (and other conditions met), then it will generate iLDT files. Any validation errors will be reported, usually along with file name and line number, like the result of `grep -n`.

Additionally, it can handle almost all types of application source files. While validating the annotated files against the annotation standards of supported interface types, if files that do not match will be ignored.

The parser will return an exit value of 0 if no errors occurred during processing. Otherwise, it will return a count of the number of files that had errors. Files with incomplete information for generation (class resolution) are considered errors only if the `-generate` flag is used.

However, before executing the Integration Repository Parser, you need to install `perl` modules and apply necessary patches. For setup information, see *Setting Up and Using Integration Repository Parser, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

For information on the Integration Repository Parser (`irep_parser.pl`) usage details including supported file types and options, files specifications, and environment, see *Integration Repository Parser (irep\_parser.pl) Usage Details, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

## Step 4: Uploading Annotated File Back to a Database

After validating the annotated source file `.wfx`, upload the file back to the database where you downloaded it earlier so that the annotated file can be stored in the appropriate tables in business event system for future references.

**Note:** To view custom business events through the Integration Repository browser window, an integration repository administrator needs to upload the generated iLDT files to the Integration Repository. For information on uploading iLDT files, see Step 5: Uploading iLDT Files to Integration Repository, page 10-21.

The Workflow XML Loader lets you upload business event system XML definitions in either normal upload mode (`-u`) or force upload mode (`-uf`):

- Normal upload mode (`-u`): If you created an event with a customization level of Core or Limit, the Workflow XML Loader will be able to update IREP\_ANNOTATION into the Business Event System WF\_EVENTS table in the

database. This normal mode will not make any updates to events or subscriptions with a customization level of User.

Use the following command to upload the annotated .wfx file back to a database:

```
java oracle.apps.fnd.wf.WFXLoad -u apps_read_only apps  
hostdb:12345:sid100 thin US wfdemoe.wfx
```

- Force upload mode (-uf): The Workflow XML Loader loads the object definitions from the source XML file into the Business Event System tables in the database and overwrites any existing definitions, even for events or subscriptions with a customization level of User.

Therefore, if you created an event with a customization level of User, use the following force upload option to make sure the IREP\_ANNOTATION can be uploaded back into the database.

```
java oracle.apps.fnd.wf.WFXLoad -uf apps_read_only apps  
hostdb:12345:sid100 thin US wfdemoe.wfx
```

For more information on how to use Workflow XML Loader, see Using the Workflow XML Loader, *Oracle Workflow Administrator's Guide*.

## Step 5: Uploading ILDT Files to Integration Repository

After the validation using the Integration Repository Parser, an iLDT file will be generated if no error occurs during the iLDT generation. In order for users to view the custom business events through the Integration Repository, an integration repository administrator needs to manually upload the generated iLDT file to the Integration Repository using FNDLOAD command.

```
$FND_TOP/bin/FNDLOAD <db_connect> 0 Y UPLOAD  
$fnd/patch/115/import/wfirep.lct <ildt file>
```

For example, FND\_TOP/bin/FNDLOAD apps/apps@instance\_name 0 Y UPLOAD  
\$FND\_TOP/patch/115/import/wfirep.lct SOAIS\_pls.ildt

For detailed information on how to upload the iLDT files, see Uploading ILDT Files to Integration Repository, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

## Viewing Custom Interfaces and Performing Administrative Tasks

### Searching and Viewing Custom Interfaces

Annotated custom interface definitions, once they are uploaded successfully, are merged into the interface types they belong to and displayed together with Oracle interfaces from the Integration Repository browser window. To easily distinguish annotated custom interface definitions from Oracle interfaces, the Interface Source "Custom" is used to categorize those custom integration interfaces in contrast to Interface Source "Oracle" for Oracle interfaces.

To search for custom integration interfaces, you can use either one of the following

ways:

- From the Interface List page, select 'Custom' from the Interface Source drop-down list along with a value for the Scope field to restrict the custom integration interfaces display.
- From the Search page, click **Show More Search Options** to select 'Custom' from the Interface Source drop-down list along with any interface type (such as 'Business Event'), product family, or scope if needed as the search criteria.

After executing the search, all matched custom integration interfaces will be displayed. For more information on how to search and view custom integration interfaces, see *Searching Custom Integration Interfaces, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide* and *Viewing Custom Integration Interfaces, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

### Performing Administrative Tasks

Once custom business events are uploaded and displayed from the Integration Repository browser interface types, all the administrative tasks are the same for the native interfaces. These administrative tasks including creating security grants for newly created custom events if needed, and subscribing to custom business events. See *Administering Custom Integration Interfaces and Services, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

How to use custom integration interfaces as Web services to perform necessary transactions for your business needs, see a custom interface example described in the *Using Custom Integration Interfaces as Web Services*, page 10-22.

## Using Custom Integration Interfaces as Web Services

### Overview

With appropriate annotation and validation, a custom integration interface can be created or built around a business entity for the interface type that Oracle Integration Repository supports. If the interface type that the custom interface belongs to can be service enabled, you can use the custom interface as a Web service to update or retrieve data from Oracle E-Business Suite or perform other business transactions over the Web.

For example, an integration developer can create a new or customized interface for Supplier Ship and Debit Request business entity using a PL/SQL API. Once the interface is uploaded to Oracle Integration Repository, it will be displayed under the PL/SQL API interface type from the Integration Repository browser. To differentiate the custom interfaces from Oracle native packaged ones, all custom integration interfaces have Interface Source 'Custom' in contrast to Oracle interfaces with Interface Source 'Oracle' when you view them from the repository.

To better understand how to use deployed custom interfaces as Web services in fulfilling your business needs, detailed design-time and run-time tasks in creating and

deploying a BPEL process are discussed in this section. For the example described in the following sections, we use Oracle JDeveloper 10.1.3.3.0 as a design-time tool to create the BPEL process and use Oracle SOA Suite BPEL server 10.1.3.3.0 for the process deployment.

## Using Custom Interface WSDL in Creating a BPEL Process at Design Time

### BPEL Process Scenario

Take a custom PL/SQL API `ZZ_SDREQUEST` as an example to explain the BPEL process creation.

When the request of creating a supplier ship and debit request is received, the request information including payload and request number will be read and passed to create a supplier ship and debit request. Once the supplier ship and debit request for a product is created, the request number will then be returned to the requestor.

After deploying the BPEL process, you should find a supplier ship and debit request is created in the Oracle E-Business Suite. The request number should be the same as the payload input value.

### Prerequisites to Create a BPEL Process Using a Custom Web Service

Before performing design-time tasks for concurrent programs, you need to ensure the following tasks are in place:

- An integration repository administrator needs to locate the custom interface and then create security grants so that the right person can access the interface.
- An integration repository administrator needs to successfully deploy the generated custom Web service to the application server.
- An integration developer needs to locate and record the deployed WSDL URL for the custom interface exposed as a Web service.

### *Creating Security Grants on the Custom Interface*

To be able to verify and use this custom interface, the administrator will first locate the custom interface (with 'Custom' interface source) from the repository, and then create security grants on the custom interface so that users with appropriate privileges can execute the interface and access the application for secured transactions.

For example, the administrator can grant the custom API access privilege to a user who has Oracle Trade Management access responsibility. After the execution of this custom API, the granted user can log on to Oracle Trade Management and verify the supplier and debit request creation details.

How to create security grants, see *Creating Grants, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

### *Deploying the WSDL URL for the Custom Interface*

An integration repository administrator must perform the following steps before letting

integration developers use the deployed WSDL in creating a BPEL process:

1. To generate a Web service, locate the interface definition first (such as a custom PL/SQL interface ZZ\_SDREQUEST) and click **Generate WSDL** in the interface details page.

Once the service is successfully generated, the Web Service - SOA Provider region appears in the interface details page. For detailed instruction on how to generate a Web service, see *Generating Web Services, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

2. To deploy a generated Web service, select at least one authentication type and click **Deploy** in the Web Service - SOA Provider region of the interface details page to deploy the service.

Once the service is successfully deployed, the selected authentication type(s) will be displayed along with 'Deployed' Web Service Status. For more information on securing Web services with authentication types, see *Managing Web Service Security, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

For detailed instruction on how to deploy a Web service, see *Deploying, Undeploying, and Redeploying Web Services, Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

### ***Searching and Recording WSDL URL***

Apart from the required tasks performed by the administrators, an integration developer also needs to log on to the system to locate and record the deployed Web service WSDL URL for the custom interface that needs to be orchestrated into a meaningful business process in Oracle JDeveloper using BPEL language.

This WSDL information will be used later in creating a partner link for the custom interface exposed as a Web service during the BPEL process creation at design time.

## Viewing and Recording a Deployed WSDL URL for the Custom Interface

The screenshot displays the Oracle Integration Repository interface. At the top, the Oracle logo and 'Integration Repository' text are visible, along with navigation links: Home, Logout, Preferences, Help, and Diagnostics. The main title is 'Integration Repository' and the specific interface is 'PLSQL Interface : Single ship and debit request'. Below this, there are buttons for 'Browse', 'Search', and 'Printable Page'. The interface details are as follows:

Internal Name	ZZ_SDREQUEST	Scope	Public
Type	PL/SQL	Interface Source	Custom
Product	Trade Management		
Status	Active		
Business Entity	<a href="#">Supplier Ship and Debit Request</a>		

**Full Description**  
This custom PL/SQL package can be used to create supplier ship and debit request for single product.

**Web Service - SOA Provider**  
Web Service Status: **Deployed**  
WSDL: [View WSDL](#)  
\* Authentication Type: ☐ Username Token ☒ SAML Token (Sender Vouches)

**Source Information**  
Source File: patch/115/sql/US/zz\_sdrequest\_s.pls  
Source Version: 12.0  
Source Product: OZF

**Procedures and Functions**

Name	Internal Name	Status	Description
<a href="#">Create ship and debit request</a>	ZZ_CREATE_SDREQUEST	Active	Use this procedure to create single supplier ship and debit request.

At the bottom, there are buttons for 'Browse', 'Search', and 'Printable Page'.

How to search for an interface and review the interface details, see Searching and Viewing Integration Interfaces, page 2-1.

### BPEL Process Creation Flow

Based on the supplier and debit request creation scenario, the following design-time tasks are discussed in this chapter:

1. Create a new BPEL project, page 10-26  
Use this step to create a new BPEL project called `ZZ_CreateSingle_ShipDebitRequest.bpel` using an Synchronous BPEL Process template. This automatically creates two dummy activities - Receive and Reply - to receive input from a third party application and to reply output of the BPEL process back to the request application.
2. Create a Partner Link, page 10-28  
Use this step to create an invoice in Oracle E-Business Suite by using the Single Ship and Debit Request custom API `ZZ_SDREQUEST` exposed as Web service.
3. Add a Partner Link for File Adapter, page 10-31  
Use this step to synchronous read input data details passed from the first Assign activity to create supplier ship and debit request.
4. Add Invoke activities, page 10-36

Use this step to configure two Invoke activities in order to:

- Point to the File Adapter to synchronous read input data details that is passed from the first Assign activity.
- Point to the ZZ\_SDREQUEST partner link to initiate the supplier ship and debit request creation with payload and request number details received from the Assign activities.

5. Add Assign activities, page 10-39

Use this step to configure Assign activities in order to pass application context header variables, payload information and request number to appropriate Invoke activities to facilitate the single supplier ship and debit request creation. At the end, pass the request number to the request application through the dummy Reply activity.

For general information and basic concept of a BPEL process, see Understanding BPEL Business Processes, page D-1 and *Oracle BPEL Process Manager Developer's Guide* for details.

## Creating a New BPEL Project

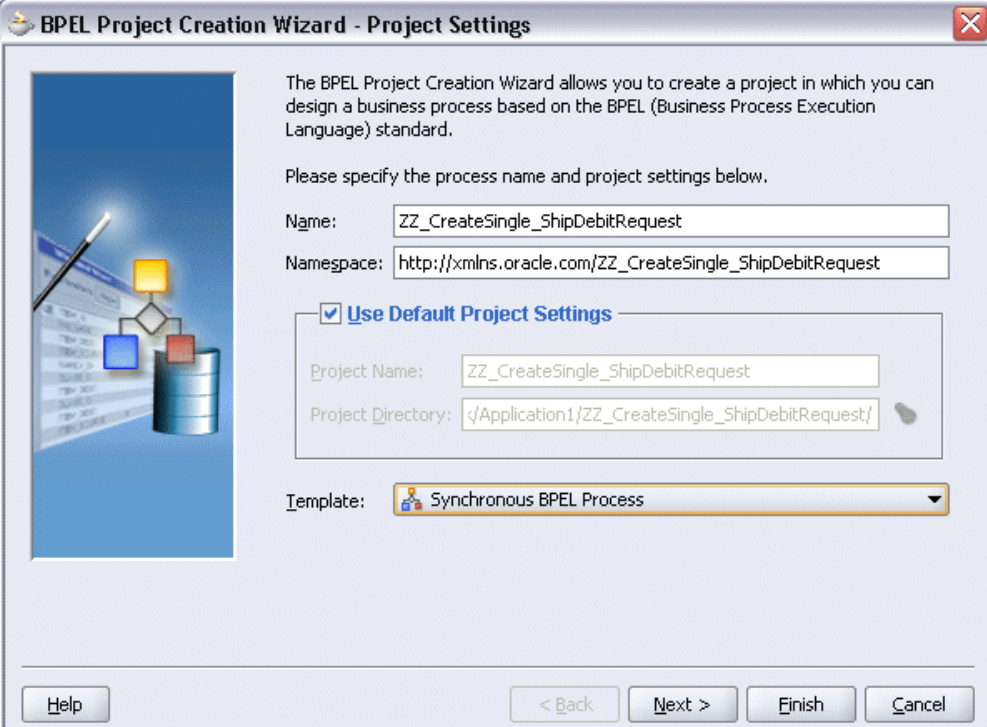
Use this step to create a new BPEL project that will contain various BPEL process activities.

**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 produces a list of available categories.
4. Expand the **General** node and select **Projects**.
5. Select **BPEL Process Project** from the **Items** group.
6. Click **OK**. The BPEL Process Project dialog box appears.



### Entering BPEL Project Information



The BPEL Project Creation Wizard allows you to create a project in which you can design a business process based on the BPEL (Business Process Execution Language) standard.

Please specify the process name and project settings below.

Name:

Namespace:

☒ **Use Default Project Settings**

Project Name:

Project Directory:

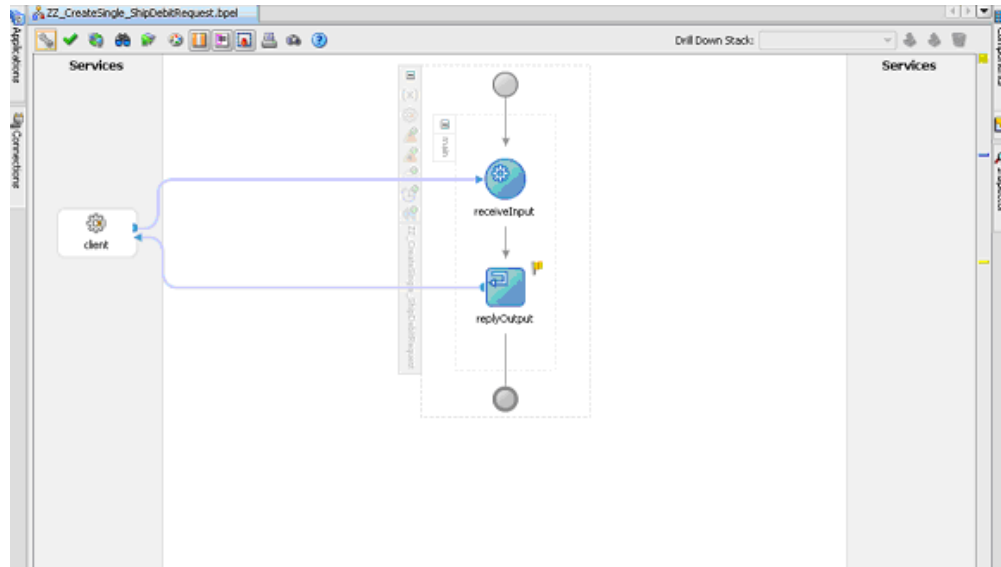
Template:

Help < Back Next > Finish Cancel

7. In the **Name** field, enter a descriptive name such as `ZZ_CreateSingle_ShipDebitRequest`.
8. From the Template list, select **Synchronous BPEL Process** and then select **Use Default Project Settings**.
9. Use the default input and output schema elements in the Input/Output Elements dialog box.
10. Click **Finish**.

A new synchronous BPEL process is created with the Receive and Reply activities. The required source files including `bpel.xml`, using the name you specified (for example, `ZZ_CreateSingle_ShipDebitRequest.bpel`) are also generated.

### New BPEL Process Diagram



### Creating a Partner Link for the Web Service

Use this step to create a Partner Link called `ZZ_CreateSD_Request`.

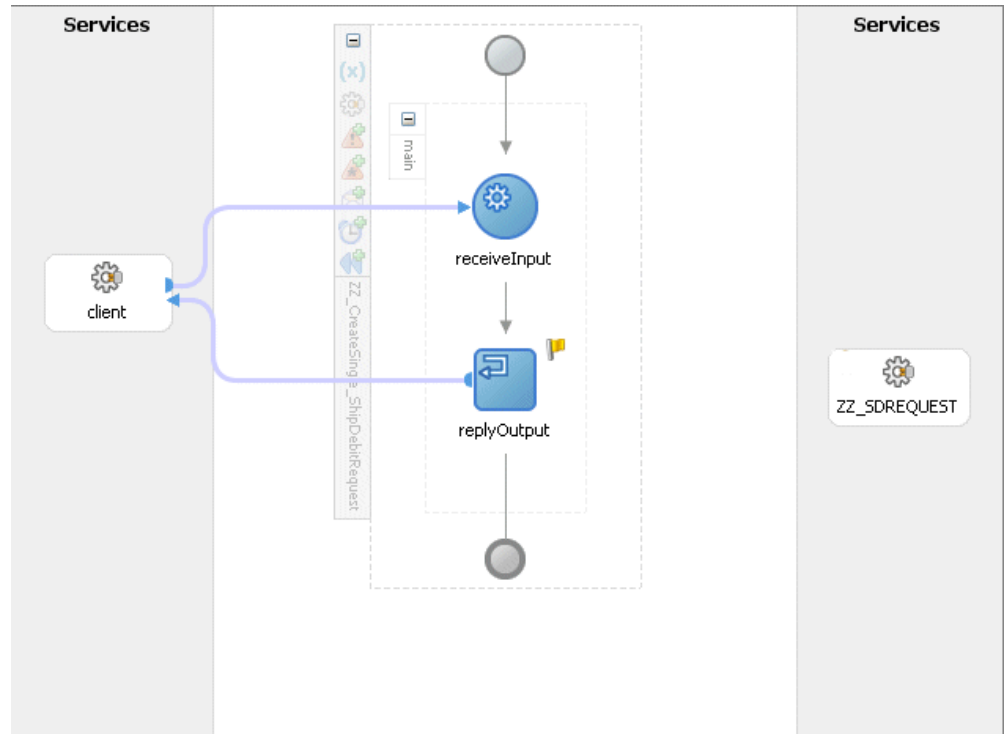
**To create a partner link for Single Ship and Debit Request Web service:**

1. In JDeveloper BPEL Designer, drag and drop the **PartnerLink** service from the Component Palette into the Partner Link border area of the process diagram. The Service Name dialog box appears.
2. Copy the WSDL URL corresponding to the custom service, `ZZ_SDREQUEST`, that you recorded earlier in the WSDL File field.
3. A Partner Link Type message dialog box appears asking whether you want the system to create a new WSDL file that will by default create partner link types for you.

Click **Yes** to have the Partner Name value populated automatically.

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.

#### Adding the Partner Link



4. You can optionally change the default partner link name by double-clicking the icon to open the Edit Partner Link window if you like.  
Select the Partner Role value from the drop-down list.  
Click **Apply**.

### Editing the Partner Link Parameters

**Create Partner Link**

General Image Property

Name: ZZ\_SDREQUEST

Process: ZZ\_CreateSingle\_ShipDebitRequest

**WSDL Settings**

WSDL File: eSingle\_ShipDebitRequest/bpel/ZZ\_SDREQUEST.wsdl

Partner Link Type: ZZ\_SDREQUEST\_PortType\_PL

Partner Role: ZZ\_SDREQUEST\_PortType\_Role

My Role: ZZ\_SDREQUEST\_PortType\_Role

Help Apply OK Cancel

5. Select the Property tab and click the **Create Property** icon to select the following properties from the property name drop-down list in order to pass the security header along with the SOAP request:

- wsseUsername

Specify the username such as `trademgr` to be passed in the Property Value box.

- wssePassword

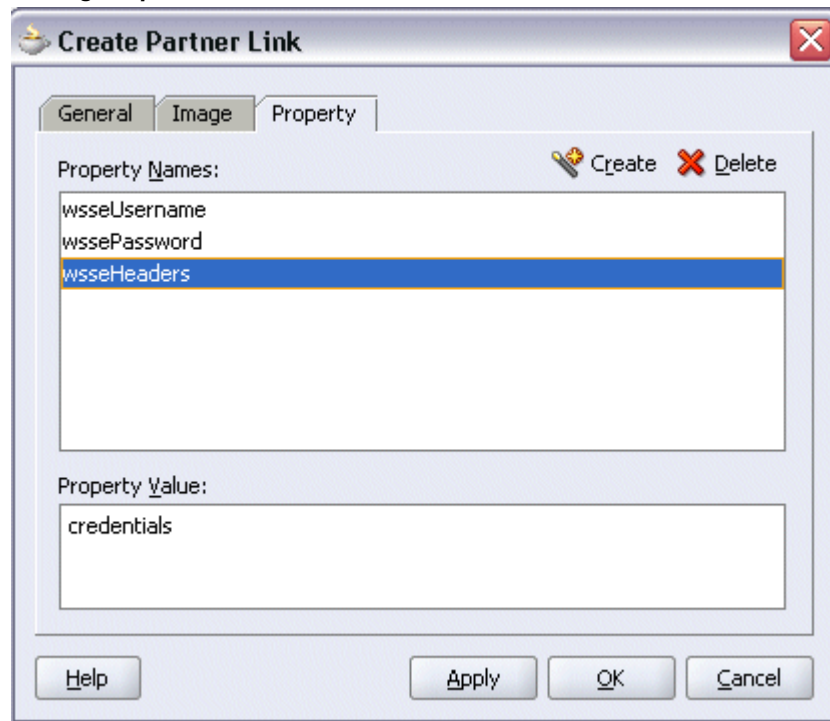
Specify the corresponding password such as `welcome` for the username to be passed in the Property Value box.

- wsseHeaders

Enter `credentials` as the property value.

Click **Apply** to save the selected property values.

### Adding Properties



6. Click **OK** to complete the partner link configuration.

### Adding a Partner Link for File Adapter

Use this step to configure a BPEL process to read input payload.

#### To add a Partner Link for File Adapter to Read Payload:

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 welcome page appears.
2. Click **Next**. The Service Name dialog box appears.
3. Enter a name for the file adapter service such as `Read_Payload`. You can add an optional description of the service.
4. Click **Next**. The Operation dialog box appears.

### Specifying the Operation



The File Adapter supports three operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, and a Synchronous Read File operation that reads the current contents of a file. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type:

- ☐ Read File
- ☐ Write File
- ☒ Synchronous Read File

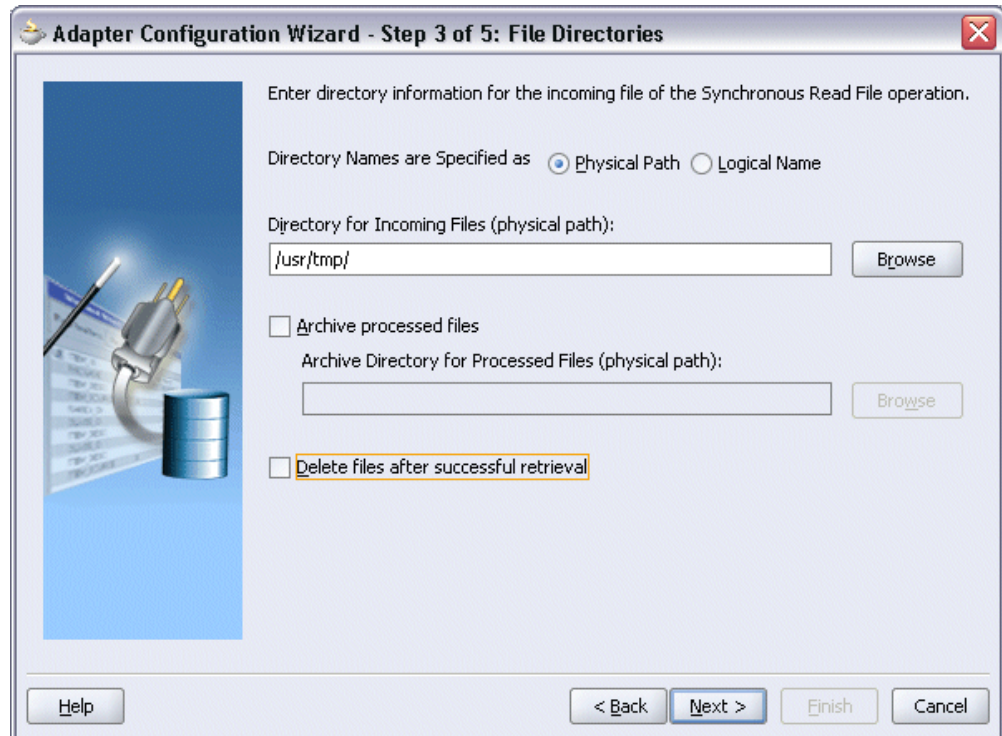
Operation Name:

Help < Back Next > Finish Cancel

5. Specify the operation type, for example **Synchronous Read File**. This automatically populates the **Operation Name** field.

Click **Next** to access the File Directories dialog box.

### Configuring the Input File



6. Select the **Physical Path** radio button and enter the physical path for incoming file directory information. For example, enter `/usr/tmp/`.

**Note:** To be able to locate the file from the physical directory you specified here, you must first place the input payload file (such as `Inputzzsdrequest.xml`) to the specified directory.

Alternatively, click **Browse** to locate the incoming file directory information.

Uncheck the **Delete Files after successful retrieval** check box. Click **Next** to open the File Name dialog box.

7. Enter the name of the file for the synchronous read file operation. For example, enter `Inputzzsdrequest.xml`. Click **Next**. The Messages dialog box appears.
8. Select **Browse** in the Schema Location field to open the Type Chooser window.  
Click **Import Schema File...** icon on the top right corner of the Type Chooser window.
9. Enter the schema location for the custom service, such as  
`http://<host>:<port>/webservices/SOAPProvider/plsql/zz_sdrequest/APPS_XX_BPEL_ZZ_CREATE_SDREQUEST_RE_ZZ_SDREQUEST_ZZ_CREATE`



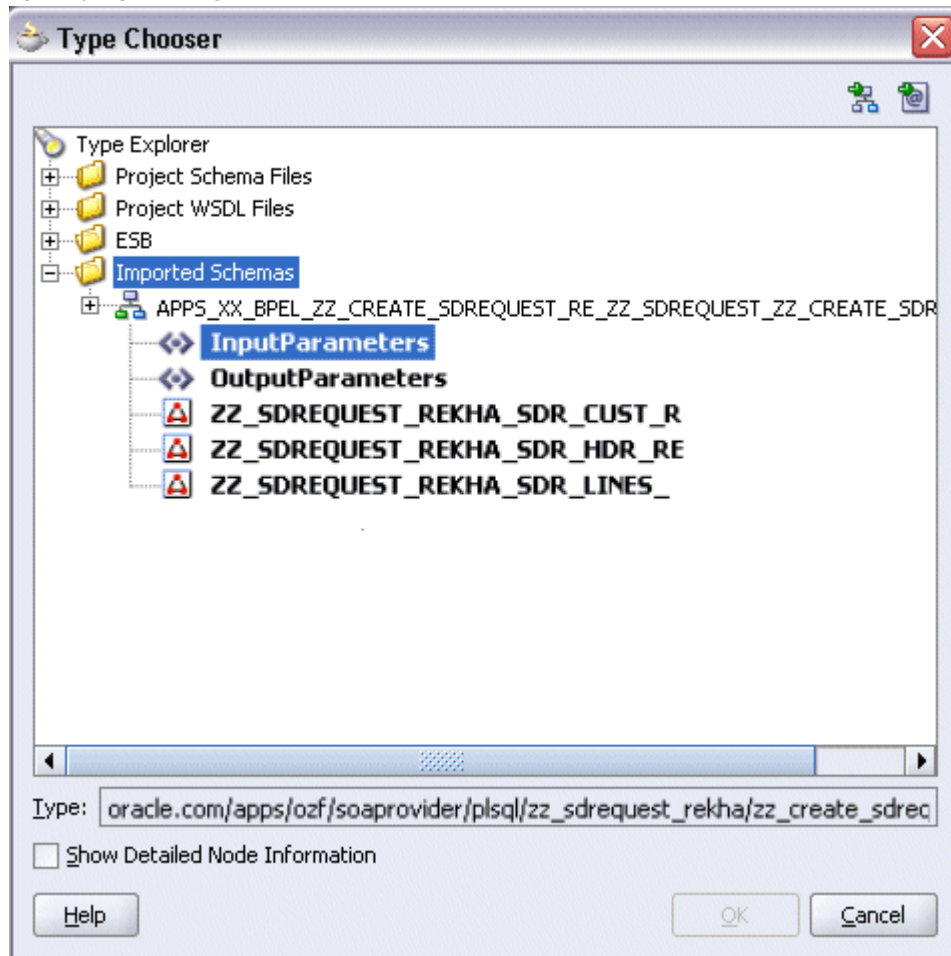
\_SDREQU.xsd.

The schema location for your custom service can be found from the custom service WSDL (for example,  
`http://<host>:<port>/webservices/SOAPProvider/plsql/zz_sdrequest/?wsdl`).

Select the 'Add to Project' check box and click **OK**.

10. Click **OK** to import schema prompt. The imported schema section will be added to the Type Choose window.

#### Specifying Message Schema



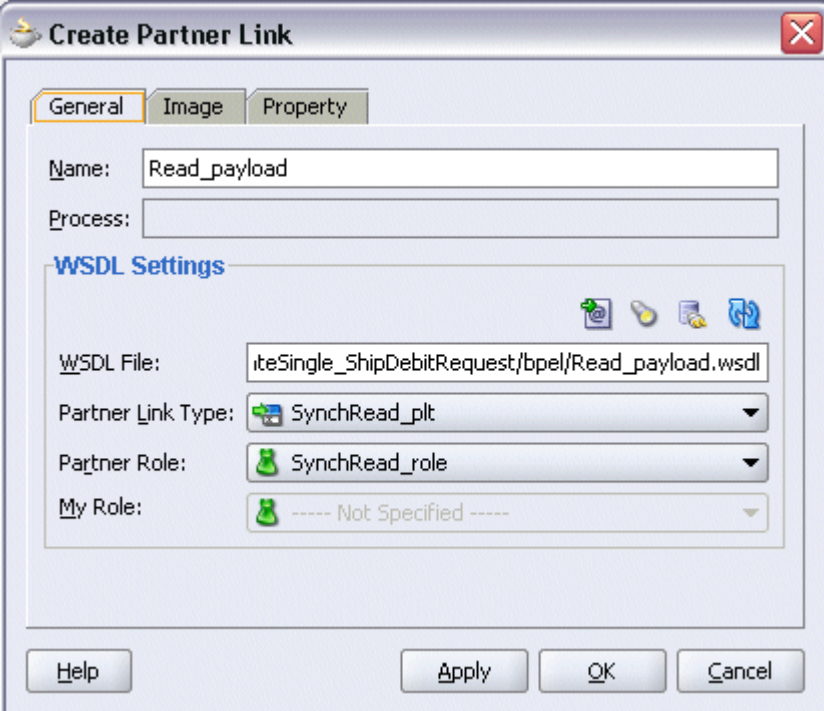
11. Browse the imported schema by selecting Imported Schemas > APPS\_XX\_BPEL\_ZZ\_CREATE\_SDREQUEST\_RE\_ZZ\_SDREQUEST\_ZZ\_CREATE\_SDREQU.xsd > InputParameters.

Click **OK**. The selected .xsd is displayed as Schema Location, and InputParameters is displayed as Schema Element.



12. Click **Next** and 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 `Read_Payload.wsdl`.

#### Completing the Partner Link Configuration



The image shows a 'Create Partner Link' dialog box with three tabs: 'General', 'Image', and 'Property'. The 'General' tab is selected. It contains the following fields and controls:

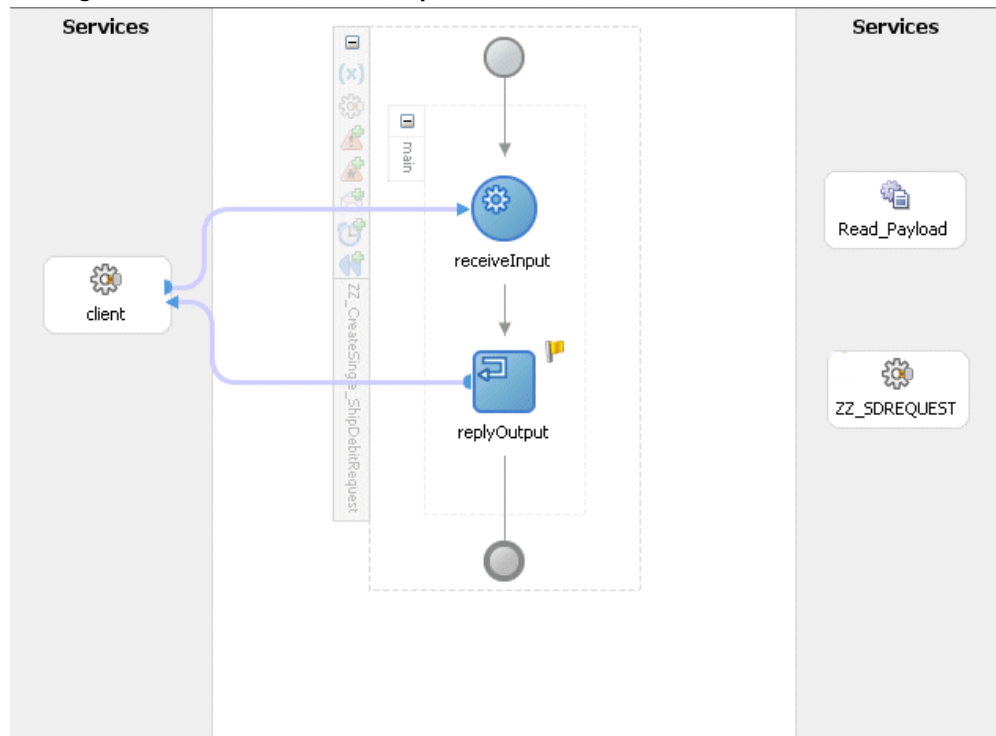
- Name:** A text box containing 'Read\_payload'.
- Process:** An empty text box.
- WSDL Settings:** A section with a title bar and a toolbar containing icons for file operations (new, open, save, print, etc.).
  - WSDL File:** A text box containing 'iteSingle\_ShipDebitRequest/bpel/Read\_payload.wsdl'.
  - Partner Link Type:** A dropdown menu showing 'SynchRead\_plt'.
  - Partner Role:** A dropdown menu showing 'SynchRead\_role'.
  - My Role:** A dropdown menu showing '---- Not Specified ----'.

At the bottom of the dialog are four buttons: 'Help', 'Apply', 'OK', and 'Cancel'.

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

The `Read_Payload` Partner Link appears in the BPEL process diagram:

### Adding the Partner Link for File Adapter



### Adding Invoke Activities

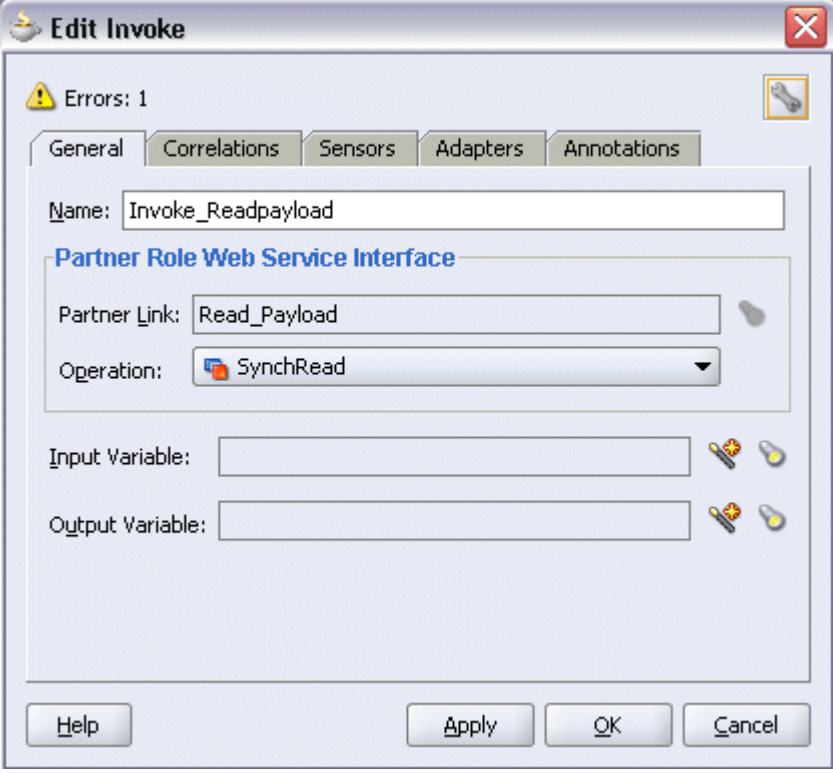
This step is to configure two Invoke activities:

- Read supplier ship and debit request creation details that is passed from the first Assign activity using Read\_Payload partner link for File Adapter.
- Send the payload and request number details received from the Assign activities to create a single supplier ship and debit request by using the ZZ\_SDREQUEST partner link.

#### To add an Invoke activity for Read\_Payload Partner Link:

1. In JDeveloper BPEL Designer, drag and drop the **Invoke** activity from the Component Palette into the Activity box of the process diagram, between the **Receive** and **Reply** activities.
2. Link the Invoke activity to the Read\_Payload service. The Invoke activity will send request data to the partner link. The Edit Invoke dialog box appears.

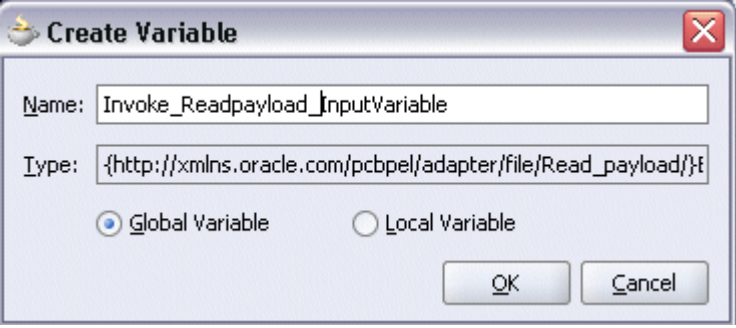
### Editing the Invoke Activity



The **Edit Invoke** dialog box is shown with the **General** tab selected. It features a **Name** field containing 'Invoke\_Readpayload'. Below this is a section titled **Partner Role Web Service Interface** with a **Partner Link** field containing 'Read\_Payload' and an **Operation** dropdown menu set to 'SynchRead'. At the bottom of this section are **Input Variable** and **Output Variable** fields, each with a **Create** icon (a lightbulb with a plus sign) to its right. The dialog also includes an **Errors: 1** indicator at the top left, a **Help** button at the bottom left, and **Apply**, **OK**, and **Cancel** buttons at the bottom right.

3. Enter a name for the Invoke activity such as 'Invoke\_Readpayload', and 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



The **Create Variable** dialog box is shown. It has a **Name** field containing 'Invoke\_Readpayload\_InputVariable' and a **Type** field containing '{http://xmlns.oracle.com/pcbpel/adapter/file/Read\_payload/}t'. Below these fields are two radio buttons: **Global Variable** (which is selected) and **Local Variable**. At the bottom right are **OK** and **Cancel** buttons.

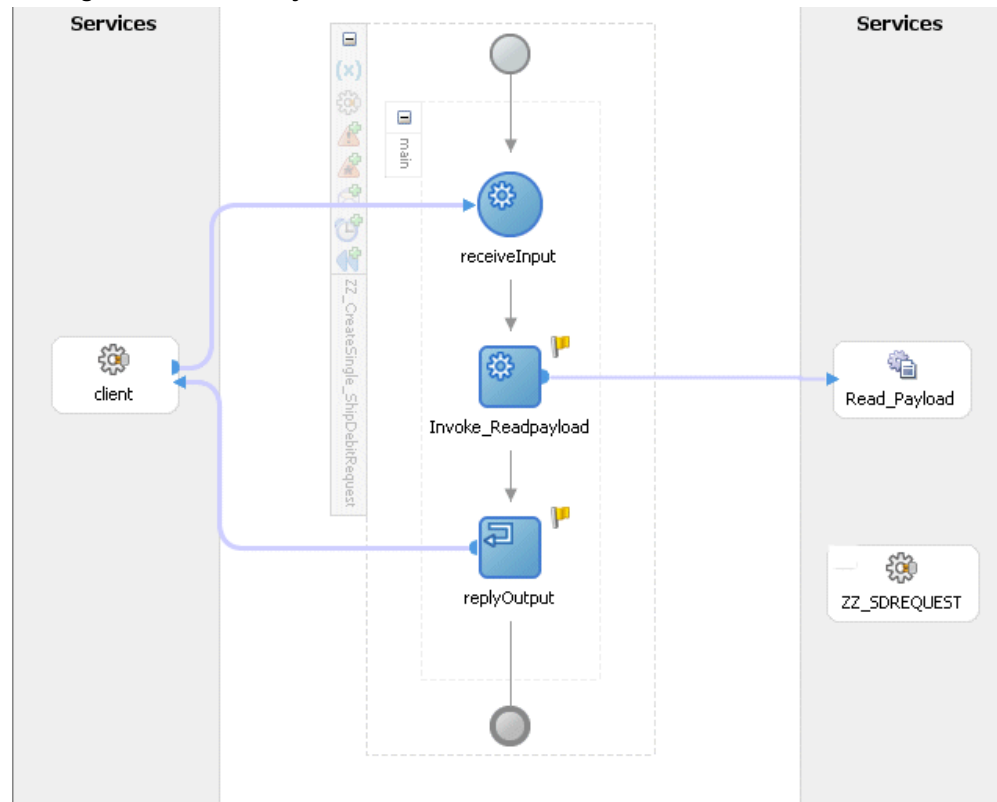
4. Enter a name for the variable such as 'Invoke\_Readpayload\_InputVariable' and select **Global Variable**. Click **OK** in the Create Variable dialog box.  
Enter a name for the output variable such as 'Invoke\_Readpayload\_OutputVariable'

and select **Global Variable**. Click **OK** in the Create Variable dialog box.

Click **Apply** and **OK** in the Edit Invoke dialog box to finish configuring the Invoke activity.

The Invoke activity appears in the process diagram.

#### **Adding an Invoke Activity**



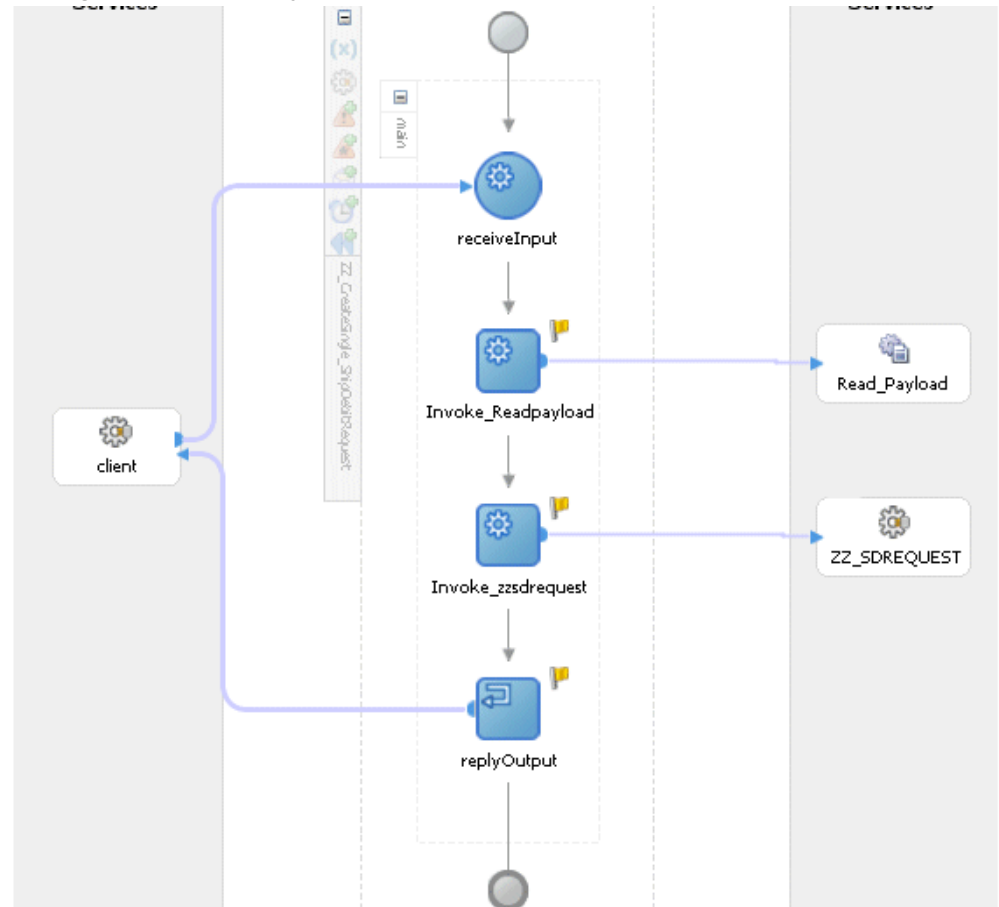
#### **To add an Invoke activity for ZZ\_SDREQUEST Partner Link:**

1. In JDeveloper BPEL Designer, drag and drop the **Invoke** activity from the Component Palette into the Activity box of the process diagram, after the **Invoke** and **Reply** activities.
2. Link the Invoke activity to the ZZ\_SDREQUEST service. The Invoke activity will send the request number to the partner link. The Edit Invoke dialog box appears.
3. Enter a name for the Invoke activity such as 'Invoke\_zzsdrequest'.  
Select the Operation as ZZ\_CREATE\_SDREQUEST.
4. Click the **Create** icon next to the **Input Variable** field to create a new variable such as 'Invoke\_zzsdrequest\_InputVariable'. Select **Global Variable** and click **OK** in the Create Variable dialog box.

5. Click the **Create** icon next to the **Output Variable** field to create a new variable such as 'Invoke\_zzsdrequest\_OutVariable'. Select **Global Variable** and click **OK** in the Create Variable dialog box. Click **Apply** and **OK** in the Edit Invoke dialog box to complete the Invoke activity creation.

The Invoke activity appears in the process diagram.

#### Adding an Invoke Activity



#### Adding Assign Activities

This step is to configure four Assign activities:

1. To set the applications context information obtained from the dummy Receive activity, that will be used in passing variables for SOAHeader elements of the SOAP request.

**Note:** You need to populate certain variables in the BPEL process for SOAHeader elements to pass values that may be required to set

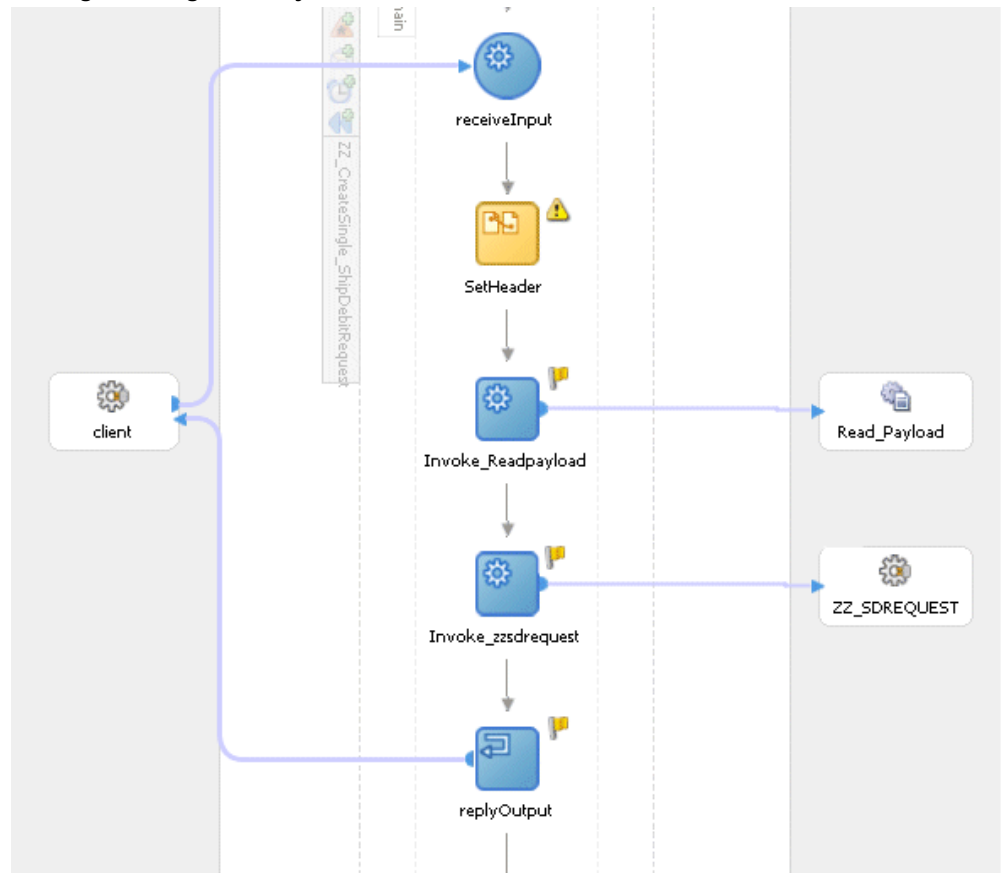
applications context during service execution. These SOAHeader elements are Responsibility, RespApplication, SecurityGroup, NLSLanguage, and Org\_Id.

2. To pass the payload information to the `Invoke_zzsdrequest` Invoke activity.
3. To pass the supplier ship and debit request number information to the `Invoke_zzsdrequest` Invoke activity.
4. To pass the supplier ship and debit request number information back to the dummy Reply activity as an output.

**To add the first Assign activity to pass applications context details to the `Invoke_Readpayload` Invoke 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 first **Invoke** activity.

### Adding an Assign Activity



2. Double-click the **Assign** activity to access the Edit Assign dialog box.
3. Click the General tab to enter the name for the Assign activity, such as 'SetHeader'.
4. On the Copy Operation tab, click **Create** and then select **Copy Operation** from the menu. The Create Copy Operation window appears.
5. Enter the first pair of parameters:
  - In the From navigation tree, select type Expression and then enter '204' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Invoke\_zzsdrequest\_InputVariable > header > ns5:SOAHeader** and select **ns5:ORG\_ID**. The XPath field should contain your selected entry.
  - Click **OK**.

6. Enter the second pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
  - In the From navigation tree, select type Expression and then enter 'TRADE\_MANAGEMENT\_USER' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Invoke\_zzsdrequest\_InputVariable > header > ns5:SOAHeader** and select **ns5:Responsibility**. The XPath field should contain your selected entry.
  - Click **OK**.
7. Enter the third pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
  - In the From navigation tree, select type Expression and then enter 'OZF' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Invoke\_zzsdrequest\_InputVariable > header > ns5:SOAHeader** and select **ns5:RespApplication**. The XPath field should contain your selected entry.
  - Click **OK**.
8. Enter the fourth pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
  - In the From navigation tree, select type Expression and then enter 'STANDARD' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Invoke\_zzsdrequest\_InputVariable > header > ns5:SOAHeader** and select **ns5:SecurityGroup**. The XPath field should contain your selected entry.
  - Click **OK**.
9. Enter the fifth pair of parameters by selecting **Copy Operation** from the Create drop-down list with the following values:
  - In the From navigation tree, select type Expression and then enter 'AMERICAN' in the Expression box.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Invoke\_zzsdrequest\_InputVariable > header > ns5:SOAHeader**



and select **ns5:NLSLanguage**. The XPath field should contain your selected entry.

- Click **OK**.

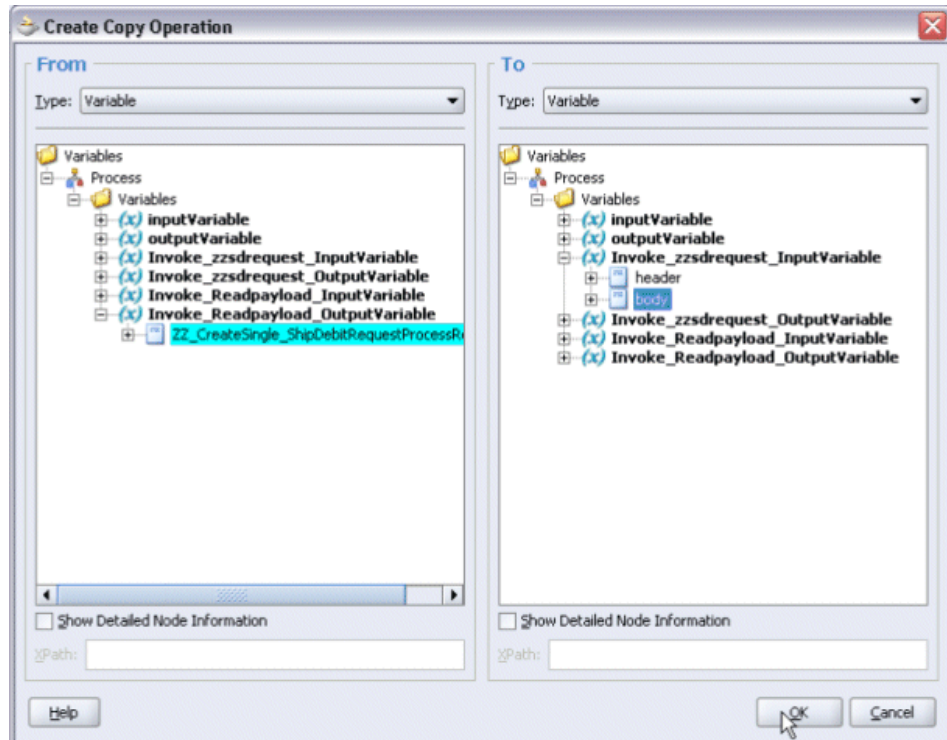
10. The Edit Assign dialog box appears.

11. Click **OK** to complete the configuration of the Assign activity.

**To enter the second Assign activity to pass payload information to the Invoke\_zzsdrequest Invoke activity:**

1. Add the second Assign activity by dragging and dropping the **Assign** activity from the Component Palette into the Activity box of the process diagram, between two **Invoke** activities.
2. Repeat Step 2 to Step 4 described in creating the first Assign activity to add the second Assign activity called 'SetPayload'.
3. Enter the following information:
  - In the From navigation tree, navigate to **Variable > Process > Variables > Invoke\_ReadPayload\_OutVariable > ZZ\_CreateSingle\_ShipDebitRequestProcessRequest**.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > Invoke\_zzsdrequest\_InputVariable > Body**.

### Assign Parameters



Click **OK** in the Create Copy Operation window.

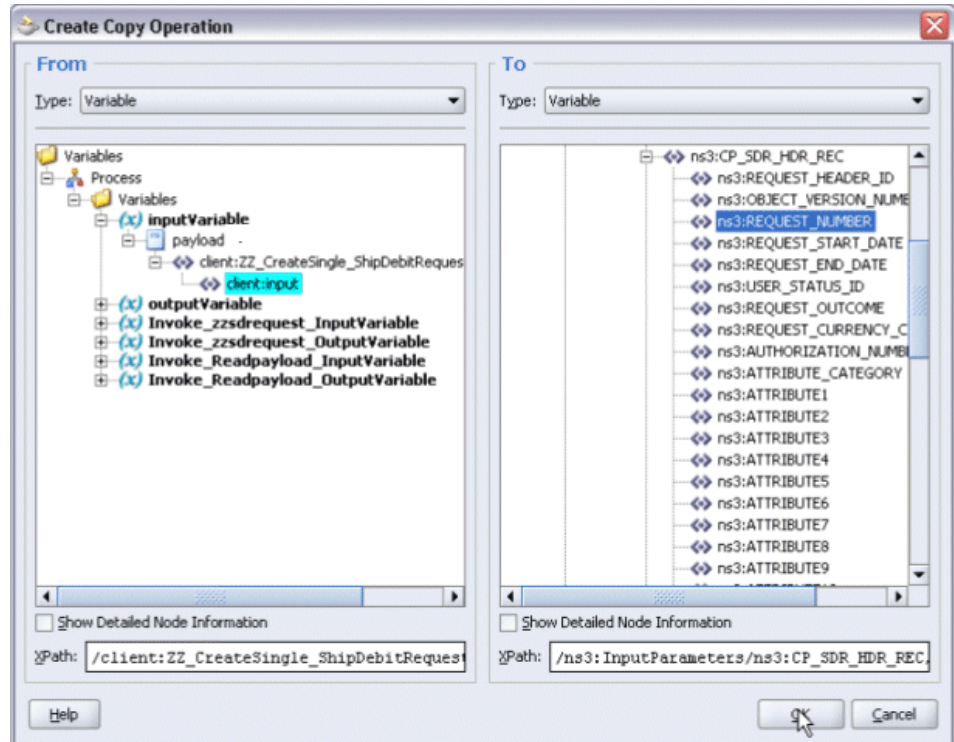
4. Click **OK** to complete the configuration of the Assign activity.

**To enter the third Assign activity to pass the supplier ship and debit request number to the Invoke\_zzsdrequest Invoke activity:**

1. Add the third Assign activity by dragging and dropping the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the second **Assign** activity and the Invoke\_zzsdrequest **Invoke** activity.
2. Repeat Step 2 to Step 4 described in creating the first Assign activity to add the third Assign activity called 'SetRequestNumber'.
3. Enter the following information:
  - In the From navigation tree, navigate to **Variable > Process > Variables > inputVariable > Payload > client:ZZ\_CreateSingle\_ShipDebitRequestProcessRequest > client:input**. The XPath field should contain your selected entry.
  - In the To navigation tree, select type Variable. Navigate to **Variable > Process >**

Variables > Invoke\_zzsdrequest\_InputVariable > Body > ns3:InputParameters > ns3:CP\_SDR\_HDR\_REC > ns3:REQUEST\_NUMBER and select ns3:TRX\_NUMBER. The XPath field should contain your selected entry.

### Assigning Parameters



- Click **OK** in the Create Copy Operation window.
4. Click **OK** in the Assign window to complete the configuration of the Assign activity.

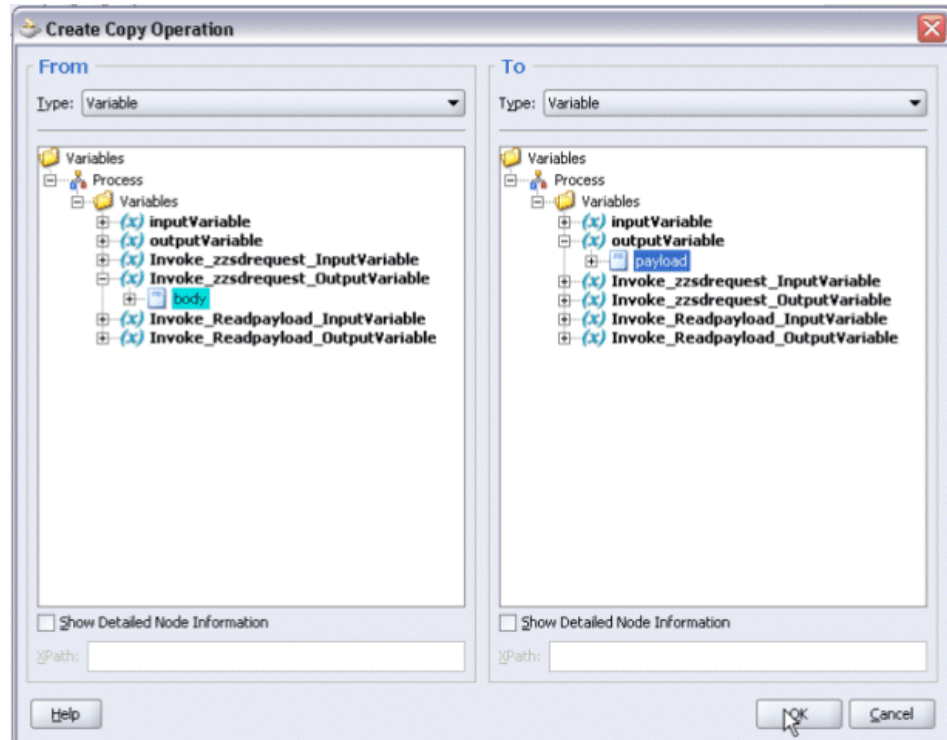
**To add the fourth Assign activity to reply back supplier ship and debit request number:**

1. Add the third Assign activity by dragging and dropping the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the Invoke\_zzsdrequest **Invoke** and the **Reply** activities.
2. Repeat Step 2 to Step 4 described in creating the first Assign activity to add the fourth Assign activity called 'SetRequestNumber'.
3. Enter the following information:
  - In the From navigation tree, select type Variable. Navigate to **Variable** >

Process > Variables > Invoke\_zzsdrequest\_OutputVariable > Body.

- In the To navigation tree, select type Variable. Navigate to **Variable > Process > Variables > outputVariable > payload**.

#### Assign Parameters



4. Click **OK** in the Create Copy Operation window.
5. Click **OK** in the Assign window to complete the configuration of the Assign activity.

## Deploying and Testing the BPEL Process at Run Time

After creating a BPEL process using the WSDL URL generated from a custom PL/SQL interface definition, you can deploy it to a BPEL server if needed. To ensure that this process is modified or orchestrated appropriately, you can also manually test the BPEL process by initiating the business process contained in the BPEL process to test the interface integration.

### Prerequisites

Before deploying the BPEL process using Oracle JDeveloper, you must ensure that you have established the connectivity between the design-time environment and the run-time servers including the application server and the integration server.

How to configure the necessary server connection, see Configuring Server Connection, page B-1.

To validate your BPEL process, perform the following run-time tasks:

1. Deploy the BPEL process, page 10-47

Once you deploy the process to a BPEL server, it becomes available so that you can run the process manually to test it for validation.

2. Test the BPEL process, page 10-49

After deploying a BPEL process, you can manage the process from the BPEL console to manually initiate the business process and test the interface integration contained in your BPEL process.

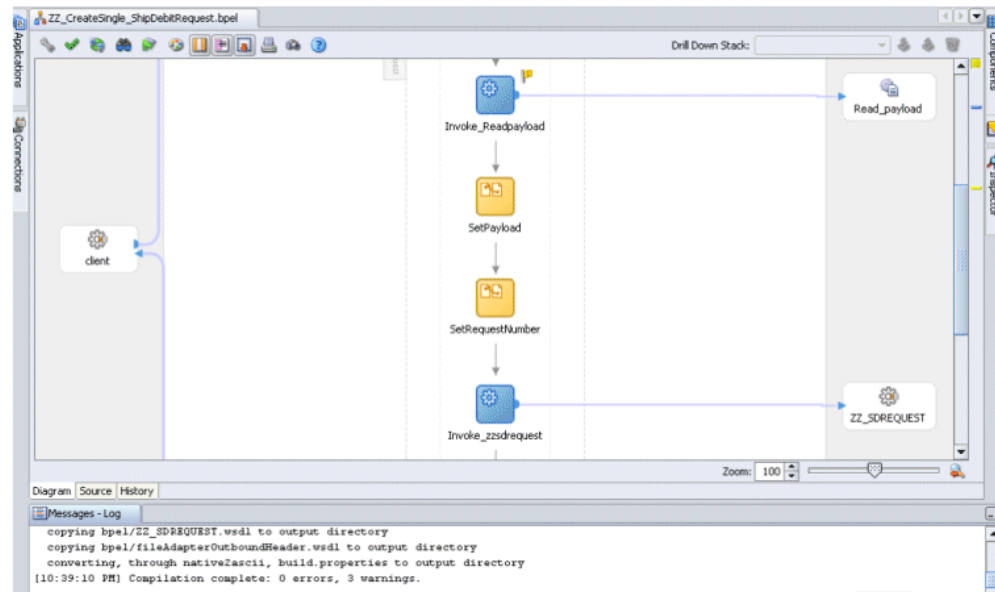
## Deploying the BPEL Process

You must deploy the Create Single Supplier Ship and Debit Request BPEL process ( `ZZ_CreateSingle_ShipDebitRequest.bpel`) that you created earlier before you can run it.

**To deploy the BPEL process:**

1. In the Applications Navigator of JDeveloper BPEL Designer, select the **ZZ\_CreateSingle\_ShipDebitRequest** project.
2. Right-click the project and select **Make** action from the menu to ensure the successful server connections.

### Validation of Server Connections

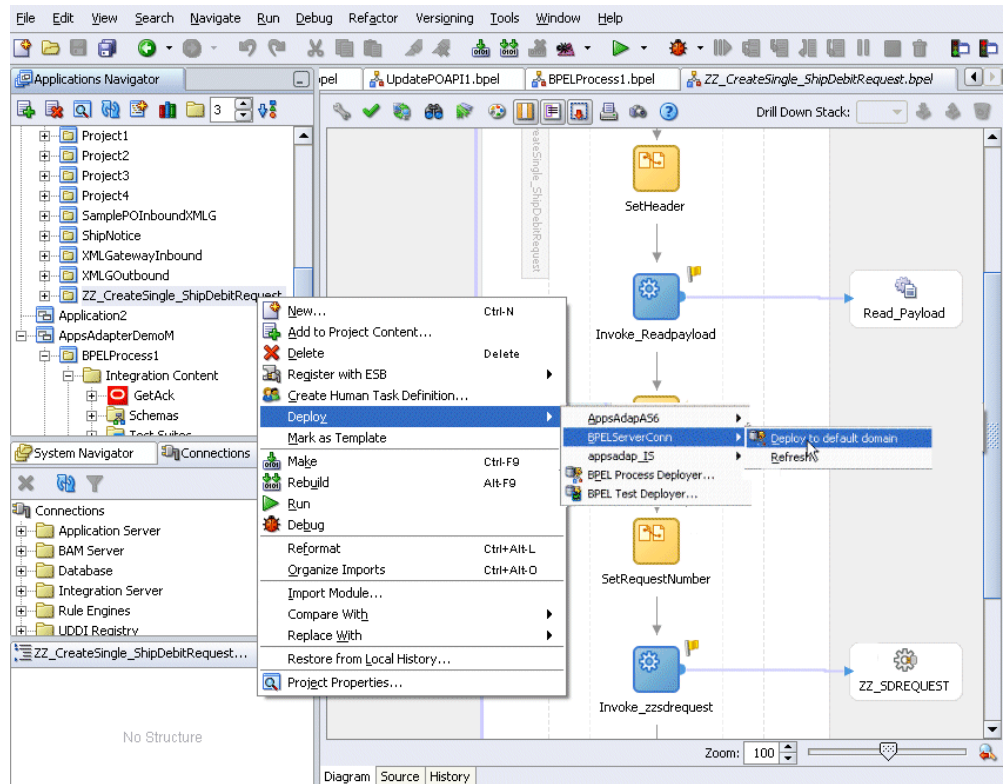


You can look for any compilation error messages in Messages Log.

3. Right-click the project and select **Deploy** action from the menu. Click on **IntegrationServerConnection name** and enter your BPEL Process Manager information.

For example, you can select **Deploy > BPELServerConn > Deploy to Default Domain** to deploy the process if you have the BPEL Process Manager set up appropriately.

## Deploying the BPEL Process



### 4. The Password Prompt dialog box appears.

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

The BPEL project is compiled and successfully deployed.

## Testing the BPEL Process

To validate whether the BPEL process that you created works or not, you need to manually initiate the process after it has been successfully deployed to the BPEL server. Therefore, the validation starts with the BPEL console to ensure that you can find the deployed BPEL process listed in the console. Then, you can log on to Oracle E-Business Suite to validate that the supplier ship and debit request is successfully created with the request number you specified.

### To test the BPEL process:

1. Log into Oracle Application Server 10g BPEL Console (<http://<soaSuiteServerHostName>:<port>/BPELConsole>). The BPEL Console login page appears.

2. Enter the username and password and click **Login**.

The Oracle Enterprise Manager 10g BPEL Control appears.

3. In the BPEL Console, confirm that ZZ\_CreateSingle\_ShipDebitRequest has been deployed.

### Deployed BPEL Processes

ORACLE® Enterprise Manager 10g  
BPEL Control

Manage BPEL Domain | Logout | Support  
Logged to domain: default

Dashboard | BPEL Processes | Instances | Activities

Deployed BPEL Processes		In-Flight BPEL Process Instances	
Name	Instance	BPEL Process	Last Modified ↑
CreateSDRequest CustomSDR1 TaskActionHandler TaskManager <a href="#">ZZ_CreateSingle_ShipDebitRequest</a>			
Recently Completed BPEL Process Instances ( <a href="#">More...</a> )			
✔ 10002 : Instance #10002 of CustomSDR1		CustomSDR1 (v. 1.0)	9/14/08 5:41:46 AM
⚠ 10001 : Instance #10001 of CustomSDR1		CustomSDR1 (v. 1.0)	9/14/08 5:35:43 AM
✔ 2 : Instance #2 of CreateSDRequest		CreateSDRequest (v. 1.0)	9/11/08 6:49:45 AM
✔ 1 : Instance #1 of CreateSDRequest		CreateSDRequest (v. 1.0)	9/11/08 6:38:37 AM

⊙ Deploy New Process

Oracle BPEL Console v10.1.3.3.0

4. Click the ZZ\_CreateSingle\_ShipDebitRequest link to open the Initiate tab.
5. Enter Payload input field, such as 'SD-Request1' and click **Post XML Message** to initiate the process.

**Note:** The Request Number entered here should be unique each time that you initiate the process because this number will be used as the Supplier Ship and Debit number across users in Oracle Trade Management.



## Entering Payload Information

Testing this BPEL Process

Through SOAP | Through Java Delivery API

**Initiating a test instance**  
To create a new 'test' instance of this BPEL Process, fill this form and click on the 'Post XML Message' button.

Operation: process | HTML Form | XML Source

WS-Security ☐ Include In Header

WS-Addressing ☐ Include In Header

payload

input: SD-Request1 | xsd:string

Note: XML source view contents will not be reflected in the HTML form view

☐ Save Test

☐ Perform stress test

PostXMLMessage

Help: [XML Schema Type Formats](#)

Oracle BPEL Console v10.1.3.3.0

- You can verify SOAP Response in BPEL Console.

## Verifying SOAP Response in BPEL Console

ORACLE Enterprise Manager 10g  
BPEL Control

Manage BPEL L

Dashboard | BPEL Processes | Instances

BPEL Process: ZZ\_CreateSingle\_ShipDebitRequest | Version: 1.0 | Lifecycle: Active

Statistics: [Open Instances: 1](#) | [Closed Instances: 0](#)

Manage | Initiate | Descriptor | WSDL | Sensors | Source | Test Suites | Reports

Test Instance Initiated

Your test request was processed synchronously. It took 12.755seconds to finish and generated the following output:

Value: 

```
<ZZ_CreateSingle_ShipDebitRequestProcessResponsehttp://xmlns.oracle.com/ZZ_CreateSingle_ShipDebitRequest>
  <CX_RETURN_STATUShttp://xmlns.oracle.com/apps/oxf/soapprovider/plsql/zz_sdrequest/zz_create_sdrequest/>S</CX_RETURN_STATUS>
  <CX_MSG_COUNThttp://xmlns.oracle.com/apps/oxf/soapprovider/plsql/zz_sdrequest/zz_create_sdrequest/>23</CX_MSG_COUNT>
  <CX_REQUEST_HEADER_IDhttp://xmlns.oracle.com/apps/oxf/soapprovider/plsql/zz_sdrequest/zz_create_sdrequest/>111</CX_REQUEST_HEADER_ID>
</ZZ_CreateSingle_ShipDebitRequestProcessResponse>
```

For more information:

[Visual Flow](#) | [Audit Instance](#) | [Debug Instance](#)

Click [here](#) to initiate another test instance.

Look for 'S' in CX\_RETURN\_STATUS for success. If 'E' is displayed instead, then it means error has occurred while processing the service. Look for detailed exception message in SOA Monitor.

- Log on to Oracle E-Business Suite as `trademgr/welcome` and then select the

Oracle Trade Management User responsibility. Select the 'Supplier Ship and Debit' link from the navigation menu to open the Ship and Debit Overview window.

8. Verify if the request number 'SD-Request1' that you entered in Step 5 appears in the list.

**Warning**  
Low-level Diagnostic Logging is turned on. This may temporarily reduce performance.

**Ship and Debit Overview**

Views

View: My Ship and Debit Requests Go Personalize Advanced Search

Create Ship and Debit Request Previous 1-10 Next 10

Request Number	Accrual Type	Supplier	Supplier Location	Authorization Number	Status
<a href="#">SD-Request1</a>	Supplier	Abbott Laboratories, Inc.	CORP HQ		Assigned
<a href="#">SDR-CREATE-110</a>	Supplier	Abbott Laboratories, Inc.	CORP HQ		Assigned
<a href="#">SDR-CREATE-M12</a>	Supplier	Abbott Laboratories, Inc.	CORP HQ		Assigned
<a href="#">sd1</a>	Supplier	Abbott Laboratories, Inc.	CORP HQ		Assigned
<a href="#">SDR-CREATE-SDR10</a>	Supplier	Abbott Laboratories, Inc.	CORP HQ		Assigned
<a href="#">SDR-CREATE-M11</a>	Supplier	Abbott Laboratories, Inc.	CORP HQ		Assigned
<a href="#">SDR-CREATE-M41</a>	Supplier	Abbott Laboratories, Inc.	CORP HQ		Assigned
<a href="#">SDR-CREATE-M3</a>	Supplier	Abbott Laboratories, Inc.	CORP HQ		Assigned
<a href="#">SDR-CREATE-M1</a>	Supplier	Abbott Laboratories, Inc.	CORP HQ		Assigned

9. Click the request number 'SD-Request' link to open the Ship and Debit Request Details page. You can verify the request details.

**ORACLE® Trade Management**

Home Trade Management Direct-ADS-Disco Profile Sign

Customer Product Quota Budget Trade Planning Claim Indirect Sales Management Administration Calendar Supplier Ship and Debit

Requests Batch Creation Batch Summary

Quick Find Offer  Go

Logged In As TRADEMGR

**Warning**  
Low-level Diagnostic Logging is turned on. This may temporarily reduce performance.

**Ship and Debit Request Details**

\* Indicates required field

Cancel Save Apply Copy Report

Accrual Type	Supplier	Operating Unit	Vision Operations
Request Type	Bid Request	Authorization Number	
* Request Number	SD-Request1	* Supplier	Abbott Laboratories, Inc.
Supplier Contact			1045 Sansome St, San Francisco, CA 94111
Supplier Contact Email Address	sdr.supplier@testing.com	Supplier Contact Phone Number	2255
* Requestor	Sonneshein, Mr. Evans	Supplier Response By Date	
Assignee	Sonneshein, Mr. Evans	Assignee Response By Date	
* Start Date	18-Aug-2008	Status	Assigned
* End Date	18-Oct-2008	Request Outcome	In Progress
Request Currency	US dollar	Supplier Quote Number	
Sales Order Currency		Internal Order Number	
Description	Create		



---

## Working With Oracle Workflow Business Event System to Invoke Web Services

This chapter covers the following topics:

- Oracle Workflow and Service Invocation Framework Overview
- Web Service Invocation Using Service Invocation Framework
- Calling Back to Oracle E-Business Suite With Web Service Response
- Invoking Web Services
- Managing Errors
- Testing Web Service Invocation
- Troubleshooting Web Service Invocation Failure
- Extending Seeded Java Rule Function
- Other Invocation Usage Considerations

### Oracle Workflow and Service Invocation Framework Overview

Oracle E-Business Suite Integrated SOA Gateway leverages Oracle Workflow Java Business Event System to provide infrastructure for Web Service Invocation natively from Oracle E-Business Suite.

Oracle Workflow is the primary process management solution within Oracle E-Business Suite. It consists of some key components enabling you model and automate business processes and activities in a process diagram based on user-defined business rules, providing routing mechanism to support each decision maker in the process, facilitating subscriptions to significant events or services between systems, and implementing workflow process definitions at run time with monitoring capability of each workflow state as well as handling errors. Since it provides a total solution of managing and streamlining complex business processes and supporting highly-integrated workflow in and out from Oracle E-Business Suite, Oracle E-Business Suite Integrated SOA Gateway relies on Oracle Workflow to enable the service invocation process and provide the

following functionality:

- It relies on Business Event System to create events and event subscriptions as well as to parse a given WSDL representing a Web service to be consumed as subscription parameters.
- It uses the Oracle Workflow seeded Java rule function `oracle.apps.fnd.wf.bes.WebServiceInvokerSubscription` to help invoke Web services.
- It relies on the Oracle Workflow Test Business Event page to test service invocation by raising an invoker event raised from PL/SQL or Java and execute synchronous and asynchronous subscriptions to the event.
- It utilizes the Error processing feature provided in Business Event System to manage errors during subscription execution and sends error notifications to SYSADMIN user with Web service definition, error and event details.
- It utilizes workflow Notification System to send error notifications to and process responses from SYSADMIN.

For detailed information about Oracle Workflow, see *Oracle Workflow User's Guide*, *Oracle Workflow Developer's Guide*, and *Oracle Workflow Administrator's Guide*.

To better understand how service invocation framework is used in facilitating the invocation of Web services, the following topics are discussed in this chapter:

- Web Service Invocation Using Service Invocation Framework, page 11-2
- Calling Back to Oracle E-Business Suite With Web Service Response, page 11-28
- Invoking Web Services, page 11-30
- Managing Errors, page 11-36
- Testing Web Service Invocation, page 11-37
- Troubleshooting Web Service Invocation Failure, page 11-43
- Extending Seeded Java Rule Function, page 11-48
- Other Invocation Usage Considerations, page 11-54

## Web Service Invocation Using Service Invocation Framework

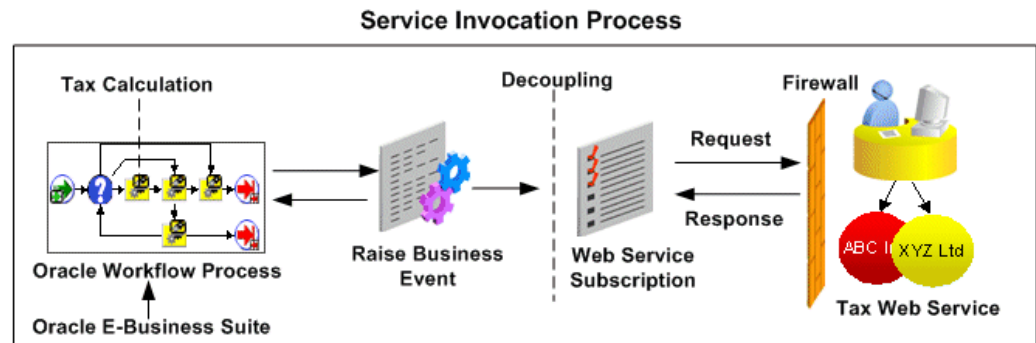
Service invocation framework provides an infrastructure allowing developers to interact with Web services through WSDL descriptions and to invoke Web services from Oracle E-Business Suite.

To achieve this goal, the invocation framework uses a wizard based user interface in Oracle Workflow Business Event System to parse a given Web service WSDL URL during the subscription creation and store identified service information or metadata as subscription parameters that will be used later during service invocation.

Since a WSDL URL is used in representing a Web service, the underlying service can be a simple native Web Service or it can be a BPEL process.

Please note that the service invocation framework discussed here only supports document-based Web service invocation. The invocation framework does not support RPC (remote procedure call) style Web service invocation.

The following diagram illustrates the high level service invocation process flow:



To successfully invoke Web services at run time, Web service invocation metadata must first be in place. In addition to defining the invocation metadata, the concepts of message patterns, Web service input message parts, and Web service security that the service invocation framework supports are also introduced in this section.

The section covers the following topics:

- Understanding Message Patterns, page 11-3
- Defining Web Service Invocation Metadata, page 11-5
  1. Creating a Web Service Invoker Business Event, page 11-6
  2. Creating Local and Error Event Subscriptions to the Invoker Event, page 11-8
  3. Creating a Receive Event and Event Subscription (Optional), page 11-17
- Understanding Web Service Input Message Parts, page 11-21
- Supporting WS-Security, page 11-26

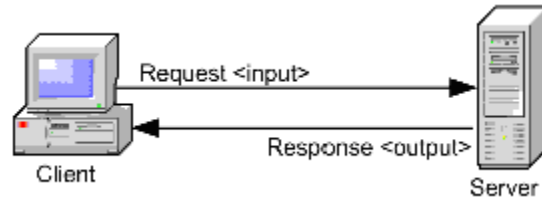
## Understanding Message Patterns in WSDL

There are two major message exchange patterns — a request-response pattern, and a one-way (request - only) pattern.

## Request - Response Message Pattern

The *request - response* message exchange pattern is where a client asks a service provider a question and then receives the answer to the question. The answer may come in the form of a fault or exception. Both the request and the response are independent messages. The request - response pattern is often implemented using synchronous operations for simple operations. For longer running operations, asynchronous (with message correlation) is often chosen.

### Request - Response Message Pattern



- A *synchronous* operation is one that waits for a response before continuing on. This forces operations to occur in a serial order. It is often said that an operation, "blocks" or waits for a response. Many online banking tasks are programmed in request/response mode.

For example, a request for an account balance is executed as follows:

- A customer (the client) sends a request for an account balance to the Account Record Storage System (the server).
- The Account Record Storage System (the server) sends a reply to the customer (the client), specifying the dollar amount in the designated account.
- An *asynchronous* operation is one that does not wait for a response before continuing on. This allows operations to occur in parallel. Thus, the operation does not, "block" or wait for the response. Asynchronous operations let clients continue to perform their work while waiting for responses that may be delayed. This is accomplished by returning an asynchronous handle that runs a thread in the background, allowing the client to continue execution until the response is ready.

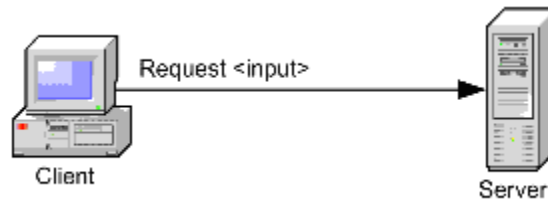
**Important:** In this release, the Web service invocation framework only supports Synchronous Request - Response message pattern and One - Way (Request Only) message pattern.

## Request Only Message Pattern

The *request only* operation model includes one **input** element, which is the client's request to the server. No response is expected.



### ***Request Only Message Pattern***



For example, client zip code locations send updated weather data to the service when local conditions change using the request only operation. The server updates the data but no response is sent back.

## **Defining Web Service Invocation Metadata**

Because the service invocation is taken place in the Oracle Workflow Business Event System, before invoking a Web service, the Web service invocation metadata including events and event subscriptions must be defined first through the Business Event System.

This section discusses the following topics:

### **1. Creating a Web Service Invoker Business Event, page 11-6**

A Web service Invoker business event that serves as a request message (or Web service input message) for a service needs to be created first.

### **2. Creating Local and Error Event Subscriptions to the Invoker Event, page 11-8**

After defining the Invoker event, you need to create the following two subscriptions:

- Create a Local subscription with 'Invoke Web Service' Action Type, page 11-8  
This event subscription indicates that when a triggering event occurs, the action item of this subscription is to invoke a Web service defined as part of this subscription.
- Create an Error subscription with 'Launch Workflow' Action Type, page 11-15  
This error subscription enables error processing in the Business Event System that is used to communicate with SYSADMIN user of an error condition in subscription execution.

### **3. Creating a Receive Event and Event Subscription (Optional), page 11-17**

This step is required only if a Web service has an output or a response message to communicate or callback to Oracle E-Business Suite. Once a receive event is in place, you must create an External subscription to the receive event to pass the Web

service response message.

If a Web service does not require a response, then you do not need to create a receive event, nor the event subscription.

## Step 1: Creating a Web Service Invoker Business 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.

Use the Oracle Workflow Business Event System to define a Web service invoker business event.

The invoker event can be served as a request message (or Web service input message) in a message pattern to send inquiries to a service.

To invoke a Web service through the Business Event System, we will first create an invoker business event, and then subscribe to the invoker event later with an appropriate action type.

**Note:** In this release, the Web service invocation framework supports the following types of service invocation:

- **One-way (request only)** service that a consumer or client sends a message to a service, and the service does not need to reply.
- **Synchronous request-response** service type that requires a response before an operation continues.

If an invoker event requires a response, then you must define a receive business event to communicate or callback into Oracle E-Business Suite after the Web service is successfully invoked. See *Creating a Receive Event and Event Subscription (Optional)*, page 11-17.

For more information about business events, see *Events, Oracle Workflow Developer's Guide*.

### To create an invoker event:

1. Log on to Oracle E-Business Suite with the Workflow Administrator Web responsibility. Select the Business Events link, and choose Events in the horizontal navigation if the Events page is not already displayed.
2. In the Events page, click **Create Event** to open the Create Event page.
3. Enter the following information in the Create Event page:

- Name: Enter an event name, such as `oracle.apps.wf.xmlgateway.invoke`
- Display Name: Enter an event display name, such as `oracle.apps.wf.xmlgateway.invoke`
- Description: Enter a description for the event
- Status: Enabled
- Generate Function: Specify a generate function for the PL/SQL based event if the application where the event occurs will not provide the event data
- Java Generate Function: Specify a generate function for the Java based event if the application where the event occurs will not provide the event data
- Owner Name: Specify the program or application name that owns the event (such as Oracle Workflow)
- Owner Tag: Specify the program or application ID that owns the event (such as 'FND')

#### Create Invoker Event

**ORACLE® Administrator Workflow**

Diagnostics Home Logout Preferences Help

Home Developer Studio Business Events Status Monitor Notifications Administration

Events Subscriptions Agents Systems

Business Events: Events >

**Create 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.

\* Indicates required field

\* Name

\* Display Name

Description

\* Status

Generate Function

Java Generate Function

\* Owner Name

\* Owner Tag

Customization Level

Cancel Apply

Home Developer Studio Business Events Status Monitor Notifications Administration Diagnostics Home Logout Preferences Help

About this Page Privacy Statement Copyright (c) 2006, Oracle. All rights reserved.

#### 4. Click **Apply** to save your work.

Leave this page open to create a receive event.

For more information on how to create a business event, see *Oracle Workflow Developer's Guide* for details.

## Step 2: Creating Local and Error Event Subscriptions to the Invoker Event

- **Create a Local subscription with 'Invoke Web Service' Action Type**, page 11-8

This event subscription indicates that when a triggering event occurs, the action item of this subscription is to invoke a Web service that you have created in the invoke event.

- **Create an Error subscription with 'Launch Workflow' Action Type**, page 11-15

This error subscription enables error processing in the Business Event System that is used to communicate with SYSADMIN user of an error condition in subscription execution.

It sends a workflow notification to SYSADMIN with Web service definition, error details, and event details allowing the SYSADMIN to process the errors if needed.

### Create a Local Subscription With 'Invoke Web Service' Action Type

To subscribe to an invoker event, you must create a subscription with 'Invoke Web Service' Action Type which indicates that when a triggering event occurs, the action item of this subscription is to invoke a Web service. This requires you enter a WSDL URL representing a Web service of any type (such as a native Web service or BPEL process) in the Create Event Subscription - Invoke Web Service wizard. That WSDL information entered in the wizard will then be parsed into service metadata for further selections.

**Note:** A BPEL process itself is a Web service, defining and supporting a client interface through WSDL and SOAP. The BPEL process WSDL URL can be created through a partner link which allows the request to be published to the Oracle BPEL Process Manager to connect to Web services.

When a triggering event occurs, the Business Event System executes the subscription through the seeded Java function and invokes the BPEL process.

After you select appropriate service metadata, this selected data will be stored as subscription parameters as follows:

- SERVICE\_WSDL\_URL
- SERVICE\_NAME
- SERVICE\_PORT
- SERVICE\_PORTTYPE
- SERVICE\_OPERATION

## The seeded Java Rule Function

`oracle.apps.fnd.wf.bes.WebServiceInvokerSubscription` uses these subscription parameters during the service invocation.

**Note:** Oracle E-Business Suite Integrated SOA Gateway allows developers to extend the invoker subscription seeded rule function using Java coding standards for more specialized service invocation processing. For more information on customizing seeded Java rule function, see *Extending Seeded Rule Function*, page 11-48.

Apart from the subscription parameters that have been parsed and stored through the Invoke Web Service Subscription page, the following information could also be captured if it is specified as additional subscription parameters that will then be used by the seeded Java rule function to enable message processing for Web service invocation:

- Message transformation

If the invoker event's XML payload (to be used as Web service input message) requires to be transformed into a form that complies with the input message schema, the seeded Java rule function could perform XSL transformation on the payload before invoking the Web service. Similarly, if the Web service output message requires to be transformed into a form that is required for processing by Oracle E-Business Suite, the seeded Java rule function could perform XSL transformation on the response before calling back to Oracle E-Business Suite.

- WFBES\_OUT\_XSL\_FILENAME
- WFBES\_IN\_XSL\_FILENAME

After event payload is either passed during the event raise or generated by generate function after the event raise, the seeded Java rule function uses these subscription parameters to obtain the XSL file names if XSL transformations are required on the Web service input and output messages. At run time, if event parameters are passed with the same names, then the event parameters override the subscription parameters.

For more information on these transformation parameters, see *Understanding Web Service Input Message Parts*, page 11-21.

- WS-Security: Information required to add UsernameToken header to a SOAP request.

If the Web service being invoked enforces Username/Password based authentication, then the service invocation framework also supports the UsernameToken based WS-Security header during Web service invocation. The SOAP username and optional password locator information will be passed to the seeded Java rule function as the following subscription parameters when the Java rule function is defined through the Invoke Web Service wizard:

- WFBES\_SOAP\_USERNAME
- WFBES\_SOAP\_PASSWORD\_MOD
- WFBES\_SOAP\_PASSWORD\_KEY

For more information on these WS-Security parameters, see Supporting WS-Security, page 11-26.

- Callback: Callback to Oracle E-Business Suite with Web service response
  - WFBES\_CALLBACK\_EVENT
  - WFBES\_CALLBACK\_AGENT

To process a Web service output or response (synchronous request - response) message, the callback mechanism is used to communicate the response using a business event back to Oracle E-Business Suite by enqueueing the event to an Inbound Workflow Agent. A new or waiting workflow process can be started or executed.

For more information on these callback parameters, see Calling Back to Oracle E-Business Suite With Web Service Response, page 11-28.

### **Creating a Local Event Subscription with 'Invoke Web Service' Action Type**

## Create an Event Subscription

**ORACLE** Administrator Workflow

Diagnostics Home Logout Preferences Help

Home Developer Studio Business Events Status Monitor Notifications Administration

Events Subscriptions Agents Systems

Business Events: Subscriptions >

Cancel Next

### Create Event Subscription

An event subscription is a registration indicating that a particular event is significant to a particular system. An event subscription specifies the processing to perform when the triggering event occurs.

\* Indicates required field

**Subscriber**

\* System ATG121U4.US.ORACLE.COM

**Triggering Event**

\* Source Type Local

\* Event Filter oracle.apps.wf.xmlgateway.invoke

Source Agent

**Execution Condition**

\* Phase 50  
Subscription with a phase 1-99 are run synchronously, 100 and above are deferred.

\* Status Enabled

\* Rule Data Message

**Action Type**

\* Action Type Invoke Web Service  
The Action Type controls the behaviour of the subscription

On Error Stop and Rollback

Cancel Next

Home Developer Studio Business Events Status Monitor Notifications Administration Diagnostics Home Logout Preferences Help

### To create a local event subscription with 'Invoke Web Service' action type:

1. Log on to Oracle E-Business Suite with the Workflow Administrator Web responsibility. Select the Business Events link, and choose Subscriptions in the horizontal navigation.
2. In the Event Subscriptions page, click **Create Subscription** to open the Create Event Subscription page.
3. Enter the following information in the Create Event Subscription page:
  - Subscriber: Select the local system
  - Source Type: Local
  - Event Filter: Select the event name that you just created, such as `oracle.apps.wf.xmlgateway.invoke`
  - Phase: 50

If the event is raised from Java, the phase number determines whether an event will be invoked right away or enqueued to WF\_JAVA\_DEFERRED queue.

**Note:** If the invoker event is raised from PL/SQL, it is always deferred to WF\_JAVA\_DEFERRED queue regardless of the

phase because the subscription has a Java rule function that cannot be executed in the database.

- If the phase is  $\geq 100$ , then the event is enqueued to WF\_JAVA\_DEFERRED queue and will be dispatched later.
  - If the phase is  $< 100$ , then the event is dispatched immediately to the Java Business Event System soon after an triggering event occurs.
- Status: Enabled
  - Rule Data: Message
  - Action Type: Invoke Web Service
  - On Error: Stop and Rollback
4. Click **Next**. This opens a Create Event Subscription - Invoke Web Service wizard allowing you to enter a WSDL URL that will be parsed into service metadata for further selection.

#### Create Event Subscription - Invoke Web Service Wizard

The screenshot shows the Oracle Administrator Workflow interface for the 'Create Event Subscription - Invoke Web Service Wizard'. The top navigation bar includes 'Home', 'Developer Studio', 'Business Events', 'Status Monitor', 'Notifications', and 'Administration'. The 'Business Events' section is expanded, showing 'Events', 'Subscriptions', 'Agents', and 'Systems'. The wizard progress bar indicates five steps: 'Load WSDL', 'Select Service', 'Select Service Port', 'Select Operation', and 'Subscription Documentation'. The current step is 'Load WSDL', titled 'Select a WSDL Source'. The instruction reads: 'Enter the URL of the WSDL to consume in the Business Event Subscription'. A note states '\* Indicates required field'. The 'WSDL URL' field contains the text 'http://fws60062rems.us.oracle.com:8077/OA\_HTML/XMLGatewayWSDL'. Below the field, an example URL is provided: 'Example: http://supplier.company.com:8888/webservices/supplier\_service.wsdl'. 'Cancel' and 'Next' buttons are visible on the right side of the form. The bottom of the page contains a footer with 'About this Page', 'Privacy Statement', and 'Copyright (c) 2006, Oracle. All rights reserved.'

1. Enter WSDL URL information for the Web service to be invoked. Click **Next** to parse the WSDL and display all services.



## Create Event Subscription - Invoke Web Service: Select Service

The screenshot shows the Oracle Administrator Workflow interface for the 'Select Service' step. The breadcrumb trail is: Home > Developer Studio > Business Events > Status Monitor > Notifications > Administration. The 'Subscriptions' tab is active. A progress bar at the top shows five steps: Load WSDL, Select Service (current), Select Service Port, Select Operation, and Subscription Documentation. The main content area has the title 'Select Service' and a sub-header 'Select a service to consume in Business Event Subscription'. It includes a legend: '\* Indicates required field'. The form fields are: 'WSDL Source URL' with the value 'http://rws60062rems.us.oracle.com:8077/OA\_HTML/XMLGatewayWSDL', 'WSDL Description' with the value 'XMLGateway', and 'Service Name' with a dropdown menu showing 'XMLGateway'. Navigation buttons 'Cancel', 'Back', 'Step 2 of 5', and 'Next' are on the right. The footer contains 'About this Page', 'Privacy Statement', and 'Copyright (c) 2009, Oracle. All rights reserved.'

2. Select an appropriate service name from the drop-down list. Click **Next** to display all ports for a selected service

## Create Event Subscription - Invoke Web Service: Select Service Port

The screenshot shows the Oracle Administrator Workflow interface for the 'Select Service Port' step. The breadcrumb trail is: Home > Developer Studio > Business Events > Status Monitor > Notifications > Administration. The 'Subscriptions' tab is active. A progress bar at the top shows five steps: Load WSDL, Select Service, Select Service Port (current), Select Operation, and Subscription Documentation. The main content area has the title 'Select Service Port' and a sub-header 'Select a service port to consume in Business Event Subscription'. It includes a legend: '\* Indicates required field'. The form fields are: 'WSDL Source URL' with the value 'http://rws60062rems.us.oracle.com:8077/OA\_HTML/XMLGatewayWSDL', 'WSDL Description' with the value 'XMLGateway', and 'Selected Service' with the value 'XMLGateway'. Below these is a table with two columns: 'Select Service Port' and 'Port End Point'. The table has one row with the values 'XMLGatewayPort' and 'http://rws60062rems.us.oracle.com:8077/webservices/AppsWSProvider/oracle/apps/fnd/XMLGateway'. Navigation buttons 'Cancel', 'Back', 'Step 3 of 5', and 'Next' are on the right. The footer contains 'About this Page', 'Privacy Statement', and 'Copyright (c) 2009, Oracle. All rights reserved.'

3. Select an appropriate service port and click **Next** to display all operations for a selected port.

### Create Event Subscription - Invoke Web Service: Select Operation

4. Select an appropriate service operation and click **Next**. This opens the last page of the Create Event Subscription - Invoke Web Service wizard.

### Create Event Subscription - Invoke Web Service: Subscription Documentation

5. In the Subscription Documentation of the Create Event Subscription - Invoke Web Service page, the default Java Rule Function name `oracle.apps.fnd.wf.bes.WebServiceInvokerSubscription` is automatically populated.

**Important:** If you have extended the functionality of the seeded rule function, manually enter your custom function name here.

6. In the Documentation region, enter an application name or program name that owns the subscription (such as 'Oracle Workflow') in the Owner Name field and the program ID (such as 'FND') in the Owner Tag field. Click **Apply**.

For more information, see Defining Event Subscriptions, *Oracle Workflow Developer's Guide*.

#### **Create an Error subscription with 'Launch Workflow' Action Type**

To enable the error processing feature during the service invocation, you must create an Error subscription to the invoker business event.

Once subscribing to this error processing, if any error occurs during the invocation, the error process sends a workflow notification to SYSADMIN. This information includes Web service definition, event details, and error details allowing SYSADMIN to easily identify the error. The notification also provides an option for SYSADMIN to respond to the error. The SYSADMIN can invoke the Web service again after the underlying issue that caused the error is resolved, abort the errored event if needed, or reassign an errored notification to another user if appropriate.

For detailed information on managing errors during Web service invocation, see Managing Errors, page 11-36.

#### **To create an error subscription with 'Launch Workflow' action type:**

1. Log on to Oracle E-Business Suite with the Workflow Administrator Web responsibility. Select the Business Events link, and choose Subscriptions in the horizontal navigation.
2. In the Event Subscriptions page, click **Create Subscription** to open the Create Event Subscription page.
3. Enter the following information in the Create Event Subscription page:
  - Subscriber: Select the local system
  - Source Type: Error
  - Event Filter: Select the event name that you just created, such as `oracle.apps.wf.xmlgateway.invoke`
  - Phase: this can be any phase number
  - Status: Enabled
  - Rule Data: Key
  - Action Type: Launch Workflow
  - On Error: Stop and Rollback

**ORACLE® Administrator Workflow** Diagnostics Home Logout Preferences Help

Home **Developer Studio** Business Events Status Monitor Notifications Administration

Events Subscriptions Agents Systems

Business Events: Subscriptions >

Cancel Next

---

**Create Event Subscription**

An event subscription is a registration indicating that a particular event is significant to a particular system. An event subscription specifies the processing to perform when the triggering event occurs.

\* Indicates required field

**Subscriber**

\* System

**Triggering Event**

\* Source Type

\* Event Filter

Source Agent

**Execution Condition**

\* Phase   
Subscription with a phase 1- 99 are run synchronously , 100 and above are deferred.

\* Status

\* Rule Data

**Action Type**

\* Action Type    
The Action Type controls the behaviour of the subscription

On Error

Cancel Next

---

Home Developer Studio Business Events Status Monitor Notifications Administration Diagnostics Home Logout Preferences Help

About this Page Privacy Statement Copyright (c) 2006, Oracle. All rights reserved

4. Click **Next** to open the Create Event Subscription - Launch Workflow page.
5. Enter the following information in the Action region:
  - Workflow Type: WFERROR
  - Workflow Process: DEFAULT\_EVENT\_ERROR2
  - Priority: Normal
6. In the Documentation region, enter an application or program name that owns the event subscription (such as Oracle Workflow) in the Owner Name field and application or program ID (such as 'FND') in the Owner Tag field.

**ORACLE Administrator Workflow** Diagnostics Home Logout Preferences Help

Business Events: Subscriptions > Create Event Subscription >  
**Create Event Subscription - Launch Workflow**

Cancel Back Apply

An event subscription can be routed to a Workflow process. Please specify the Workflow Type and Workflow Process to be launched.  
 \* Indicates required field

**Action**

\* Workflow Type:

\* Workflow Process:    
Choose a Workflow Type, before choosing the Workflow Process for that Type

\* Priority:

Additional Options:

**Subscription Parameters**

Select Object:

[Select All](#) | [Select None](#)

Select Name	Value
<input type="checkbox"/>	

Enter parameters and their values with no spaces

**Documentation**

\* Owner Name:

\* Owner Tag:

Customization Level:

Description:

Cancel Back Apply

7. Click **Apply**.

### Step 3: Creating a Receive Event and Subscription (Optional)

A receive event can serve as a communication vehicle to communicate or callback to Oracle E-Business Suite if a Web service has an output or response message required to be communicated back after the Web service is successfully invoked. However, whether you need to create a receive event and external subscription to the receive event depends on the following criteria:

- Your message pattern
- Where your event is raised from (Java or PL/SQL layer)
- Event subscription phase number

#### For Synchronous Request-Response Web Service Invocation

- If the Web service invoker event is raised from Java code in the middle tier, and the invoker subscription is synchronous with subscription phase < 100, then the Web service is invoked as soon as the event is raised, and if successful the response can be read by the calling application and is available immediately by using `BusinessEvent.getResponseData()` method after calling `BusinessEvent.raise()`.

In this case, the response may not have to be communicated back to Oracle E-Business Suite using callback event. Hence, you may not need to create a receive

event and the subscription to the event.

- If the Web service invoker event is raised from Java code with the subscription phase is  $\geq 100$ , or if the event is raised from PL/SQL, the event message will be enqueued to WF\_JAVA\_DEFERRED queue. In this situation, you will need to create a receive event and external subscription to the event if the Web service has an output or a response message. Callback event with callback agent is required to receive the output message into Oracle E-Business Suite.

This receive event can also be used as a callback into Oracle E-Business Suite to let the interested parties know through raising this event that the Web service response is available.

See: Calling Back to Oracle E-Business Suite With Web Service Response, page 11-28.

If a receive event is required, after creating the receive event, you must create an external event subscription to the receive event. The Web service response message communicated through the receive event is always enqueued to an inbound workflow agent. In order to process an event from the inbound workflow agent, an external subscription is required.

### For Request-only Web Service

If it is a request-only Web service which does not require a response, you do not need to create a receive event.

#### To create a receive event:

1. In the Events page, click **Create Event** to open another Create Event page.
2. Enter the following information in the Create Event page:
  - Name: Enter an event name, such as `oracle.apps.wf.xmlgateway.receive`
  - Display Name: Enter an event display name, such as `oracle.apps.wf.xmlgateway.receive`
  - Description: Enter a description for the event
  - Status: Enabled
  - Owner Name: Enter an application or program name that owns the event (such as 'Oracle Workflow')
  - Owner Tag: Enter the application or program ID that owns the event (such as 'FND')

### Create Receive Event

**ORACLE** Administrator Workflow

Diagnostics Home Logout Preferences Help

Home Developer Studio **Business Events** Status Monitor Notifications Administration

Events Subscriptions Agents Systems

Business Events: Events >

**Create 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.

\* Indicates required field

\* Name

\* Display Name

Description

\* Status

Generate Function

Java Generate Function

\* Owner Name

\* Owner Tag

Customization Level

Home Developer Studio Business Events Status Monitor Notifications Administration Diagnostics Home Logout Preferences Help

About this Page Privacy Statement Copyright (c) 2006, Oracle. All rights reserved.

3. Click **Apply** to create a receive event.

#### To create a receive event subscription:

1. Log on to Oracle E-Business Suite with the Workflow Administrator Web Applications responsibility. Select the Business Events link, and choose Subscriptions in the horizontal navigation.
2. In the Event Subscriptions page, click **Create Subscription** to open the Create Event Subscription page.
3. Enter the following information in the Create Event Subscription page:
  - Subscriber: Select the local system
  - Source Type: External
  - Event Filter: Select the receive event name that you just created, such as `oracle.apps.wf.xmlgateway.receive`
  - Phase: any phase number
  - Status: Enabled
  - Rule Data: Key
  - Action Type: any action type
  - On Error: Stop and Rollback

**ORACLE Administrator Workflow**

Home | Developer Studio | **Business Events** | Status Monitor | Notifications | Administration

Events | Subscriptions | Agents | Systems

Business Events: Subscriptions >

**Create Event Subscription**

An event subscription is a registration indicating that a particular event is significant to a particular system. An event subscription specifies the processing to perform when the triggering event occurs.

\* Indicates required field

**Subscriber**

\* System: ATG121U4.US.ORACLE.COM

**Triggering Event**

\* Source Type: External

\* Event Filter: oracle.apps.wf.xmlgateway.receive

Source Agent:

**Execution Condition**

\* Phase: 50  
Subscription with a phase 1- 99 are run synchronously , 100 and above are deferred.

\* Status: Enabled

\* Rule Data: Message

**Action Type**

\* Action Type: Launch Workflow  
The Action Type controls the behaviour of the subscription

On Error: Stop and Rollback

Cancel Next

Home | Developer Studio | Business Events | Status Monitor | Notifications | Administration | Diagnostics | Home | Logout | Preferences | Help

About this Page | Privacy Statement

Copyright (c) 2006, Oracle. All rights reserved.

4. Click **Next** to open the Create Event Subscription - Launch Workflow page.

Please note that the type of the Create Event Subscription page to be shown depends on the value selected in the Action Type field. If "Launch Workflow" is selected, you will see the Create Event Subscription - Launch Workflow page. If other action types are selected, different types of the create event subscription pages are displayed. By entering an appropriate action type through the subscription page, you can launch a workflow process or just execute a custom rule function for the event defined as part of this subscription.

5. Enter the following information in the Action region:
  - Workflow Type: Enter any workflow type that is waiting for the response
  - Workflow Process: Enter any workflow process that is waiting for the response
  - Priority: Normal
6. In the Documentation region, enter an application or program name in the Owner Name field (such as 'Oracle Workflow') and application or program ID in the Owner Tag field (such as 'FND').



**ORACLE<sup>®</sup> Administrator Workflow** Diagnostics Home Log

Business Events: Subscriptions > Business Events: Event Subscriptions > Update Event Subscriptions >  
**Update Event Subscription: Launch Workflow** Cancel

An event subscription can be routed to a Workflow process. Please specify the Workflow Type and Workflow Process to be launched.  
 \* Indicates required field

**Action**

\* Workflow Type

\* Workflow Process  the Workflow Process for that Type

\* Priority

Additional Options

**Subscription Parameters**

Select Name	Value
No results found.	
<input type="button" value="Add Another Row"/>	
<small>Enter parameters and their values with no spaces</small>	

**Documentation**

\* Owner Name

\* Owner Tag

Customization Level

Description

7. Click **Apply**.

## Understanding Web Service Input Message Parts

A message consists of one or more logical parts. Each part describes the logical abstract content of a message. For example, a typical document-style Web service could have a header and body part in the input message.

For example, consider the operation `PROCESSPO` in Oracle E-Business Suite XML

Gateway service (

`http://<hostname>:<port>/webservices/SOAPProvider/xmlgateway/ont_  
_poi/?wsdl`) as described below.

```

<definitions targetNamespace="ONT__POI"
targetNamespace="http://xmlns.oracle.com/apps/ont/soapprovider/xmlgateway
/ont__poi/">
<type>
  <schema elementFormDefault="qualified"
targetNamespace="http://xmlns.oracle.com/apps/ont/soapprovider/xmlgateway
/ont__poi/">
    <include
schemaLocation="http://rws60066rems.us.oracle.com:8078/webservices/SOAPr
ovider/xmlgateway/ont__poi/PROCESS_PO_007.xsd"/>
    </schema>
  ...
<message name="PROCESSPO_Input_Msg">
  <part name="header" element="tns:SOAHeader"/>
  <part name="body" element="tns1:PROCESS_PO_007"/>
</message>
...
<binding name="ONT__POI_Binding" type="tns:ONT__POI_PortType">
<soap: binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="PROCESSPO">
    <soap:operation
soapAction="http://host:port/webservices/SOAPProvider/xmlgateway/ont__poi
"/>
    <input>
      <soap:header message="tns:PROCESSPO_Input_Msg" part="header"
use="literal"/>
      <soap:body parts="body" use="literal"/>
    </input>
  </operation>
</binding>
...
</definitions>

```

The operation PROCESSPO requires input message PROCESSPO\_Input\_Msg, which has two parts:

- Body: The value of PROCESS\_PO\_007 type to be set as SOAP body is sent as business event payload.
- Header: The value of SOAHeader type to be sent in the SOAP header which is required for Web Service authorization.

To better understand the Web service operation's input message, the section includes the following topics:

- Event Payload as SOAP Body, page 11-22
- Other Web Service Input Message Parts, page 11-25

### Event Payload as SOAP Body

Any detail information needed to describe what occurred in an event, in addition to the event name and event key, is called the event data. For example, the event data for a purchase order event includes the item numbers, descriptions, and cost.

During the event creation, you can have the event data specified either with or without

using the Generate Function for an event from both PL/SQL and Java. If the application where the event occurs does not provide event data, then you can use the Generate Function while creating the event. The Generate Function will produce the complete event data from the event name, event key, and an optional parameter list at the event raise. Otherwise, you do not need to specify the Generate Function field if the application where the event occurs does provide event data. In other words, the event payload can be passed in either one of the following ways:

- Event data or payload is passed through the Generate Function during the event raise.
- Event data or payload is passed along with the event itself without using the Generate function.

**Note:** The generate function must follow a standard PL/SQL or Java API. See *Oracle Workflow Developer's Guide* and *Oracle Workflow API Reference*.

The event data can be structured as an XML document and passed as SOAP body during the event raise. The seeded Java rule function accepts this SOAP body through business event payload. The SOAP body is described in a well-formed XML element that would be embedded into SOAP envelope.

- `BusinessEvent.setData(String)`
- `WF_EVENT.Raise(... p_event_data => ...);`

#### ***Message Transformation Parameters to Support XSL Transformation***

If the invoker event's XML payload (to be used as Web service input message) requires to be transformed into a form that complies with the input message schema, the seeded Java rule function could perform XSL transformation on the payload before invoking the Web service. Similarly, if the Web service output message requires to be transformed into a form that is required for processing by Oracle E-Business Suite, the seeded Java rule function could perform XSL transformation on the response before calling back to Oracle E-Business Suite.

**Note:** An input message is the XML payload that is passed to the Web service in the SOAP request. An output message is the XML document received as a response from the Web service after a successful invocation.

For the synchronous request - response operation, when the output (response) message, an XML document, is available, if this XML document requires to be transformed to a form that is easier for Oracle E-Business Suite to understand, then XSL transformation on the output message will be performed.

**Note:** The XSL filename is given based on the format of <File Name>:<Application Short Name>:<Version>.

For example, "PO\_XSL\_1\_1\_2.xsl:FND:1.1".

The XSL file names are passed to the seeded Java rule function as the following subscription parameters while creating the subscription to the Web service invoker event through the Create Event Subscription - Invoke Web Service wizard:

- WFBES\_OUT\_XSL\_FILENAME: XSL file to perform transformation on the output (response) message

For example, WFBES\_OUT\_XSL\_FILENAME=PO\_XSL\_OUT\_2.xsl:FND:1.1

- WFBES\_IN\_XSL\_FILENAME: XSL file to perform transformation on the input message

For example, WFBES\_IN\_XSL\_FILENAME=PO\_XSL\_IN\_2.xsl:FND:1.1

At run time, the XSL filenames are passed through the same parameters as event parameters. If event parameters are passed with the same names as the subscription parameters that have been parsed and stored, the event parameter values override the subscription parameter values. For example, the event parameters are passed as follows:

- `BusinessEvent.setStringProperty("WFBES_OUT_XSL_FILENAME", "PO_XSL_OUT_2.xsl:FND:1.1");`
- `BusinessEvent.setStringProperty("WFBES_IN_XSL_FILENAME", "PO_XSL_IN_2.xsl:FND:1.1");`

If WFBES\_OUT\_XSL\_FILENAME is null, no outbound transformation will be performed.

If WFBES\_IN\_XSL\_FILENAME is null, no inbound transformation will be performed.

#### ***Loading XSL files to Oracle E-Business Suite***

The seeded Java rule function performs the XSL transformation on the input and output messages by using the XML Gateway API,

`ECX_STANDARD.perform_xslt_transformation`; therefore, the XSL files for the XSL transformation on input and output messages are loaded to Oracle XML Gateway using the `oracle.apps.ecx.loader.LoadXSLTToClob` loader.

**Note:** For information on the XSL transformation PL/SQL API, see Execution Engine APIs, *Oracle XML Gateway User's Guide*.

As a result, use the following steps to perform XSL transformation during service invocation:

1. Upload the XSL files to Oracle E-Business Suite using the `oracle.apps.ecx.loader.LoadXSLTToClob` loader in Oracle XML Gateway.

2. Specify the XSL file names (such as `PO_XSL_IN_2.xsl:FND:1.1`) in the event or subscription parameters (`WFBES_IN_XSL_FILENAME` and `WFBES_OUT_XSL_FILENAME`) if applicable for XSL transformation on input and output messages.

For example, upload the XSL files to Oracle E-Business Suite as follows:

```
java oracle.apps.ecx.loader.LoadXSLTToClob apps apps
ap601sdb:4115:owf12dev PO_XSL_IN_2.xsl FND 1.1
```

For more information, see Loading and Deleting an XSLT Style Sheet, *Oracle XML Gateway User's Guide*.

### Other Web Service Input Message Parts

Apart from passing the SOAP body part as event payload, service invocation framework also supports passing values for other parts that are defined for the Web service operation's input message using the business event parameter with the following format:

`WFBES_INPUT_<partname>`

`<partname>` is same as the part name in the input message definition in WSDL.

For example, the header part for above example is passed to business event as parameter **WFBES\_INPUT\_header** during the invoker event raise. The following code snippet shows the header part that is used to pass username, responsibility, responsibility application, and NLS language elements for Web service authorization:

```
String headerPartMsg = "<ns1:SOAHeader
xmlns:ns1=\"http://xmlns.oracle.com/xdb/SYSTEM\" \" \" +
    \"env:mustUnderstand=\"0\" \"
xmlns:env=\"http://schemas.xmlsoap.org/soap/envelope/\"> \n\" +
    \" <ns1:MESSAGE_TYPE>XML</ns1:MESSAGE_TYPE>\n\" +
    \" <ns1:MESSAGE_STANDARD>OAG</ns1:MESSAGE_STANDARD>\n\" +
    \" <ns1:TRANSACTION_TYPE>PO</ns1:TRANSACTION_TYPE>\n\" +
    \" <ns1:TRANSACTION_SUBTYPE>PROCESS</ns1:TRANSACTION_SUBTYPE>\n\"
+
    \" <ns1:DOCUMENT_NUMBER>123</ns1:DOCUMENT_NUMBER>\n\" +
    \" <ns1:PARTY_SITE_ID>4444</ns1:PARTY_SITE_ID>\n\" +
    \"</ns1:SOAHeader>\n\";
businessEvent.setStringProperty(\"WFBES_INPUT_header\", headerPartMsg);
```

**Note:** Please note that this `WFBES_INPUT_<partname>` parameter can only be passed at run time during the event raise, not through the event subscription. Several constants are defined in interface `oracle.apps.fnd.wf.bes.InvokerConstants` for use in Java code.

If the Web service input message definition has several parts, value for the part that is sent as SOAP body is passed as event payload. Values for all other parts are passed as event parameters with parameter name format `WFBES_INPUT_<partname>`. If the value for a specific input message part is optional to invoke the Web service, you still have to pass the parameter with null value so that invoker subscription knows to which

part the event payload should be set as SOAP body.

For example, if input message part `myheader` for a Web service is optional and does not require a valid value for the invocation to succeed, the event parameter for the input should still be set with null value as follows.

```
businessEvent.setStringProperty("WFBES_INPUT_myheader", null);
```

## Supporting WS-Security

Web service security (WS-Security) is a communication protocol providing a means for applying security to Web services. It describes enhancements to SOAP messaging to provide quality of protection through message integrity and single message authentication. It also describes how to attach security tokens to SOAP messages to enhance security features.

Service invocation framework supports WS-Security in a general-purpose mechanism for associating security tokens with messages to authenticate Web service requests and service invocation from Oracle E-Business Suite.

To accomplish this goal, service invocation framework supports WS-Security through UsernameToken based security.

### UsernameToken Based Security

This security mechanism provides a basic authentication for Web service invocation by passing a *username* and an optional *password* in the SOAP Header of a SOAP request sent to the Web service provider.

Please note that the username/password information discussed in this UsernameToken based security model is the concept of Oracle E-Business Suite username/password.

If the Web service being invoked enforces Username/Password based authentication, then the service invocation framework also supports the UsernameToken based WS-Security header during Web service invocation.

**Note:** SOAP requests invoking the Web services should include security header consisting of Username and Plain text password. Encryption is not supported in this release.

#### *Username*

The username is a clear text, and its value for the operation is stored in the subscription parameter:

```
WFBES_SOAP_USERNAME
```

For example, `WFBES_SOAP_USERNAME = SYSADMIN`

Following sample code describes how the username is stored with the subscription as Web service metadata:

```
SERVICE_WSDL_URL=http://myhost.us.oracle.com:8040/OA_HTML/XMLGatewayWSDL
SERVICE_NAME=XMLGateway SERVICE_PORT=XMLGatewayPort
SERVICE_PORTTYPE=XMLGatewayPortType SERVICE_OPERATION=ReceiveDocument
WFBES_SOAP_USERNAME=kwalker
```

### ***Password***

Password is the most sensitive part of the UsernameToken profile. Service invocation framework supports the UsernameToken based WS-Security during service invocation with username and an optional password with Type PasswordText.

**Note:** The PasswordText password type is the password written in clear text. There is another password type called 'PasswordDigest' which is a base64-encoded SHA1 hash value of the UTF8-encoded password and this type of password is not supported in this release.

The password corresponding to the SOAP username is stored in FND vault using a PL/SQL script `$FND_TOP/sql/afvltput.sql`. For example,

```
sqlplus apps/apps@db @$FND_TOP/sql/afvltput.sql <Module> <Key>
<Value>
```

- `<Module>` parameter represents the application code such as FND, PO, AR, AP and so on for which the service invocation is being implemented.
- `<Key>` parameter should be uniquely named within the module in consideration for namespace issues. It is the developer's discretion to name the key.
- `<Value>` parameter represents the password value.

The module and key values to retrieve the password corresponding to the SOAP username are passed to the following invoker subscription parameters:

- WFBES\_SOAP\_PASSWORD\_MOD

For example, WFBES\_SOAP\_PASSWORD\_MOD=FND

- WFBES\_SOAP\_PASSWORD\_KEY

For example, WFBES\_SOAP\_PASSWORD\_KEY=sysadmin

For example, if you want to store the password "sysadmin" for a Web service, it could be stored as follows:

```
sqlplus apps/apps@db @$FND_TOP/sql/afvltput.sql FND sysadmin
sysadmin
```

At run time, if event parameters are passed with the same names as the subscription parameters that have been parsed and stored, the event parameter values take precedence over subscription parameters.

For example, the event parameters are passed as follows:

- `BusinessEvent.setStringProperty("WFBES_SOAP_USERNAME",`

```
"SYSADMIN");
```

- `BusinessEvent.setStringProperty("WFBES_SOAP_PASSWORD_MOD", "FND" );`
- `BusinessEvent.setStringProperty("WFBES_SOAP_PASSWORD_KEY", "sysadmin");`

In summary, use the following steps to pass username and password for WS-Security during Web service invocation:

1. Pass the *username* to event or subscription parameter WFBES\_SOAP\_USERNAME.
2. Store the password in FND Vault using `$FND_TOP/sql/afvltput.sql` script through appropriate module and key.
3. Pass the module name and key to event or subscription parameters WFBES\_SOAP\_PASSWORD\_MOD and WFBES\_SOAP\_PASSWORD\_KEY.

## Calling Back to Oracle E-Business Suite With Web Service Response

As mentioned earlier that Oracle Workflow is the primary process management solution within Oracle E-Business Suite; Oracle Workflow Business Event System, an essential component within Oracle Workflow, provides event and subscription features that help identify integration points within Oracle E-Business Suite. Thus, to successfully invoke Web services from Oracle E-Business Suite requires highly integrated environment with Oracle Workflow.

To support synchronous request - response service operation, if a Web service has an output or a response message, service invocation framework uses the *callback* mechanism in Oracle Workflow to communicate the response message back to Oracle E-Business Suite through the Business Event System.

**Note:** A synchronous request - response message is a common message exchange pattern in Web service operation where a client asks a service provider a question and then waits for a response before continuing on. For more information, see: Understand Message Patterns, page 11-3.

This callback feature takes the invoker event's event key to enqueue the callback event to the specified inbound agent (the callback agent) for the response. In addition, if a workflow process invokes a Web service using "Raise" event activity and waits for Web service response using "Receive" event activity, the invoker event key should be same as the invoker and/or waiting workflow process's item key so that when callback is performed, the waiting workflow process is correctly identified by `WF_ENGINE.EVENT` API.

By using both the *callback* events and agents, Web service invocation can be integrated back with a waiting workflow process or any other module within Oracle E-Business



Suite. Web service invocation uses the following callback subscription or event parameters:

- WFBES\_CALLBACK\_EVENT

This parameter can have a valid business event to be raised upon completion of the Web service with the service output message as payload.

For example, it can be like:

```
WFBES_CALLBACK_EVENT=oracle.apps.wf.myservice.callback
```

- WFBES\_CALLBACK\_AGENT

This parameter can have a valid business event system agent to which the event with the service response message as payload can be enqueued.

**Important:** This parameter will work only if WFBES\_CALLBACK\_EVENT is not null, otherwise the output message is lost and there is no callback.

For example, it can be like the default inbound agent (or any other inbound queue) for Web service messages:

```
WFBES_CALLBACK_AGENT=WF_WS_JMS_IN
```

**Note:** If you have defined custom agents, you can also specify the custom agent name as the parameter value.

Since Web service output message is enqueued to the inbound agent mentioned in WFBES\_CALLBACK\_AGENT, it is required to set up a Workflow Agent Listener on the inbound agent (if it is not yet set up) in order to process the callback/receive business event messages.

**Note:** Callback event can be used as correlation ID when the response message is enqueued to a callback agent. This helps administrators to create specialized agent listeners on a callback agent to process callback events.

For example, if the callback event for a service invocation is `oracle.apps.wf.myservice.callback`, and the callback agent is `WF_WS_JMS_IN`, when this event is enqueued to `WF_WS_JMS_IN` upon a successful service invocation, the event `oracle.apps.wf.myservice.callback` is used as Correlation ID in `WF_WS_JMS_IN` to help create an agent listener to process that event.

At run time, if event parameters are passed with the same names as the subscription parameters that have been parsed and stored, the event parameter values take

precedence over subscription parameters. For example, the event parameters are passed as follows:

- `BusinessEvent.setStringProperty("WFBES_CALLBACK_EVENT", "oracle.apps.wf.myservice.callback");`
- `BusinessEvent.setStringProperty("WFBES_CALLBACK_AGENT", "WF_WS_JMS_IN");`

To use the callback feature during the service invocation, you must create a receive event and subscribe to the receive event. See: [Creating a Receive Event and Event Subscription \(Optional\)](#), page 11-17.

To better understand how to invoke a Web service, see [Example of Invoking a Web Service From a Workflow Process](#), page 11-33

## Invoking Web Services

Oracle Workflow Business Event System is a workflow component that allows events to be raised from both PL/SQL and Java layers. Therefore, the service invocation from Oracle E-Business Suite can be from PL/SQL or Java

### Service Invocation from PL/SQL

1. Application raises a business event using PL/SQL API `WF_EVENT.Raise`.

The event data can be passed to the Event Manager within the call to the `WF_EVENT.Raise` API, or the Event Manager can obtain the event data or message payload by calling the `Generate` function for the event if the data or payload is required for a subscription.

**Note:** See Oracle Workflow API Reference for information about `WF_EVENT.Raise` API.

2. Oracle Workflow Business Event System (BES) identifies that the event has a subscription with Java Rule Function `oracle.apps.fnd.wf.bes.WebServiceInvokerSubscription`.
3. The Business Event System enqueues the event message to `WF_JAVA_DEFERRED` queue. The Java Deferred Agent Listener then dequeues and executes the subscription whose Java rule function invokes the Web service.
4. If callback event and agent parameters are mentioned, the Web service response is communicated back to Oracle E-Business Suite using the callback information. The Java Deferred Agent Listener process that runs in Concurrent Manager (CM) tier invokes the Web service.

### Service Invocation from Java

1. Java Application raises a business event using Java method `oracle.apps.fnd.wf.bes.BusinessEvent.raise` either from OA Framework page controller/AMImpl or Java code running on Concurrent Manager tier.
2. Since the event is raised in Java where the subscription's seeded Java Rule Function `oracle.apps.fnd.wf.bes.WebServiceInvokerSubscription` is accessible, whether the rule function is executed inline or deferred is determined by the phase of the subscription.
  - If the invoker subscription is created with Phase  $\geq 100$ , the event is enqueued to `WF_JAVA_DEFERRED` queue.
  - If the invoker subscription is created with Phase  $< 100$ , the event is dispatched inline.

If the event is raised from OA Framework page, the dispatch logic executes (that uses WSIF to invoke the Web service) within `OACORE OC4J` container.

**Note:** If the Web service invoker event is raised from Java code in the middle tier, and the invoker subscription is synchronous with subscription phase  $< 100$ , then the Web service is invoked as soon as the event is raised, and if successful the response can be read by the calling application and is available immediately by using method `BusinessEvent.getResponseData()`.

If the event is raised from Java code with the subscription phase is  $\geq 100$  or if the event is raised from PL/SQL, the event message will be enqueued to `WF_JAVA_DEFERRED` queue. If the Web service has an output or a response message, callback event with callback agent is required to receive the output message into Oracle E-Business Suite.

The following sample Java code raises a business event that invokes Web service and reads the response in the same session:

```

package oracle.apps.fnd.wf.bes;

import java.sql.Connection;

import oracle.apps.fnd.common.AppsLog;
import oracle.apps.fnd.common.Log;
import oracle.apps.fnd.wf.bes.InvokerConstants;
import oracle.apps.fnd.wf.common.WorkflowContext;

public class InvokeWebService {

    static Log mLog;
    static WorkflowContext mCtx;

    public InvokeWebService() {
    }

    public static Connection getConnection(String dbcFile) {
        Connection conn = null;

        System.setProperty("dbcfile", dbcFile);
        WorkflowContext mCtx = new WorkflowContext();

        mLog = mCtx.getLog();
        mLog.setLevel(Log.STATEMENT);
        ((AppsLog)mLog).reInitialize();
        mLog.setModule("%");

        return mCtx.getJDBCCConnection();
    }

    public static void main(String[] args)
    {
        BusinessEvent event;
        Connection conn;
        conn = getConnection(args[0]);

        try {
            // Proxyt host and port requires to be set in Java
options
            System.setProperty("http.proxyHost", args[1]);
            System.setProperty("http.proxyPort", args[2]);

            event = new BusinessEvent
("oracle.apps.wf.IrepService.invoke", "eventKey1");

            // Input XML message for Web Service

            String input = null;
            input =
"<ns3:IntegrationRepositoryService_GetInterfaceFunctionByName
xmlns:ns3=\"http://xmlns.oracle.com/apps/fnd/rep/ws\"> \n"+
<fullName>SERVICEBEAN:/oracle/apps/fnd/rep/ws/IntegrationRepos
itoryService:getInterfaceFunctionByNameSERVICEBEAN:/oracle/apps/fnd/
rep/ws/IntegrationRepositoryService:getInterfaceFunctionByName</full
MethodName>\n"+
"</ns3:IntegrationRepositoryService_GetInterfaceFunctionByName>";
            event.setData(input);

            String headerPartMsg = "<ns1:SOAHeader
xmlns:ns1=\"http://xmlns.oracle.com/xdm/SYSTEM\" " " +

```

```

"env:mustUnderstand=\""0\"
xmlns:env=\""http://schemas.xmlsoap.org/soap/envelope/\"> \n" +
    " <ns1:MESSAGE_TYPE>XML</ns1:MESSAGE_TYPE>\n" +
    " <ns1:MESSAGE_STANDARD>OAG</ns1:MESSAGE_STANDARD>\n" +
    " <ns1:TRANSACTION_TYPE>PO</ns1:TRANSACTION_TYPE>\n" +
    "
<ns1:TRANSACTION_SUBTYPE>PROCESS</ns1:TRANSACTION_SUBTYPE>\n" +
    " <ns1:DOCUMENT_NUMBER>123</ns1:DOCUMENT_NUMBER>\n" +
    " <ns1:PARTY_SITE_ID>4444</ns1:PARTY_SITE_ID>\n" +
    "</ns1:SOAHeader>\n";
businessEvent.setStringProperty("WFBES_INPUT_header",
headerPartMsg);

        event.raise(conn);
        conn.commit();

        Object resp = event.getResponseData();
        if (resp != null) {
            System.out.println(resp.toString());
        }
        else {
            System.out.println("No response received");
        }
    }
    catch (Exception e) {
        System.out.println("Exception occurred " +
e.getMessage());
        e.printStackTrace();
    }
}
}

```

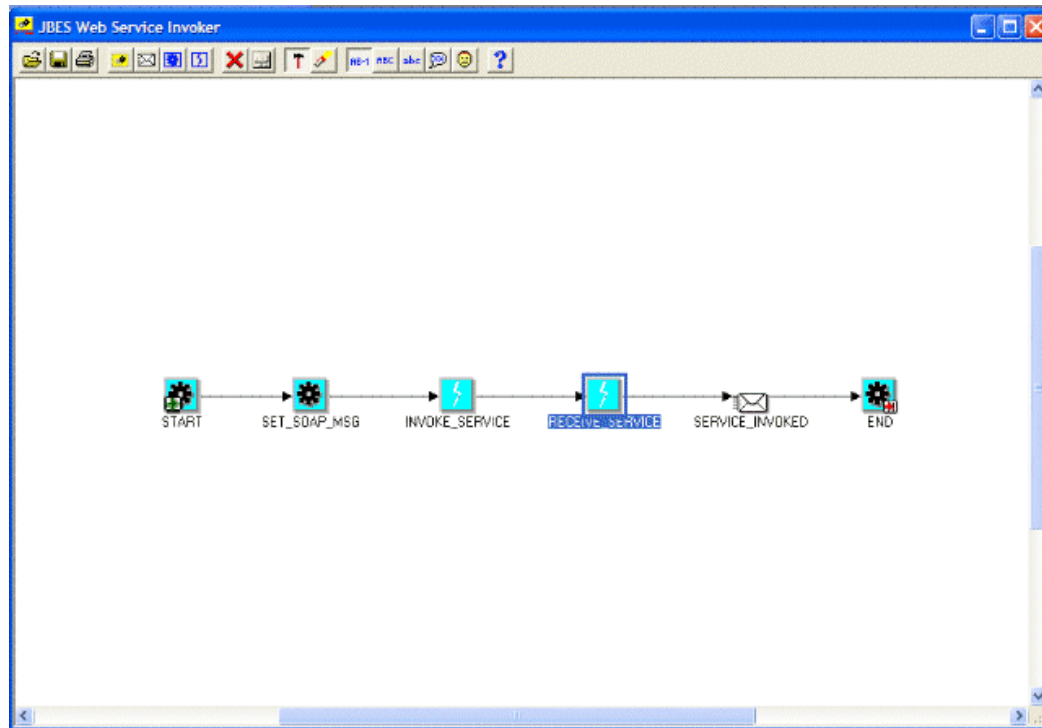
## Example of Invoking a Web Service From a Workflow Process

The following example is to invoke a Web service through launching a workflow process including the following nodes or activities:

- An invoker business event to invoke a Web service.  
For example, INVOKE\_SERVICE is an event activity with event action "Raise".
- A receive business event to receive a response or Web service output message.  
For example, RECEIVE\_SERVICE is an event activity with event action "Receive".
- Other activities could be used in the process for XML message processing, notifying users of Web service invocation response, regular transaction processing and so on.  
For example, SERVICE\_INVOKED is a notification activity to send a notification message when a Web service is successfully invoked.

The following workflow process diagram illustrates the service invocation process flow:

### Workflow Process Diagram to Invoke a Web Service



### Defining Service Invocation Metadata

To define the service invocation metadata with the callback feature, you must have the following necessary event and subscription in place:

1. An invoker event, such as INVOKE\_SERVICE in the workflow diagram.  
This activity is used to pass the event XML payload to be used as SOAP body and other required event parameters required for Web service invocation as already discussed.  
See: Creating a Web Service Invoker Business Event, page 11-6.
2. Local and error event subscriptions to the invoker event. See: Creating Local and Error Event Subscriptions to the Invoker Event, page 11-8.
3. A receive event (such as RECEIVE\_SERVICE in the workflow diagram) and the External subscription to the receive event.

**Important:** The receive event is raised with the same event key as the event key for invoker event. It is important that the waiting workflow process's item key and the invoker event's event key are the same.

If callback event and agent parameters are set, this activity waits for the receive event to occur after Web service invocation is successful.

See: Creating a Receive Event and Event Subscription (Optional), page 11-17.

### Verifying Workflow Agent Listener Status

In order to process a Web service response message from the inbound agent, you need to verify if a Workflow Agent Listener is running on that agent.

Use the following steps for verification:

1. Log on to Oracle Workflow with Oracle Workflow Web Administrator responsibility.
2. From the navigation menu, select Oracle Applications Manager, and click the Workflow Manager link.
3. Click the Agent Listener status icon to open the Service Components page.
4. Locate the Workflow Agent Listener that you use for the callback agent listener. For example, locate the 'Workflow Inbound JMS Agent Listener' for processing a Web service response message to ensure it is up and running.

#### Validating a Workflow Agent Listener's Status

The screenshot displays the Oracle Applications Manager interface. At the top, there's a navigation bar with 'Support Cart', 'Setup', 'Home', 'Logout', and 'Help'. Below this, the 'Applications Dashboard' and 'Site Map' are visible. The main content area is titled 'Service Components: r121xb7a'. It shows a table with columns: Name, Status, Type, Startup Mode, Container Type, Container, and Actions. Two entries are listed: 'Workflow Inbound JMS Agent Listener' and 'Workflow Inbound Notifications Agent Listener', both with a status of 'Running'. The 'Actions' column for each entry includes 'Refresh' and 'Go' buttons. A tip at the bottom left states 'TIP GSM = Generic Service Management'.

Select	Name	Status	Type	Startup Mode	Container Type	Container	Actions
<input checked="" type="radio"/>	Workflow Inbound JMS Agent Listener	Running	Workflow Agent Listener	Automatic	Oracle Applications GSM	Workflow Agent Listener Service	Refresh Go
<input type="radio"/>	Workflow Inbound Notifications Agent Listener	Running	Workflow Agent Listener	Automatic	Oracle Applications GSM	Workflow Agent Listener Service	Refresh Go

After the verification, you can launch the workflow process to invoke a Web service with a callback response through Oracle Workflow. You can also validate the process by reviewing the progress status of each activity contained in your workflow process diagram.

When the Web service has been successfully invoked from the automated workflow process, you should receive a workflow notification message if the notification activity is included in the process.

## Receiving a Notification Message

The screenshot displays the Oracle Administrator Workflow web interface. At the top, there's a blue header with the Oracle logo and 'Administrator Workflow'. Below the header, a breadcrumb trail shows 'Status Monitor > Monitor Search > Monitor Activities History >'. The main content area is titled 'Web Service Invoked' and shows details for a message sent to 'SYSADMIN SYSADMIN' on '17-Jun-2008 20:42:44'. The message ID is '3625284'. A table-like structure shows the response details: 'Response Code' is '200', 'Response Message' is '4FE5470E1D895527E040B98BCA150507', and 'Response Info' is 'Document received and pushed into queue for asynchronous processing. Enqueued message id is '4FE5470E1D895527E040B98BCA150507''. At the bottom, there's a 'Return to Worklist' link and a footer with 'About this Page', 'Privacy Statement', 'Diagnostics', 'Home', 'Logout', 'Preferences', 'Help', and 'Copyright (c)'.

For more information on how to create and launch a workflow, see *Oracle Workflow Developer's Guide*.

## Managing Errors

Service invocation framework uses the same way of handling errors in Business Event System to manage errors occurred during the execution of business event subscriptions. If the service invocation returns a fault message, the event is enqueued to error queue to trigger error processing. If an exception occurred during invocation process is due to service unavailability, the service faults should be logged and error subscription should be invoked.

To effectively process run-time exceptions for the events that are enqueued to an error queue, service invocation framework uses the following event ERROR process to specifically trigger error processing during the service invocation:

- DEFAULT\_EVENT\_ERROR2: Default Event Error Process (One Retry Option)

**Note:** The DEFAULT\_EVENT\_ERROR2 Error workflow process is created under WFERROR itemtype.

For example, if there is a run-time exception when the Workflow Java Deferred Agent Listener executes event subscription to invoke the Web service, the event is enqueued to WF\_JAVA\_ERROR queue. If the event has an Error subscription defined to launch Error workflow process WFERROR:DEFAULT\_EVENT\_ERROR2, the Workflow Java Error Agent Listener executes the error subscription which sends a notification to SYSADMIN with Web service definition, error details and event details. Since Oracle Workflow default event error handler provides options for SYSADMIN to retry the Web service invocation process after verifying that the reported error has been corrected, SYSADMIN can invoke the Web service again from the notification if necessary.

However, if there is a run-time exception when invoking the Web service by raising the Invoker event with synchronous subscription (phase <100), the exception thrown to the



calling application. It is the responsibility of the calling application to manage the exception.

### Enabling Error Processing During Service Invocation

To enable the error processing feature during the service invocation, you must create an Error subscription with the following values:

- 'Error' source type
- 'Launch Workflow' action type
- 'WFERROR:DEFAULT\_EVENT\_ERROR2' workflow process

### Create an Error Subscription With 'Launch Workflow' Action Type

The screenshot shows the Oracle Workflow Administrator web application. The title bar reads "ORACLE® Administrator Workflow" with navigation links for "Diagnostics", "Home", "Logout", "Preferences", and "Help". The breadcrumb trail is "Business Events: Subscriptions > Create Event Subscription >". The page title is "Create Event Subscription - Launch Workflow".

At the top right are buttons for "Cancel", "Back", and "Apply". Below this is a note: "An event subscription can be routed to a Workflow process. Please specify the Workflow Type and Workflow Process to be launched." followed by an asterisk indicating a required field.

The "Action" section contains three required fields: "Workflow Type" (set to "WFERROR"), "Workflow Process" (set to "DEFAULT\_EVENT\_ERROR2"), and "Priority" (set to "Normal"). A hint for the Workflow Process field says "Choose a Workflow Type, before choosing the Workflow Process for that Type". There is also an "Additional Options" dropdown menu.

The "Subscription Parameters" section has a "Select Object:" dropdown set to "Delete". Below it are "Select All" and "Select None" links. A table with two columns, "Select Name" and "Value", is shown with one empty row. An "Add Another Row" button is at the bottom of the table. A note below the table says "Enter parameters and their values with no spaces".

The "Documentation" section contains two required fields: "Owner Name" (set to "Oracle Workflow") and "Owner Tag" (set to "FND"). Below these are "Customization Level" (set to "User") and a "Description" text area.

At the bottom right are buttons for "Cancel", "Back", and "Apply".

To access the Create Event Subscription page, log on to Oracle E-Business Suite with the Workflow Administrator Web Applications responsibility. Select the Business Events link and choose the Subscriptions subtab. In the Event Subscriptions page, click **Create Subscription**.

Detailed information on how to create an error subscription for service invocation, see Create an Error subscription with 'Launch Workflow' Action Type, page 11-15.

## Testing Web Service Invocation

Service invocation framework uses the Oracle Workflow Test Business Event page to

check the basic operation of Business Event System by raising a test event from either Java or PL/SQL layer and executing synchronous and asynchronous subscriptions to that event. This testing feature provides a flexible mechanism which easily lets you validate whether a Web service can be successfully invoked from concurrent manager tier and OACORE OC4J.

You can test a Web service invocation using one of the following ways:

- Using the Test Business Event Page to Manually Raise an Event, page 11-38
- Using Command Line to Raise an Event, page 11-42

## Using the Test Business Event Page

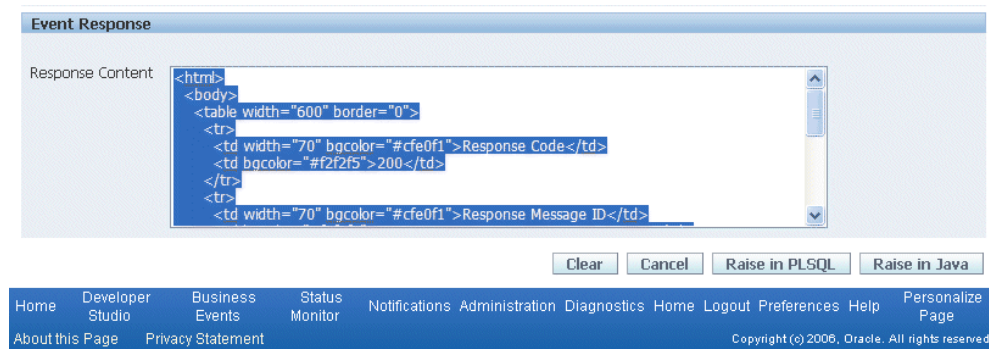
Use the Test Business Event page to test a event by raising it from both PL/SQL API and Java method.

- For an invoker event raised using **Raise in Java** option, the Web service is invoked from OACORE OC4J if the subscription phase < 100.

If the Web service is successfully invoked, the Test Business Event page reloads and displays the XML Response region right after the XML Content field.

If there is a run-time exception when invoking the Web service using synchronous subscription, the exception message is shown on the Test Business Event page.

### Displaying XML Response for Successful Service Invocation



- For an invoker event raised using **Raise in PLSQL** option, the Web service is invoked from the concurrent manager tier. The raised event will be enqueued to WF\_JAVA\_DEFERRED and then dispatched by Workflow Java Deferred Agent Listener.

The seeded Java rule function uses the callback event and agent to communicate the response or Web service output message back to Oracle E-Business Suite through Business Event System.

**Note:** Since Java Deferred Agent Listener is responsible for dispatching the subscription and invoking Web services from concurrent manager tier, please ensure that Workflow Java Deferred Agent Listener is up and running.

To validate, log on to Oracle Applications Manager and select the Workflow Manager link. Choose Agent Listeners and search on Workflow Java Deferred Agent Listener to view its status.

### Testing Service Invocations

After logging on to Oracle Workflow with the Workflow Administrator Web responsibility. Select the Business Events link to search for an event that you want to test. From the search result table, click the **Test** icon next to the event you want to raise. This opens the Test Business Event page where you can raise the event with an unique event key. Enter event parameters for the invoker event subscription and a valid XML message that complies with input message schema. The Test Business Event page will also display response XML message if appropriate.

Please note that the Test Business Event page will retain all the data entered. Therefore, if there is a need to raise another event, you must click **Clear** to clear all data that you have entered.

Following parameters may be specified when raising the event from the Test Business Event page to invoke a Web service:

- Message transformation: XSL transformation for Web service input message and output message
  - WFBES\_OUT\_XSL\_FILENAME
  - WFBES\_IN\_XSL\_FILENAME
- WS-Security: Information required to add UsernameToken header to a SOAP request
  - WFBES\_SOAP\_USERNAME
  - WFBES\_SOAP\_PASSWORD\_MOD
  - WFBES\_SOAP\_PASSWORD\_KEY
- Input Message part value: Pass values for any part that may be required to embed application context into SOAP envelopes
  - WFBES\_INPUT\_<partname>

**Note:** The WFBES\_INPUT\_<partname> parameter can only be passed at run time during event raise.

- Callback: Callback to Oracle E-Business Suite with Web service response
  - WFBES\_CALLBACK\_EVENT
  - WFBES\_CALLBACK\_AGENT
- SOAP Body:
  - XML Input message (Required)

For information about these parameters, see:

- Understanding Web Service Input Message Parts, page 11-21
- Supporting WS-Security, page 11-26
- Calling Back to Oracle E-Business Suite With Web Service Response, page 11-28

### Testing Invocation with Callback Required

If you want to test an invocation with callback to Oracle E-Business Suite, then you must enter the following parameters and values:

- WFBES\_CALLBACK\_EVENT: receive event
- WFBES\_CALLBACK\_AGENT: WF\_WS\_JMS\_IN (or any other Inbound Queue as the value)

Please note that for testing from the Test Business Event page, since the XML message is prewritten and entered in the XML Content field, if there is an error in the input XML message, the error notification will not provide you with an option to correct it before retrying the process.

Before testing the invocation, for easier debugging or troubleshooting purposes throughout the test, you can enable the diagnostics and logging feature to directly display on-screen logs in the test page. For instructions on how to turn on this logging feature, see Troubleshooting invocation failures on OACORE OC4J, page 11-43.

### To test an event invocation:

1. Log on to Oracle E-Business Suite with the Workflow Administrator Web responsibility and select the Business Events link.
2. Search on a business event that you want to run the test, such as `oracle.apps.wf.xmlgateway.invoke` and click **Go**.

3. Select the business event that you want to raise from the result table and click the **Test** icon to open the Test Business Event page.
4. Enter a unique event key in the Event Key field and leave the Send Date field blank.
5. Enter appropriate parameters in the Enter Parameters region.
6. In the Event Data region, enter the following information:
  - Upload Option: Write XML
  - XML Content: Enter appropriate XML information as input message. For example, you can enter:

```
<ReceiveDocument xmlns="http://xmlns.oracle.com/xdm/SYSTEM">
  <PO_DOCUMENT>
    <PO_NUM>12345</PO_NUM>
    <PO_TYPE>standards</PO_TYPE>
  </DESCRIPTION>
</PO_DOCUMENT>
</ReceiveDocument>
```

**ORACLE Administrator Workflow**

Home | Developer Studio | **Business Events** | Status Monitor | Notifications | Administration

Business Events: Events > Business Events: Events >

**Test Business Event**

To test a business event enter an event key, a list of parameter name / value pairs ( optional ), and either paste in an XML Document or upload from the local file system and press submit.

\* Indicates required field

**Event Identifier**

\* Event Name oracle.apps.wf.xmlgateway.invoke

\* Event Key xmlglk2

Send Date

Send Date must be in the Format: DD-MON-RRRR

**Event Parameters**

Select Object: Delete

Select All | Select None

Select Label	Value
<input type="checkbox"/> WFBES_INPUT_header	ns1:SOAHeader xmlns:ns1="http://xmlns.oracle.com/xdm/SYSTEM"
<input type="checkbox"/> WFBES_OUT_XLS_FILENAME	XMLGResponse.xls:FND:1.0

Add Another Row

**Event Data**

Upload Option Write XML

XML Content

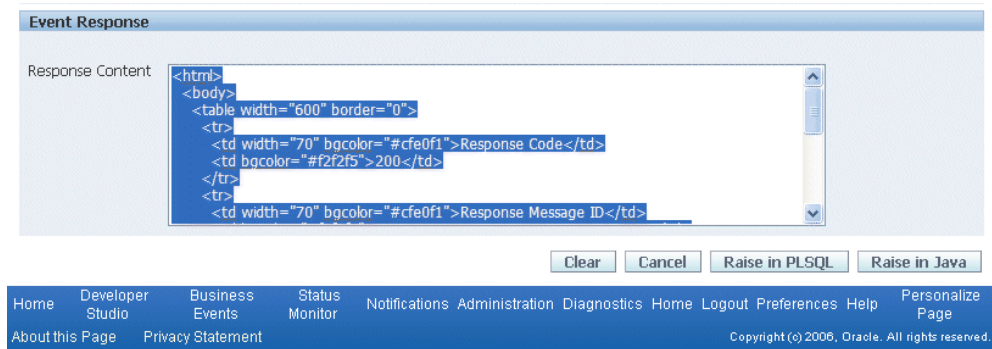
```
<ReceiveDocument xmlns="http://xmlns.oracle.com/xdm/SYSTEM">
  <PO_DOCUMENT>
    <PO_NUM>12345</PO_NUM>
    <PO_TYPE>standards</PO_TYPE>
  </DESCRIPTION>
</PO_DOCUMENT>
</ReceiveDocument>
```

If XML content is greater than 4000 characters, Select "Upload XML" option.

Clear Cancel Raise in PLSQL Raise in Java

7. Click **Raise in Java** to raise an event from OACORE OC4J.

If the Web service is successfully invoked, the Test Business Event page reloads and displays the XML Response region right after the XML Content field.



8. Click **Raise in PLSQL** to raise an event is from the concurrent manager tier.

For more information about testing business events, see *To Raise a Test Event, Oracle Workflow Developer's Guide*.

## Using Command Lines

You can also use the command line API based test method to raise both PL/SQL or Java based events.

- For PL/SQL based events, use PL/SQL `WF_EVENT.Raise` API to test Web service invocation from the concurrent manager tier JVM. Java Deferred Agent Listener dispatches the subscription and invokes Web services from concurrent manager tier.

**Note:** Since Java Deferred Agent Listener is responsible for dispatching the subscription and invoking Web services from concurrent manager tier, please ensure that Workflow Java Deferred Agent Listener is up and running.

To validate, log on to Oracle Applications Manager and select the Workflow Manager link. Choose Agent Listeners and search on Workflow Java Deferred Agent Listener to view its status.

- For Java based Web events, use Java method `oracle.apps.fnd.wf.bes.BusinessEvent.raise` to test Web service invocation.

For example, we could have a test class

`oracle.apps.fnd.wf.bes.WFInvokerTestCase` with classpath set to `$AF_CLASSPATH`.

```
java oracle.apps.fnd.wf.bes.WSInvokerTestCase <DBC file> <proxy
host> <proxy port>
```

## Troubleshooting Web Service Invocation Failure

Web services can be invoked from any one of following tiers:

- **OACORE OC4J:** Web service invocations from OA Framework page using a synchronous event subscription (phase < 100) is executed from within the OACORE OC4J container.
- **Concurrent Manager (CM) Tier JVM:** The following Web service invocations are executed from CM tier JVM within Java Deferred Agent Listener that runs within Workflow Agent Listener Service:
  - Invocations from PL/SQL either through synchronous or asynchronous event subscriptions
  - Invocations from Java/OA Framework through synchronous event subscriptions
- **Standalone JVM:** Web service invocations from a Java process that runs outside OACORE or CM using a synchronous event subscription executes from within that JVM.

In most cases, the Web service resides outside the firewall and the executing host does not have direct access to the WSDL or the Web service endpoint to send the SOAP request. Without properly setting up and configuring the proxy parameters for each tier that Web service invocations occur, WSDL files will not be parsed and consumed during subscription or Web services will not be successfully invoked.

How to set up proxy host and port appropriately at each layer, see detailed information described in the Setup Tasks, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

At run time, if a Web service invocation fails, an exception is thrown and the invoker event is enqueued to WF\_ERROR queue. Since the Web service can be invoked from any one of the layers described earlier, how to troubleshoot and resolve the failure invocation can be discussed as follows based on layer that Web service invocations occur:

- Troubleshooting invocation failure on OACORE OC4J, page 11-43
- Troubleshooting invocation failure on Concurrent Manager (CM) Tier JVM, page 11-47
- Troubleshooting invocation failure on Standalone JVM, page 11-47

## Troubleshooting Invocation Failure on OACORE OC4J

For the purposes of easier debugging or troubleshooting throughout a test run of the

Web service invocation from within an OA Framework page, on-screen logging mechanism should be used.

### Enabling On-screen Logging

You can enable the on-screen logging feature and have the logs directly displayed at the bottom of the Test Business Event page. These logs provide processing details while executing the code to invoke the Web service.

If there is a fault or a run-time exception in processing the event and invoking the service, the on-screen logging quickly discloses what is happening.

Enabling on-screen logging involves the following two steps:

1. Setting FND: Diagnostics Profile Option, page 11-44
2. Displaying On-screen Logging, page 11-44

### Setting FND: Diagnostics Profile Option

Before using the Test Business Event page, first set the *FND: Diagnostics* profile option to 'Yes' at an appropriate level to enable the Diagnostics link on the global menu of the HTML-based application pages.

**Note:** Through the Diagnostics link, we can enable database trace, profiling, and on-screen logging that will help troubleshooting the transactions performed from the HTML-based application pages.

#### Setting FND: Diagnostics Profile Option

The screenshot shows a window titled 'System Profile Values' with a table of profile options. The table has five columns: Profile Option Name, Site, Application, Responsibility, and User. The first row shows 'FND: Diagnostics' in the 'Profile Option Name' column and 'Yes' in the 'Site' column. The rest of the table is empty.

Profile Option Name	Site	Application	Responsibility	User
FND: Diagnostics	Yes			

With the diagnostics feature, the on-screen logging can be enabled which helps us track the `WebServiceInvokeSubscription`'s log messages when an invoker event is raised from the Test Business Event page and subsequently the Web service is invoked.

### Displaying On-screen Logging

After setting the FND: Diagnostics profile option to 'Yes', you should find the



Diagnostics link available in the upper right corner of your HTML page.

By selecting the Diagnostics link and entering appropriate information, the on-screen logging feature can be enabled. Once you locate a desired event and test its invocation, relevant log messages directly appear at the bottom of your test page for an easier debugging or troubleshooting if needed.

**Important:** If the *FND: Diagnostics* profile option is not set to 'Yes', then the Diagnostics link will not be visible as a global menu for selection.  
See: Setting FND: Diagnostics Profile Option, page 11-44.

**To display on-screen logs while testing your service invocation in the Test Business Event page:**

1. Log on to Oracle Workflow with appropriate responsibility, and select the Business Events link to locate an invoker business event that you want to run the test, such as `oracle.apps.wf.xmlgateway.invoke` and click **Go** to perform a search.
2. From the search result table, select the business event that you want to raise and click the **Test** icon to open the Test Business Event page.
3. Click the Diagnostics link in the upper right corner of the page.
4. Enter the following information to enable the on-screen logs:
  - Diagnostics: Show Log on Screen
  - Log Level: Statement (1)
  - Module: %
5. Click **Go**. The on-screen logging is now enabled.

ORACLE®

[Diagnostics](#)
[Home](#)
[Logout](#)
[Preferences](#)

Information

Screen logging is now enabled. Please navigate to the page you want to debug, and you will see the log messages appended at the bottom of the page.  
  
Note: Screen Logging provides quick Java UI logging, and is independent of persistent (File/Database) Logging. For more comprehensive (Java, PL/SQL) persistent logging please configure and use 'Show Log'.

[Oracle Diagnostics](#)

Diagnostic:

Log Level:

☐ Unexpected (6)
☐ Error (5)
☐ Exception (4)
☐ Event (3)
☐ Procedure (2)
☒ Statement (1)
☐ Turn off screen logging

Module:

Go

[About this Page](#)
[Privacy Statement](#)

[Diagnostics](#)
[Home](#)
[Logout](#)
[Preferences](#)

Copyright (c) 2006, Oracle. All rights reserved.

Request Parameters

evtSrcRowId=  
Diagnostic=2  
\_t1=320250339  
oapc=6

- Navigate to the Test Business Event page and raise an event to execute the invocation testing.

## Review On-Screen Log Messages

After you have enabled the on-screen logging feature, during the testing, you should find relevant log messages displayed at the bottom of the Test Business Event page. This provides the detailed information of all processing by the code that invokes the Web service.

For example, you can review `WebServiceInvokerSubscription` log messages displayed on the same page to verify the service execution status, exception or fault if there is any, and whether the callback succeeded or not.

The following example log indicates that the service execution is completed with callback response message enqueued to `WF_WS_JMS_IN` inbound queue if the `'WFBES_CALLBACK_EVENT'` parameter value is set to receive event and `'WFBES_CALLBACK_AGENT'` parameter value is set to `'WF_WS_JMS_IN'`:

### WebServiceInvokerSubscription Logs

```

5351]:PROCEDURE:[fnd.wf.bes.QueueHandlerInvoker]:enqueue() BEGIN
5351]:PROCEDURE:[fnd.wf.bes.PLSQLQueueHandler]:enqueue() :BEGIN (BusinessEvent{name=Receive event, key=002, priority=1, corr
5351]:EXCEPTION:[fnd.wf.bes.PLSQLQueueHandler]:prepareEnqueueStatement() : Successfully prepared enqueue statement. Call=(
5749]:EXCEPTION:[fnd.wf.bes.PLSQLQueueHandler]:enqueue() : Enqueued the following BusinessEvent -> BusinessEvent{name=Receive
5750]:EXCEPTION:[fnd.wf.bes.QueueHandlerInvoker]:enqueue() Enqueued the following BusinessEvent -> BusinessEvent{name=Receive
5750]:STATEMENT:[fnd.wf.bes.WebServiceInvokerSubscription]:performCallback():Enqueued response to WF_WS_JMS_IN
5750]:PROCEDURE:[fnd.wf.bes.WebServiceInvokerSubscription]:performCallback():END
5750]:PROCEDURE:[fnd.wf.bes.WebServiceInvokerSubscription]:postInvokeService():END
5750]:STATEMENT:[fnd.wf.bes.WebServiceInvokerSubscription]:onBusinessEvent():Service invocation complete
5750]:PROCEDURE:[fnd.wf.bes.WebServiceInvokerSubscription]:onBusinessEvent():END

```

For detailed information on how to enable the logging feature, see [Enabling On-Screen Logging](#), page 11-43.

## Troubleshooting Invocation Failure on Concurrent Manager (CM) Tier JVM

To troubleshoot Web service invocation failure on Concurrent Manager (CM) Tier JVM, you must ensure that the Error subscription is created for the all Web service invoker events to capture complete exception details when invocation happens from Workflow Java Deferred Agent Listener.

### Error Subscription

For all Web service invoker events, error subscription is required to enable error processing in the Business Event System that is used to communicate with SYSADMIN user of an error condition in subscription execution. It sends a workflow notification to SYSADMIN with Web service definition, error details, and event details allowing the SYSADMIN to process the errors if needed.

For example, if an error occurs during the invocation and the event is enqueued to WF\_JAVA\_ERROR queue, with an Error subscription defined to launch Error workflow process WFERROR:DEFAULT\_EVENT\_ERROR2, the Workflow Java Error Agent Listener executes the error subscription which sends a notification to SYSADMIN with Web service definition, error details and event details.

For more information, see Managing Errors, page 11-36.

### Enabling Workflow Java Deferred Agent Listener Logging

Since Oracle Workflow default event error handler provides options for SYSADMIN to retry the Web service invocation process after verifying that the reported error has been corrected, SYSADMIN can invoke the Web service again from the notification if necessary. However, if further analysis of the steps leading to the exception is required, use Workflow Java Deferred Agent Listener logging mechanism to set STATEMENT level log for Workflow Java Deferred Agent Listener and retry the failed Web service invocation to obtain detailed steps leading to the exception.

For more information, see Java Agent Listeners, *Oracle Workflow Administrator's Guide*.

## Troubleshooting Invocation Failure on Standalone JVM

When invoking a Web service from a Java process that runs outside OACORE or CM by calling `BusinessEvent.raise` method to raise the invoker event with a synchronous 'Invoke Web Service' subscription, the following situation can occur:

- If the invocation is successful, the method returns the response message.
- If there was a run-time exception, `BusinessEventException`, thrown by the method that could be used to get the complete stack trace.

For details, see the sample Java code in the Service Invocation from Java section, Invoking Web Services, page 11-30.

## Extending Seeded Java Rule Function

Oracle E-Business Suite Integrated SOA Gateway allows developers to extend the invoker subscription seeded rule function

`oracle.apps.fnd.wf.bes.WebServiceInvokerSubscription` using Java coding standards for more specialized processing.

Developers can extend the seeded rule function to override following methods:

- `preInvokeService`
- `postInvokeService`
- `invokeService`
- `addWSSecurityHeader`
- `setInputParts`

For detailed information about these methods, see *Oracle Workflow API Reference*.

### **preInvokeService**

This method is used for pre processing before Web service invocations.

```
protected String preInvokeService(Subscription eo,
                                   BusinessEvent event,
                                   WorkflowContext context)
throws BusinessEventException;
```

The Web service input message or request message is available by calling `event.getData()`. This is the business event payload passed when raising the invoker event or generated by business event Generate function.

This method can perform additional processing on the request data if required. The default implementation through the seeded Java rule function performs XSL transformation using the XSL file specified in `WFBES_IN_XSL_FILENAME` if input payload message is available.

### **postInvokeService**

```
protected void postInvokeService(Subscription eo,
                                   BusinessEvent event,
                                   WorkflowContext context,
                                   String requestData,
                                   String responseData)
throws BusinessEventException;
```

If the operation is synchronous request - response, the response is available in parameter `responseData`.

This method performs additional processing on the response and update application state if required. The default implementation through seeded Java rule function

performs the following tasks:

- XSL transformation on a response or Web service output message based on WFBES\_OUT\_XSL\_FILENAME
- Call back to Workflow Business Event System based on WFBES\_CALLBACK\_EVENT and WFBES\_CALLBACK\_AGENT parameter values

## invokeService

```
protected String invokeService(String wsdlLocation,  
                               String serviceName,  
                               String invokePort,  
                               String portTypeName,  
                               String operationName,  
                               String eventData)  
  
throws Exception;
```

This `invokeService` method provides the implementation that makes use of Web service invocation metadata and invokes the Web service using WSIF APIs. This method can be overridden if a different implementation is required other than WSIF to invoke Web services.

## addWSSecurityHeader

```
protected void addWSSecurityHeader(ArrayList headersList) throws  
Exception;
```

This method adds WS-Security compliant header to the SOAP request. The default implementation through Java seeded rule function adds *UsernameToken* element to the security header based on event parameters WFBES\_SOAP\_USERNAME, WFBES\_SOAP\_PASSWORD\_MOD, and WFBES\_SOAP\_PASSWORD\_KEY. This method can be overridden to add any WS-Security header or have custom logic to retrieve username and password to build UsernameToken element. The well-formed XML Element should be added to the ArrayList.

The following code snippet shows WS-Security added to a SOAP header:

```

try {
    DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
    factory.setNamespaceAware(true);
    DocumentBuilder bldr = factory.newDocumentBuilder();
    Document doc = bldr.newDocument();

    Element sec = doc.createElement("wsse:Security");
    Attr attr = doc.createAttribute("xmlns:wsse");

    attr.setValue("http://docs.oasis-open.org/wss/2004/01/oasis-200401-wssws
security-secext-1.0.xsd");
    sec.setAttributeNode(attr);
    doc.appendChild(sec);

    Element unt = doc.createElement("wsse:UsernameToken");
    sec.appendChild(unt);
    .... build XML message ....
    }
    catch (Exception e) {
    }
}
headersList.add(doc.getDocumentElement());

```

## setInputParts

```

protected void setInputParts(WSIFMessage inputMessage, Input input, String
eventData) throws Exception

```

This `setInputParts` method supports setting input part values such as header, body, or other parts, that are defined for the Web service operation's input message. The default implementation through Java seeded rule function adds the event data payload as the body of the input message. It also adds any other parts provided as event parameters in the triggering event.

The event parameters that contain input message parts must be identified by parameter names with the following format:

```
WFBES_INPUT_<partname>
```

This method can be overridden to set specific input parts that you require or to set values for RPC (remote procedure call) style Web service invocation.

**Important:** The service invocation framework supports invoking document-based Web service only. The RPC style Web service invocation is not naturally supported in this release unless you extend this method to set input part values for RPC style.

The following code snippet shows how this method is used to set values for document style Web service invocation:

```

protected void setInputParts(WSIFMessage inputMessage, Input input,
String eventData)
throws Exception {
    BusinessEvent event = this.getBusinessEvent();
    String bindingStyle = this.getBindingStyle();
    if (bindingStyle.equalsIgnoreCase("document") {
        String headerPartMsg =
event.getStringProperty("WFBES_INPUT_header");
        // header part
        inputMessage.setObjectPart("header",
getDocumentElement(headerPartMsg));
        // body part
        inputMessage.setObjectPart("body", getDocumentElement(eventData));
    else {
        // Web service style is RPC
        // Code can be added to set input parts for RPC style invocation
    }
}

```

## Sample Codes

The following code shows how to extend the seeded Java rule function:

```

package oracle.apps.fnd.wf.bes;

import java.io.ByteArrayInputStream;

import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import oracle.apps.fnd.common.Log;
import oracle.apps.fnd.wf.bes.server.Subscription;
import oracle.apps.fnd.wf.common.WorkflowContext;

import org.apache.wsif.WSIFConstants;
import org.apache.wsif.WSIFMessage;
import org.apache.wsif.WSIFOperation;

import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class MyWebServiceInvoker
    extends WebServiceInvokerSubscription
    implements SubscriptionInterface
{
    private static final String CLASS_PREFIX =
        MyWebServiceInvoker.class.getName() + ".";

    public MyWebServiceInvoker()
    {
    }

    final protected String invokeService( String wsdlLocation,
        String serviceName,
        String invokePort,
        String portTypeName,
        String operationName,
        String eventData)
        throws Exception {
        super.preInvokeService(wsdlLocation, serviceName, invokePort,
            portTypeName, operationName, eventData);

        // Perform special pre invocation processing like updating application
        // state and so on.
    }

    final protected void postInvokeService(Subscription eo,
        BusinessEvent event,
        WorkflowContext context,
        String requestData,
        String responseData)
        throws Exception {

        super.postInvokeService(eo, event, context, requestData,
            responseData);

        // Perform special post invocation processing like updating application
        // state and so on.
    }
}

```



```

/**
 * Implementing addSOAPHeaders method to include custom header
 * required to invoke an EBS Web Service.
 */
final protected void addSOAPHeaders(WSIFOperation operation)
throws Exception {
    String METHOD_NAME = "addSOAPHeaders";
    mLog.write(CLASS_PREFIX + METHOD_NAME, "BEGIN", Log.PROCEDURE);

    WSIFMessage hdrMsg = operation.getContext();
    ArrayList hdr = new ArrayList();

    // Call seeded implementation to add WS-Security header
    super.addWSSecurityHeader(hdr);

    // Add my own Custom header
    mLog.write(CLASS_PREFIX + METHOD_NAME, "Adding Custom header",
    Log.STATEMENT);
    addMyCustomHeader(hdr);

    // Set the headers to WSIFOperation
    hdrMsg.setObjectPart(WSIFConstants.CONTEXT_REQUEST_SOAP_HEADERS,
    hdr);
    operation.setContext(hdrMsg);

    mLog.write(CLASS_PREFIX + METHOD_NAME, "END", Log.PROCEDURE);
}

final protected void addMyCustomHeader(ArrayList headersList)
throws Exception {
    String METHOD_NAME = "addMyCustomHeader";
    // Adding special Custom Header to the WSIF's SOAP request
    String custHdr =
"ns1:SOAHeader
xmlns:ns1=\"http://xmlns.oracle.com/xdb/SYSTEM\" " +
"env:mustUnderstand=\"0\"
xmlns:env=\"http://schemas.xmlsoap.org/soap/envelope/\"> \n" +
"  <ns1:MESSAGE_TYPE>XML</ns1:MESSAGE_TYPE>\n" +
"  <ns1:MESSAGE_STANDARD>OAG</ns1:MESSAGE_STANDARD>\n" +
"  <ns1:TRANSACTION_TYPE>PO</ns1:TRANSACTION_TYPE>\n" +
"  <ns1:TRANSACTION_SUBTYPE>PROCESS</ns1:TRANSACTION_SUBTYPE>\n" +
"  <ns1:DOCUMENT_NUMBER>123</ns1:DOCUMENT_NUMBER>\n" +
"  <ns1:PARTY_SITE_ID>4444</ns1:PARTY_SITE_ID>\n" +
"  </ns1:SOAHeader>\n";

    if (custHdr != null && !"".equals(custHdr)) {
        try {
            DocumentBuilderFactory factory =
            DocumentBuilderFactory.newInstance();
            factory.setNamespaceAware(true);
            DocumentBuilder bldr = factory.newDocumentBuilder();
            Document doc = bldr.newDocument();

            doc = bldr.parse(new
            ByteArrayInputStream(custHdr.getBytes()));

            // Add the element to the Headers list
            headersList.add((Element)doc.getFirstChild());
        }
        catch (Exception e) {
            throw new BusinessEventException(

```

```

"Exception when creating header element - "+e.getMessage());
    }
}
mLog.write(CLASS_PREFIX + METHOD_NAME, "END", Log.PROCEDURE);
}
}

```

## Other Invocation Usage Considerations

While implementing the service invocation framework to invoke Web services, some limitations need to be considered.

- WFBES\_INPUT\_<partname> parameter can only be passed at run time during the event raise.
- The service invocation framework supports invoking only document-based Web services.
- Support One-to-One relationship of event subscriptions

To successfully invoke Web services, each event should only have one subscription (with 'Invoker Web Service' action type) associated with it. This one-to-one relationship of event subscription is especially important in regards to synchronous request - response service invocation.

For detailed information about implementation consideration on service invocation framework, see Implementation Limitation and Consideration, *Oracle E-Business Suite Integrated SOA Gateway Implementation Guide*.

---

# Integration Repository Annotation Standards

## General Guidelines

The Oracle Integration Repository is a centralized repository that contains numerous integration interface endpoints exposed by applications throughout the entire Oracle E-Business Suite. The Integration Repository is populated by the parsing of annotated source code files. Source code files are the "source of truth" for Integration Repository metadata, and it is vitally important that they are annotated in a prescribed and standardized fashion.

This section describes what you should know in general about Integration Repository annotations, regardless of the source code file type that you are working with.

### Annotation Syntax

Annotations are modifiers that contain an annotation type and zero or more member-value pairs. Each member-value pair associates a value with a different member of the annotation type.

The annotation syntax is similar to Javadoc syntax:

```
@Namespace:TypeName keyString
```

```
@Namespace:TypeName freeString
```

```
@Namespace:TypeName keyString keyString keyString
```

```
@Namespace:TypeName keyString freeString
```

```
@Namespace:TypeName {inline annotation} {inline annotation}
```

### Element Definitions

`Namespace` identifies the group of annotations that you are using. It is case sensitive. The annotations currently in use are in the `rep` namespace. Future annotations may be introduced in different namespaces.

`TypeName` identifies the name of the annotation type. It is case sensitive. For

consistency across product teams, always use lowercase typenames.

`keyString` is the first word that follows the annotation. It is a whole string that excludes spaces.

`freeString` is a string that follows the `keyString`. It may have spaces or inline annotations. It is terminated at the beginning of the next annotation or at the end of the documentation comment.

### Format Requirement

In your source code file, repository annotations will appear as a Javadoc-style block of comments.

Use the following general procedure. (If you are working in Java and your file already has robust Javadoc comments, then in many cases you'll only need to add the appropriate `@rep:` tags.)

- Choose which interfaces you will expose to the Integration Repository. Be mindful that you can annotate interfaces as *public*, *private*, or *internal*, as well as *active*, *obsolete*, *deprecated*, or *planned*.

Only interfaces that you annotate as public will appear in the external Integration Repository UI; private and internal interfaces will appear in an internal-only Oracle UI. Consequently, all interfaces that have previously been documented as public in customer manuals should be defined as public in your source file annotations.

- In your source file, set off the beginning of the annotation block according to the following conditional rule:
  - For Java, insert "slash-star-star" characters (`/**`).
  - For non-Java files, insert "slash-star-pound" characters (`/*#`).
- Enter a text description. Use complete sentences and standard English.
- Where applicable, add plain Javadoc tags such as `@param` and `@return`.
- Next, add `@rep:` tags such as `@rep:scope` and `@rep:product`.
- Optionally, add a nonpublishable comment using the `@rep:comment` annotation. (Use for reminders, notes, and so on. The parsers skip this annotation.)
- End the annotation block with a "star-slash" (`*/`).

Refer to the following example. Note that the first line could alternatively be slash-star-pound (`/*#`) if the source file was PL/SQL or another non-Java technology.

```

/**
 * This is the first sentence of a description of a sample
 * interface. This description can span multiple lines.
 * Be careful for public interfaces, where the description is
 * displayed externally in the Integration Repository UI.
 * It should be reviewed for content as well as spelling and
 * grammar errors. Additionally, the first sentence of
 * the description should be a concise summary of the
 * interface or method, as the repository UI will display
 * the first sentence by itself.
 *
 * @param <param name> <parameter description>
 * @rep:paraminfo {@rep:innertype <typeName>} {@rep:precision <value>}
 {@rep:required}
 * @rep:scope <public | internal | private>
 * @rep:product <product short code>
 * @rep:displayname Sample Interface
 */

```

### Annotation Syntax Checker and iLDT Generator

A syntax checker is available at the following directory:

```
$IAS_ORACLE_HOME/perl/bin/perl $FND_TOP/bin/irep_parser.pl
```

Details about the checker can be found by using the -h flag.

### Class Level vs. Method Level

For the purpose of classifying annotation requirements, we are using loose definitions of the terms "class" and "method". In the context of interface annotations, PL/SQL packages are thought of as classes, and PL/SQL functions or procedures are thought of as methods. For some technologies there are different annotation requirements at the class level and the method level. See the "Required" and "Optional" annotation lists below for details.

### Concurrent Program Considerations

In cases where a Concurrent Program (CP) is implemented with an underlying technology that is also an interface type (such as a PL/SQL or Java CP) there may be some confusion as to what needs to be annotated.

Assuming that you intend to have the Concurrent Program exposed by the repository, you should annotate the Concurrent Program. Do not annotate the underlying implementation (such as PL/SQL file) unless you intend to expose it separately from the concurrent program in the repository.

The annotation standards for the following integration interfaces are discussed in this chapter:

- Java Annotations, page A-4
- PL/SQL Annotations, page A-11
- Concurrent Program Annotations, page A-17
- XML Gateway Annotations, page A-19

- Business Event Annotations, page A-31
- Business Entity Annotations, page A-37
- Composite Service - BPEL Annotations, page A-105
- Glossary of Annotations, page A-112

## Java Annotations

Users will place their annotations in Javadoc comments, immediately before the declaration of the class or method.

### Required Class-level Annotations

- must begin with description sentence(s), page A-112
- rep:scope, page A-115
- rep:product, page A-116
- rep:implementation, page A-116 (only required for Java business service objects; not required for plain Java or SDOs)
- rep:displayname, page A-117
- rep:service, page A-135
- rep:servicedoc, page A-136

### Optional Class-level Annotations

- link, page A-120
  - see, page A-121
  - rep:lifecycle, page A-119
  - rep:category, page A-125
- Use BUSINESS\_ENTITY at the class level only if all underlying methods have the same business entity. In those cases, you do not need to repeat the annotation at the method level.
- rep:compatibility, page A-119
  - rep:standard, page A-128

- `rep:ihelp`, page A-122
- `rep:metalink`, page A-123
- `rep:doccd`, page A-124
- `rep:synchronicity`, page A-137

#### **Required Method-level Annotations**

- must begin with description sentence(s), page A-112
- `param`, page A-130  
Use only when applicable and when other tags such as `@see` and `@rep:metalink` do not provide parameter explanations.
- `return`, page A-131 (if applicable)
- `rep:paraminfo`, page A-131
- `rep:displayname`, page A-117
- `rep:businessevent`, page A-134 (if an event is raised)

#### **Optional Method-level Annotations**

- `link`, page A-120
- `see`, page A-121
- `rep:scope`, page A-115
- `rep:lifecycle`, page A-119
- `rep:compatibility`, page A-119
- `rep:category`, page A-125  
Use `BUSINESS_ENTITY` at the method level only when a class methods have heterogeneous business entities.
- `rep:ihelp`, page A-122
- `rep:metalink`, page A-123
- `rep:doccd`, page A-124
- `rep:appscontext`, page A-138

- rep:synchronicity, page A-137
- rep:primaryinstance, page A-139

## Template

You can use the following template when annotating Business Service Objects:

Interface Template:

```
/**
 * < Interface description
 *   ...
 * >
 *
 * @rep:scope <public|private|internal>
 * @rep:displayname <Interface display name>
 * @rep:lifecycle <active|deprecated|obsolete|planned>
 * @rep:product <product code>
 * @rep:compatibility <S|N>
 * @rep:implementation <full implementation class name>
 * @rep:category <lookupType> <lookupCode> <sequenceNumber>
 */
```

Methods Template:

```
/**
 * < Method description
 *   ...
 * >
 *
 * @param <paramName> < Parameter description
 *   ... >
 * @rep:paraminfo {@rep:innertype <typeName>} {@rep:precision <value>}
 {@rep:required}
 *
 *
 * @return < Parameter description
 *   ... >
 * @rep:paraminfo {@rep:innertype <typeName>} {@rep:precision <value>}
 {@rep:required}
 *
 *
 * @rep:scope <public|private|internal>
 * @rep:displayname <Interface display name>
 * @rep:lifecycle <active|deprecated|obsolete|planned>
 * @rep:compatibility <S|N>
 * @rep:category <lookupType> <lookupCode> <sequenceNumber>
 * @rep:businessevent <businessEventName>
 */
```

## Examples

For reference, here is an example of an annotated Purchase Order service:



```

...
package oracle.apps.po.tutorial;

import oracle.jbo.domain.Number;

import oracle.svc.data.DataList;
import oracle.svc.data.DataService;
import oracle.svc.msg.MessageService;

import oracle.apps.fnd.common.VersionInfo;

/**
 * The Purchase Order service lets you to view, update, acknowledge and
 * approve purchase orders. It also lets you receive items, and obtain
 * pricing by line item.
 *
 * @see oracle.apps.fnd.framework.toolbox.tutorial.PurchaseOrderSDO
 * @see
oracle.apps.fnd.framework.toolbox.tutorial.PurchaseOrderAcknowledgements
SDO
 * @see
oracle.apps.fnd.framework.toolbox.tutorial.PurchaseOrderReceiptsSDO
 *
 * @rep:scope public
 * @rep:displayname Purchase Order Service
 * @rep:implementation
oracle.apps.fnd.framework.toolbox.tutorial.server.PurchaseOrderSAMImpl
 * @rep:product PO
 * @rep:category BUSINESS_ENTITY PO_PURCHASE_ORDER
 * @rep:service
 */
public interface PurchaseOrder extends DataService, MessageService
{
    public static final String RCS_ID="$Header$";
    public static final boolean RCS_ID_RECORDED =
        VersionInfo.recordClassVersion(RCS_ID,
"oracle.apps.fnd.framework.toolbox.tutorial");

/**
 * Approves a purchase order.
 *
 * @param purchaseOrder purchase order unique identifier
 * @rep:paraminfo {@rep:required}
 *
 * @rep:scope public
 * @rep:displayname Approve Purchase Orders
 * @rep:businessevent oracle.apps.po.approve
 */
    public void approvePurchaseOrder(Number poNumber);

/**
 * Acknowledges purchase orders, including whether the terms have
 * been accepted or not. You can also provide updated line
 * item pricing and shipment promise dates with the acknowledgement.
 *
 * @param purchaseOrders list of purchase order objects
 * @rep:paraminfo {@rep:innertype
oracle.apps.fnd.framework.toolbox.tutorial.PurchaseOrderAcknowledgements
SDO} {@rep:required}
 *
 * @rep:scope public

```

```

* @rep:displayname Receive Purchase Order Items
* @rep:businessevent oracle.apps.po.acknowledge
*/
public void acknowledgePurchaseOrders(DataList purchaseOrders);

/**
 * Receives purchase order items. For each given purchase order
 * shipment, indicate the quantity to be received and, optionally,
 * the receipt date if today's date is not an acceptable receipt date.
 *
 * @param purchaseOrders list of purchase order objects
 * @rep:paraminfo {@rep:innertype
oracle.apps.fnd.framework.toolbox.tutorial.PurchaseOrderReceiptsSDO}
{@required}
 *
 * @rep:scope public
 * @rep:displayname Receive Purchase Order Items
 * @rep:businessevent oracle.apps.po.receive_item
 */
public void receiveItems(DataList purchaseOrders);

/**
 * Gets the price for a purchase order line item.
 *
 * @param poNumber purchase order unique identifier
 * @rep:paraminfo {@required}
 * @param lineNumber purchase order line unique identifier
 * @rep:paraminfo {@required}
 * @return the item price for the given purchase order line
 *
 * @rep:scope public
 * @rep:displayname Get Purchase Order Line Item Price
 */
public Number getItemPrice(Number poNumber,
                           Number lineNumber);

```

Here is an example of an annotated Purchase Order SDO data object:

```

/*=====
====+
|      Copyright (c) 2004 Oracle Corporation, Redwood Shores, CA, USA
|
|      All rights reserved.
|

+=====
====+
|  HISTORY
|

+=====
====*/
package oracle.apps.po.tutorial;

import oracle.jbo.domain.Number;

import oracle.svc.data.DataObjectImpl;
import oracle.svc.data.DataList;

/**
 * The Purchase Order Data Object holds the purchase order data
 * including
 * nested data objects such as lines and shipments.
 *
 * @see oracle.apps.fnd.framework.toolbox.tutorial.PurchaseOrderLineSDO
 *
 * @rep:scope public
 * @rep:displayname Purchase Order Data Object
 * @rep:product PO
 * @rep:category BUSINESS_ENTITY PO_PURCHASE_ORDER
 * @rep:servicedoc
 */
public class PurchaseOrderSDO extends DataObjectImpl
{
    public PurchaseOrderSDO ()
    {
        super();
    }

    /**
     * Returns the purchase order header id.
     *
     * @return purchase order header id.
     */
    public Number getHeaderId()
    {
        return (Number)getAttribute("HeaderId");
    }

    /**
     * Sets the purchase order header id.
     *
     * @param value purchase order header id.
     * @rep:paraminfo {@rep:precision 5} {@rep:required}
     */
    public void setHeaderId(Number value)
    {
        setAttribute("HeaderId", value);
    }
}

```

```

/**
 * Returns the purchase order name.
 *
 * @return purchase order name.
 * @rep:paraminfo {rep:precision 80}
 */
public String getName()
{
    return (String)getAttribute("Name");
}

/**
 * Sets the purchase order header name.
 *
 * @param value purchase order header name.
 * @rep:paraminfo {@rep:precision 80}
 */
public void setName(String value)
{
    setAttribute("Name", value);
}

/**
 * Returns the purchase order description.
 *
 * @return purchase order description.
 * @rep:paraminfo {rep:precision 120}
 */
public String getDescription()
{
    return (String)getAttribute("Description");
}

/**
 * Sets the purchase order header description.
 *
 * @param value purchase order header description.
 * @rep:paraminfo {@rep:precision 80}
 */
public void setDescription(String value)
{
    setAttribute("Description", value);
}

/**
 * @return the purchase order lines DataList.
 * @rep:paraminfo {@rep:innertype
oracle.apps.fnd.framework.toolbox.tutorial.PurchaseOrderLineSDO}
 */
public DataList getLines()
{
    return (DataList)getAttribute("Lines");
}

/**
 * @param list the putrchase order lines DataList.
 * @rep:paraminfo {@rep:innertype
oracle.apps.fnd.framework.toolbox.tutorial.PurchaseOrderLineSDO}
 */
public void setLines(DataList list)
{

```

```

setAttribute("Lines", list);
    }

}

```

## PL/SQL Annotations

You can annotate \*.pls and \*.pkh files.

For PL/SQL packages, only the package spec should be annotated. Do not annotate the body.

Before annotating, make sure that no comments beginning with /\*# are present. The "slash-star-pound" characters are used to set off repository annotations, and will result in either an error or undesirable behavior if used with normal comments.

To annotate, use a text editor (such as emacs or vi.) to edit the file. For each package, begin your annotations at the second line immediately after the CREATE OR REPLACE PACKAGE <package\_name> AS line. (The first line after CREATE OR REPLACE PACKAGE <package\_name> AS should be the /\* \$Header: \$ \*/ line.)

### Required Class-level Annotations

- must begin with description sentence(s), page A-112
- rep:scope, page A-115
- rep:product, page A-116
- rep:displayname, page A-117
- rep:category, page A-125

Use BUSINESS\_ENTITY at the class level only if all underlying methods have the same business entity. In those cases, you do not need to repeat the annotation at the method level.

- rep:businessevent, page A-134 (if an event is raised)

### Optional Class-level Annotations

- link, page A-120
- see, page A-121
- rep:lifecycle, page A-119
- rep:compatibility, page A-119
- rep:ihelp, page A-122

- rep:metalink, page A-123
- rep:doccd, page A-124

#### **Required Method-level Annotations**

- must begin with description sentence(s), page A-112
- param, page A-130  
Use only when applicable and when other tags such as @see and @rep:metalink do not provide parameter explanations.
- return, page A-131 (if applicable)
- rep:displayname, page A-117
- rep:paraminfo, page A-131
- rep:businessevent, page A-134 (if an event is raised)

#### **Optional Method-level Annotations**

- link, page A-120
- see, page A-121
- rep:scope, page A-115
- rep:lifecycle, page A-119
- rep:compatibility, page A-119
- rep:category, page A-125  
Use BUSINESS\_ENTITY at the method level only when a class methods have heterogeneous business entities.
- rep:ihelp, page A-122
- rep:metalink, page A-123
- rep:doccd, page A-124
- rep:appscontext, page A-138
- rep:primaryinstance, page A-139

## Template

You can use the following template when annotating PL/SQL files:

```

.
.
.
CREATE OR REPLACE PACKAGE <package name> AS
/* $Header: $ */
/*#
  * <Put your long package description here
  * it can span multiple lines>
  * @rep:scope <scope>
  * @rep:product <product or pseudoproduct short code>
  * @rep:lifecycle <lifecycle>
  * @rep:displayname <display name>
  * @rep:compatibility <compatibility code>
  * @rep:businessevent <Business event name>
  * @rep:category BUSINESS_ENTITY <entity name>
  */

.
.
.

/**
  * <Put your long procedure description here
  * it can span multiple lines>
  * @param <param name 1> <param description 1>
  * @param <param name 2> <param description 2>
  * @rep:scope <scope>
  * @rep:product <product or pseudoproduct short code>
  * @rep:lifecycle <lifecycle>
  * @rep:displayname <display name>
  * @rep:compatibility <compatibility code>
  * @rep:businessevent <Business event name>
  */
PROCEDURE <procedure name> ( . . . );

.
.
.

/**
  * <Put your long function description here
  * it can span multiple lines>
  * @param <param name 1> <param description 1>
  * @param <param name 2> <param description 2>
  * @return <return description>
  * @rep:scope <scope>
  * @rep:product <product or pseudoproduct short code>
  * @rep:lifecycle <lifecycle>
  * @rep:displayname <display name>
  * @rep:compatibility <compatibility code>
  * @rep:businessevent <Business event name>
  */
FUNCTION <function name> ( . . . );

.
.
.

END <package name>;

```



```
/
commit;
exit;
```

### **Example**

For reference, here is an example of an annotated PL/SQL file:

```

set verify off
whenever sqlerror exit failure rollback;
whenever oserror exit failure rollback;

create or replace package WF_ENGINE as

/*#
 * This is the public interface for the Workflow engine. It allows
 * execution of various WF engine functions.
 * @rep:scope public
 * @rep:product WF
 * @rep:displayname Workflow Engine
 * @rep:lifecycle active
 * @rep:compatibility S
 * @rep:category BUSINESS_ENTITY WF_WORKFLOW_ENGINE
 */

g_nid number;          -- current notification id
g_text varchar2(2000); -- text information

--
-- AddItemAttr (PUBLIC)
-- Add a new unvalidated run-time item attribute.
-- IN:
-- itemtype - item type
-- itemkey - item key
-- aname - attribute name
-- text_value - add text value to it if provided.
-- number_value - add number value to it if provided.
-- date_value - add date value to it if provided.
-- NOTE:
-- The new attribute has no type associated. Get/set usages of the
-- attribute must insure type consistency.
--
/*#
 * Adds Item Attribute
 * @param itemtype item type
 * @param itemkey item key
 * @param aname attribute name
 * @param text_value add text value to it if provided.
 * @param number_value add number value to it if provided.
 * @param date_value add date value to it if provided.
 * @rep:scope public
 * @rep:lifecycle active
 * @rep:displayname Add Item Attribute
 */
procedure AddItemAttr(itemtype in varchar2,
                     itemkey in varchar2,
                     aname in varchar2,
                     text_value in varchar2 default null,
                     number_value in number default null,
                     date_value in date default null);

--
-- AddItemAttrTextArray (PUBLIC)
-- Add an array of new unvalidated run-time item attributes of type
text.

```

```

-- IN:
--   itemtype - item type
--   itemkey - item key
--   aname - Array of Names
--   avalue - Array of New values for attribute
-- NOTE:
--   The new attributes have no type associated.  Get/set usages of
these
--   attributes must insure type consistency.
--

END WF_ENGINE;
/

commit;
exit;

```

## Concurrent Program Annotations

To annotate a concurrent program, select the System Administration responsibility and click on OA Framework based Define Concurrent Program page. Query the Concurrent Program and go to the Annotations field. Enter your annotations there and commit to save your work.

After annotating and committing, you will need to use FNDLOAD to recreate the LDTs for your concurrent programs.

### Required Class-level Annotations

- must begin with description sentence(s), page A-112  
The annotation takes precedence over the concurrent program own definition in the LDT. One or the other must exist; otherwise, interface generation will fail.
- rep:scope, page A-115
- rep:product, page A-116
- rep:displayname, page A-117  
The annotation takes precedence over the concurrent program own definition in the LDT. One or the other must exist; otherwise, interface generation will fail.
- rep:category, page A-125
- rep:businessevent, page A-134 (if an event is raised)

**Note:** There is no required method-level annotations for concurrent programs.

### Optional Class-level Annotations

- link, page A-120
- see, page A-121
- rep:lifecycle, page A-119
- rep:compatibility, page A-119
- rep:ihelp, page A-122
- rep:metalink, page A-123
- rep:doccd, page A-124
- rep:usestable, page A-127
- rep:usesmap, page A-140

**Note:** There is no optional method-level annotations for concurrent programs.

### Template

You can use the following template when annotating Concurrent Programs:

```
/*#
 * <Put your long description here
 * it can span multiple lines>
 * @rep:scope <scope>
 * @rep:product <product or pseudoproduct short code>
 * @rep:lifecycle <lifecycle>
 * @rep:category OPEN_INTERFACE <open interface name> <sequence_num>
 * @rep:usestable <table or view name> <sequence_num> <direction>
 * @rep:category BUSINESS_ENTITY <BO type>
 * @rep:category <other category> <other value>
 * @rep:businessevent <name of business event>
 */
```

### Example

For reference, here is an example of an annotated Concurrent Program:

```

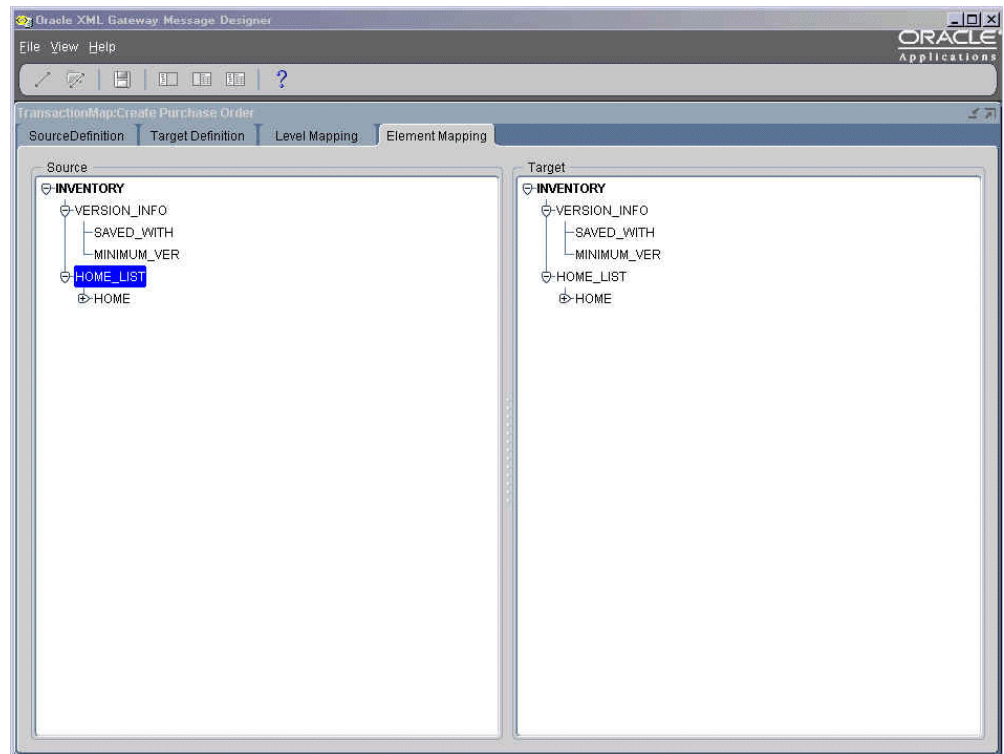
/**
 * Executes the Open Interface for Accounts Payable Invoices. It uses
the
 * following tables: AP_INVOICES_INTERFACE, AP_INVOICE_LINES_INTERFACE.
 * @rep:scope public
 * @rep:product AP
 * @rep:lifecycle active
 * @rep:category OPEN_INTERFACES AP_INVOICES_INTERFACE 1
 * @rep:usestable AP_INVOICES_INTERFACE 2 IN
 * @rep:usestable AP_INVOICE_LINES_INTERFACE 3 IN
 * @rep:category BUSINESS_ENTITY AP_INVOICE
 */

```

## XML Gateway Annotations

Use the following procedure to annotate an XML Gateway map for transaction information:

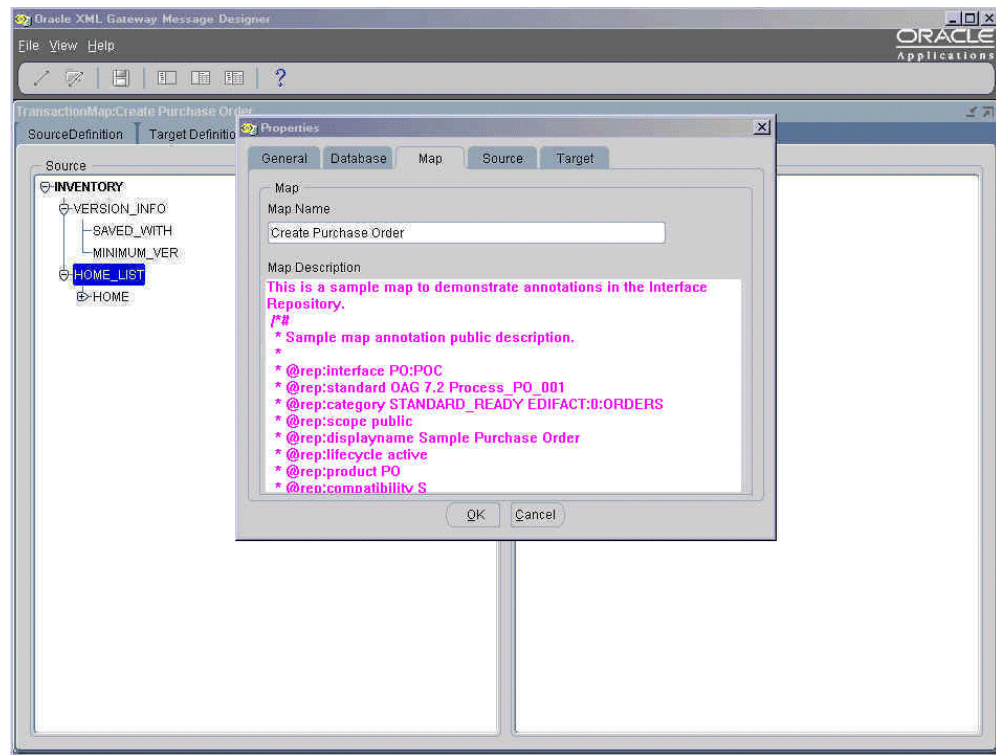
1. Check out an existing map from source code and open it in Message Designer.



2. Find out which Internal Transaction Type, Subtype, Standard, and Direction this particular map is associated with. Note that this entry must exist in XML Gateway to be loaded into the Integration Repository.

Click **Message Designer File.... > Properties** and select the Map tab. Annotate the map using the Map Description field after your existing description. Be sure to enter the @rep:interface annotation with <Internal Transaction

Type>:<Subtype>, @rep:standard, and @rep:direction accordingly.



(Optional) If this map is designed to fully support a given standard such as OAG, then set @rep:standard to the standard, version and spec name. However, if the map is designed with the intention of supporting standards through additional custom transformations (such as, it is "ready" for the standard), then use the rep:category\_STANDARD\_READY, page A-125 annotation to denote this.

- A given Internal Transaction Type and Subtype should have only one map seeded by product teams for a given Standard and Direction (regardless of Party Type). Additional maps containing the same types in the annotations would be rejected and treated as errors. Note that there may exist different maps based on the External Transaction Type and Subtype, but as these are meant to be Trading Partner-specific, we do not enter them in the repository. In future releases, we will enforce these rules natively within XML Gateway.
- If a single map is reused in more than one Internal Transaction Type and Subtype, then you may enter multiple annotations, each within its own comment block (i.e. between /\*# ... \*/). The parser will create entries in the Integration Repository for each annotation set. Although this capability is supported, you are

encouraged to use two different maps to accommodate potentially changing interfaces. See the following example of map reuse:

Int T	Int ST	D	Ext T	Ext ST	STD	Party Type
AR	Invoice	O	Invoice	Process	OAG	C
AR	Credit	O	Invoice	Process	OAG	C
AR	Debit	O	Invoice	Process	OAG	C

In this scenario, since the external representation does not change, the same map can be reused. However, the internal processing and authorization considerations may differ based on the Internal Transaction Type and Subtype. In this case, the map can have three annotation blocks, one for each Internal Transaction Type and Subtype; such as. AR-Invoice, AR-Credit, and AR-Debit.

- Parameters are typically used in outbound maps for specifying keys used in queries to produce outbound data. Inbound maps do not have parameters.
3. Save the annotated map, check it into source control, and release as a patch as usual. The annotations are updated as part of the Integration Repository loaders.

#### Required Class-level Annotations

- must begin with description sentence(s), page A-112
- rep:scope, page A-115
- rep:product, page A-116
- rep:displayname, page A-117
- rep:category, page A-125
- rep:standard, page A-128
- rep:interface, page A-129

- rep:businessevent, page A-134 (if an event is raised)
- rep:direction, page A-135

**Note:** There is no required method-level annotations for XML Gateway.

#### Optional Class-level Annotations

- link, page A-120
- see, page A-121
- param, page A-130  
Use only when applicable and when other tags such as @see and @rep:metalink do not provide parameter explanations.
- rep:paraminfo, page A-131
- rep:lifecycle, page A-119
- rep:compatibility, page A-119
- rep:ihelp, page A-122
- rep:metalink, page A-123
- rep:doccd, page A-124
- rep:synchronicity, page A-137

**Note:** There is no optional method-level annotations for XML Gateway.

#### Template

You can use the following template when annotating XML Gateway:



#### Sample Inbound Map Annotation

```
/*#
 * Sample map annotation public description.
 *
 * @rep:interface <transaction_type:sub_type>
 * @rep:standard <OAG|cXML> <7.2|7.3> <specname>
 * @rep:direction IN
 * @rep:scope <public|private|internal>
 * @rep:displayname <Interface display name>
 * @rep:lifecycle <active|deprecated|obsolete|planned>
 * @rep:product <product code>
 * @rep:compatibility <S|N>
 * @rep:category <lookupType> <lookupCode> <sequenceNumber>
 * @rep:category STANDARD_READY <standard:version:specification>
 * @rep:businessevent <businessEventName>
 */
```

#### Sample Outbound Map Annotation

```
/*#
 * Sample map annotation public description.
 *
 * @param <paramName> <Parameter description>
 * @rep:paraminfo {@rep:required}
 *
 * @rep:interface <transaction_type:sub_type>
 * @rep:standard <OAG|cXML> <7.2|7.3> <specname>
 * @rep:direction OUT
 * @rep:scope <public|private|internal>
 * @rep:displayname <Interface display name>
 * @rep:lifecycle <active|deprecated|obsolete|planned>
 * @rep:product <product code>
 * @rep:compatibility <S|N>
 * @rep:category <lookupType> <lookupCode> <sequenceNumber>
 * @rep:category STANDARD_READY <standard:version:specification>
 * @rep:businessevent <businessEventName>
 */
```

#### Important Note

A given map should be unique to a given Internal Transaction Type / Subtype, Standard and Direction. This is because the External Transaction Type / Subtype are meant for Trading Partner specific values to be specified in the Trading Partner Details form and the entries in the Integration Repository are NOT Trading Partner specific. Moreover, there should not be a need to change maps on a per Trading Partner basis, and if it does, then those maps should not be part of the Integration Repository entries.

Given the current data model however, it is possible that a given map could differ by External Transaction Type / Subtype and even by Trading Partner. Going forward, this would not be allowed for seeded maps and the Integration Repository parser would return an error if it finds multiple maps which point to the same Internal Transaction Type / Subtype.

#### Additional Notes

\* Parameters are typically used in outbound maps for specifying keys used in queries to produce outbound data

**Example**

For reference, here is an example of an annotated XML Gateway interface:

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- $Header: MapPrinter.java 115.12 2009/06/13 21:17:58 mtai noship $
-->
<!-- WARNING: This file should only be edited using Message Designer -->
<?xGateway mapType="MAP" ?>
<?xGatewayVersion designerVersion="2.6.3.0.0" ?>
<ECX_MAPPINGS>
<MAP_CODE>Create Purchase Order</MAP_CODE>
<DESCRIPTION>This is a sample map to demonstrate annotations in the
Interface Repository.
/*#
 * Sample map annotation public description.
 *
 * @rep:interface PO:POC
 * @rep:standard OAG 7.2 Process_PO_001
 * @rep:direction IN
 * @rep:scope public
 * @rep:displayname Create Purchase Order
 * @rep:lifecycle active
 * @rep:product PO
 * @rep:compatibility S
 * @rep:category BUSINESS_OBJECT PURCHASE_ORDER
 * @rep:businessevent oracle.apps.po.received
 */
/*#
 * Sample map annotation public description for reused transaction
 *
 * @rep:interface PO:POU
 * @rep:standard OAG 7.2 Process_PO_001
 * @rep:direction IN
 * @rep:scope public
 * @rep:displayname Update Purchase Order
 * @rep:lifecycle active
 * @rep:product PO
 * @rep:compatibility S
 * @rep:category BUSINESS_OBJECT PURCHASE_ORDER
 * @rep:businessevent oracle.apps.po.received
 */
</DESCRIPTION>
<OBJECT_ID_SOURCE>1</OBJECT_ID_SOURCE>
<OBJECT_ID_TARGET>2</OBJECT_ID_TARGET>
<ENABLED>Y</ENABLED>
<ECX_MAJOR_VERSION>2</ECX_MAJOR_VERSION>
<ECX_MINOR_VERSION>6</ECX_MINOR_VERSION>
<ECX_OBJECTS>
<OBJECT_ID>1</OBJECT_ID>
<OBJECT_NAME>SRC</OBJECT_NAME>
<OBJECT_TYPE>XML</OBJECT_TYPE>
<OBJECT_DESCRIPTION>Source Definition</OBJECT_DESCRIPTION>
<OBJECT_STANDARD>OAG</OBJECT_STANDARD>
<ROOT_ELEMENT>INVENTORY</ROOT_ELEMENT>

<ECX_OBJECT_LEVELS>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<OBJECT_ID>1</OBJECT_ID>
<OBJECT_LEVEL>0</OBJECT_LEVEL>
<OBJECT_LEVEL_NAME>INVENTORY</OBJECT_LEVEL_NAME>
<PARENT_LEVEL>0</PARENT_LEVEL>
<ENABLED>Y</ENABLED>
<ECX_OBJECT_ATTRIBUTES>

```

```

<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>0</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>INVENTORY</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>

<PARENT_ATTRIBUTE_ID>0</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>Y</REQUIRED_FLAG>
<IS_MAPPED>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>1</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>VERSION_INFO</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>0</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>2</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>SAVED_WITH</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>1</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>3</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>MINIMUM_VER</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>1</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>4</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>HOME_LIST</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>

```

```

<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>0</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>5</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>HOME</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>4</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>4</HAS_ATTRIBUTES>
<LEAF_NODE>0</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>6</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>NAME</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>7</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>LOC</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>8</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>TYPE</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>

```

```

<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>0</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>9</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>IDX</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
</ECX_OBJECT_LEVELS>
</ECX_OBJECTS>
<ECX_OBJECTS>
<OBJECT_ID>2</OBJECT_ID>
<OBJECT_NAME>TGT</OBJECT_NAME>
<OBJECT_TYPE>XML</OBJECT_TYPE>
<OBJECT_DESCRIPTION>Target Definition</OBJECT_DESCRIPTION>
<OBJECT_STANDARD>OAG</OBJECT_STANDARD>
<ROOT_ELEMENT>INVENTORY</ROOT_ELEMENT>

<ECX_OBJECT_LEVELS>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<OBJECT_ID>2</OBJECT_ID>
<OBJECT_LEVEL>0</OBJECT_LEVEL>
<OBJECT_LEVEL_NAME>INVENTORY</OBJECT_LEVEL_NAME>
<PARENT_LEVEL>0</PARENT_LEVEL>
<ENABLED>Y</ENABLED>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>0</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>INVENTORY</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>

<PARENT_ATTRIBUTE_ID>0</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>Y</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>1</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>VERSION_INFO</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>0</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>

```

```

<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>2</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>SAVED_WITH</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>1</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>3</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>MINIMUM_VER</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>1</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>4</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>HOME_LIST</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>0</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>5</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>HOME</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>4</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>1</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>4</HAS_ATTRIBUTES>
<LEAF_NODE>0</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>

```

```

<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>6</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>NAME</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>7</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>LOC</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>8</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>TYPE</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
<ECX_OBJECT_ATTRIBUTES>
<OBJECTLEVEL_ID>1</OBJECTLEVEL_ID>
<ATTRIBUTE_ID>9</ATTRIBUTE_ID>
<ATTRIBUTE_NAME>IDX</ATTRIBUTE_NAME>
<OBJECT_COLUMN_FLAG>N</OBJECT_COLUMN_FLAG>
<DATA_TYPE>VARCHAR2</DATA_TYPE>
<PARENT_ATTRIBUTE_ID>5</PARENT_ATTRIBUTE_ID>

<ATTRIBUTE_TYPE>2</ATTRIBUTE_TYPE>
<HAS_ATTRIBUTES>0</HAS_ATTRIBUTES>
<LEAF_NODE>1</LEAF_NODE>
<REQUIRED_FLAG>N</REQUIRED_FLAG>
<IS_MAPPED>>false</IS_MAPPED>
</ECX_OBJECT_ATTRIBUTES>
</ECX_OBJECT_LEVELS>
</ECX_OBJECTS>
</ECX_MAPPINGS>
<SCRIPT SRC="/oracle_smp_chronos/oracle_smp_chronos.js"></SCRIPT>

```



## Business Event Annotations

This section describes what you should know about Integration Repository annotations for business events, and includes the following topics:

- Annotating Business Events
- Annotations for Business Events - Syntax
- Required Annotations
- Optional Annotations
- Template
- Example

### Annotating Business Events

- You should annotate business events in \*.wfx files.
- You should annotate only events. Subscriptions need not be annotated; they will not be available in Integration Repository.
- Before annotating, make sure that no comments beginning with /\*# are present. These "slash-star-pound" characters are used to mark the start of repository annotations, and will produce errors or unspecified behavior if used in normal comments.
- To annotate, use a text editor such as emacs or vi to edit the file.
- In the .wfx file, place the annotations within the <IREP\_ANNOTATION> tag for the business event. Note that the <IREP\_ANNOTATION> tag is a child node of the <WF\_EVENTS> tag.
- For .wfx files having multiple business event definitions, each of the business event definitions should be separately annotated. That is, you should place the annotation within an <IREP\_ANNOTATION> tag for the appropriate business events.
- Enter a meaningful description that covers the condition under which the business event is raised, and the UI action that invokes the business event.
- Define product codes in FND\_APPLICATION.
- Use existing business entities for your events. For the list of existing business entities, see Business Entity Annotation Guidelines, page A-37.

- If you decide not to annotate or publish the event after all, you should remove the annotation only, and not the associated tags.

The presence of either the `<IREP_ANNOTATION/>` tag or `<IREP_ANNOTATION></IREP_ANNOTATION>` tag is an indication to the loader that the business event has been reviewed for annotation and does not need to be published to integration repository. The next time the user downloads these events, the loader will insert empty `<IREP_ANNOTATION>` tags.

- If you remove the entire `<IREP_ANNOTATION>` tag for the business event and then upload it, on a subsequent download the loader will insert a partially filled annotation template for the business event.

### Annotations for Business Events - Syntax

The annotations for business events are:

```
<IREP_ANNOTATION>
/*#
* This event is raised after the Purchase Order has been pushed
* to Oracle Order management open interface tables. This event
* will start the workflow OEOI/R_OEOI_ORDER_IMPORT to import the
* order.
* @rep:scope public
* @rep:displayname OM Generic Inbound Event
* @rep:product ONT
* @rep:category BUSINESS_ENTITY ONT_SALES_ORDER
*/
</IREP_ANNOTATION>
```

Refer to General Guidelines for Annotations, page A-1 for details of element definitions.

### Required Annotations

Follow the links below to view syntax and usage of each annotation.

- Must begin with description sentence(s)
- `rep:displayname`, page A-117
- `rep:scope`, page A-115
- `rep:product`, page A-116
- `rep:category BUSINESS_ENTITY`, page A-125

### Optional Annotations

- `link`, page A-120
- `see`, page A-121
- `rep:lifecycle`, page A-119

- rep:compatibility, page A-119
- rep:ihelp, page A-122
- rep:metalink, page A-123
- rep:doccd, page A-124

### **Template**

You can use this template when annotating .wfx files.

```

.
.
.

<oracle.apps.wf.event.all.sync>
.
.
.
<WF_TABLE_DATA>
  <WF_EVENTS>
    <VERSION>...</VERSION>
    <GUID>....</GUID>
    <NAME>event name</NAME>
    <TYPE>EVENT</TYPE>
    <STATUS>ENABLED</STATUS>
    <GENERATE_FUNCTION/>
    <OWNER_NAME> ... </OWNER_NAME>
    <OWNER_TAG>...</OWNER_TAG>
    <CUSTOMIZATION_LEVEL>...</CUSTOMIZATION_LEVEL>
    <LICENSED_FLAG>..</LICENSED_FLAG>
    <DISPLAY_NAME>...</DISPLAY_NAME>
    <DESCRIPTION> Description for business event </DESCRIPTION>
    <IREP_ANNOTATION>

/*#
 * Put your long package description here; it can span multiple lines.
 *
 * @rep:scope <scope>
 * @rep:displayname <display name>
 * @rep:product <product or pseudoproduct short code>
 * @rep:category BUSINESS_ENTITY <entity name>
 */
    </IREP_ANNOTATION>
  </WF_EVENTS>
</WF_TABLE_DATA>

.
.
.

<WF_TABLE_DATA>
  <WF_EVENTS>
    <VERSION>...</VERSION>
    <GUID>....</GUID>
    <NAME>event name</NAME>
    <TYPE>EVENT</TYPE>
    <STATUS>ENABLED</STATUS>
    <GENERATE_FUNCTION/>
    <OWNER_NAME> ... </OWNER_NAME>
    <OWNER_TAG>...</OWNER_TAG>
    <CUSTOMIZATION_LEVEL>...</CUSTOMIZATION_LEVEL>
    <LICENSED_FLAG>..</LICENSED_FLAG>
    <DISPLAY_NAME>...</DISPLAY_NAME>
    <DESCRIPTION> Description for business event </DESCRIPTION>
    <IREP_ANNOTATION>

/*#
 * Put your long package description here; it can span multiple lines.
 *
 * @rep:scope <scope>
 * @rep:displayname <display name>
 * @rep:product <product or pseudoproduct short code>

```

```
* @rep:category BUSINESS_ENTITY <entity name>
*/
  </IREP_ANNOTATION>
</WF_EVENTS>
</WF_TABLE_DATA>

.
.
.
</oracle.apps.wf.event.all.sync>
```

### Example

For reference, here is an example of an annotated .wfx file:

```

<?xml version="1.0" encoding="UTF-8" ?>
- <!-- $Header: oeevtname.wfx 120.0 2005/06/01 23:11:59 appldev noship
$ -->
- <!-- dbdrv: exec java oracle/apps/fnd/wf WFXLoad.class java
&phase=daa+38 \ -->
- <!-- dbdrv: checkfile(115.2=120.0):~PROD:~PATH:~FILE \ -->
- <!-- dbdrv: -u &un_apps &pw_apps &jdbc_db_addr &jdbc_protocol US \
-->
- <!-- dbdrv: &fullpath_~PROD_~PATH_~FILE -->
- <oracle.apps.wf.event.all.sync>
- <ExternalElement>
- <OraTranslatibility>
- <XlatElement Name="WF_EVENTS">
- <XlatID>
  <Key>NAME</Key>
</XlatID>
  <XlatElement Name="DISPLAY_NAME" MaxLen="80" Expansion="50" />
- <XlatID>
  <Key Type="CONSTANT">DISPLAY_NAME</Key>
</XlatID>
  <XlatElement Name="DESCRIPTION" MaxLen="2000" Expansion="50" />
- <XlatID>
  <Key Type="CONSTANT">DESCRIPTION</Key>
</XlatID>
</XlatElement>
</OraTranslatibility>
</ExternalElement>
- <WF_TABLE_DATA>
+ <WF_EVENTS>
  <VERSION>1.0</VERSION>
  <GUID>A3BBD9D401776AE4E0340800208ACA52</GUID>
  <NAME>oracle.apps.ont.oi.po_ack.create</NAME>
  <TYPE>EVENT</TYPE>
  <STATUS>ENABLED</STATUS>
  <GENERATE_FUNCTION />
  <OWNER_NAME>Oracle Order Management</OWNER_NAME>
  <OWNER_TAG>ONT</OWNER_TAG>
  <CUSTOMIZATION_LEVEL>L</CUSTOMIZATION_LEVEL>
  <LICENSED_FLAG>Y</LICENSED_FLAG>
  <DISPLAY_NAME>Event for 3A4 Outbound Acknowledgment</DISPLAY_NAME>
  <DESCRIPTION>Event for 3A4 Outbound Acknowledgment</DESCRIPTION>
  <IREP_ANNOTATION>/# * This event confirms the buyer of the results of
order import. This event will start the workflow
OEOA/R_OEOA_SEND_ACKNOWLEDGMENT. * * @rep:scope public *
@rep:displayname Event for 3A4 Outbound Acknowledgment * @rep:product
ONT * @rep:category BUSINESS_ENTITY ONT_SALES_ORDER */</IREP_ANNOTATION>

  </WF_EVENTS>
</WF_TABLE_DATA>
- <WF_TABLE_DATA>
- <WF_EVENTS>
  <VERSION>1.0</VERSION>
  <GUID>9B8BF9DB705D09C9E0340800208ACA52</GUID>
  <NAME>oracle.apps.ont.oi.po_inbound.create</NAME>
  <TYPE>EVENT</TYPE>
  <STATUS>ENABLED</STATUS>
  <GENERATE_FUNCTION />
  <OWNER_NAME>Oracle Order Management</OWNER_NAME>
  <OWNER_TAG>ONT</OWNER_TAG>
  <CUSTOMIZATION_LEVEL>L</CUSTOMIZATION_LEVEL>
  <LICENSED_FLAG>Y</LICENSED_FLAG>

```

```

<DISPLAY_NAME>OM Generic Inbound Event</DISPLAY_NAME>
  <DESCRIPTION>OM Generic Inbound Event</DESCRIPTION>
  <IREP_ANNOTATION>/*# * This event is raised after the Purchase Order
has been pushed to Oracle Order management open interface tables. This
event will start the workflow OEOI/R_OEOI_ORDER_IMPORT to import the
order. * * @rep:direction OUT * @rep:scope public * @rep:displayname OM
Generic Inbound Event * @rep:lifecycle active * @rep:product ONT *
@rep:compatibility S * @rep:category BUSINESS_ENTITY ONT_SALES_ORDER
*/</IREP_ANNOTATION>
</WF_EVENTS>
</WF_TABLE_DATA>
</oracle.apps.wf.event.all.sync>

```

## Business Entity Annotation Guidelines

*Business entities* are things that either perform business activities or have business activities performed on them. Account numbers, employees, purchase orders, customers, and receipts are all examples of business entities.

### What Is the Importance of Business Entities?

Business entities are highly desired search criteria in the context of the Integration Repository. The design of the Integration Repository UI includes "browse by business entity" functionality.

### Where Do Business Entities Appear in Repository Annotations?

The `rep:category BUSINESS_ENTITY` annotation is where you associate a given interface with a business entity. For a general description of the `rep:category` annotation, see `rep:category`, page A-125.

**Note:** In certain cases where the entity's display name itself is sufficiently self-descriptive, it can serve as the description as well.

### Existing Business Entities

Custom integration interfaces can use only seeded or existing business entities.

**Note:** Integration Repository currently does not support the creation of custom Product Family and custom Business Entity.

The following table lists the existing business entities:

#### List of Business Entities

BUSINESS_ENTITY_CODE	MEANING	DESCRIPTION
AHL_DOCUMENT	Document	Electronic Document or Document Reference

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
AHL_ITEM_COMPOSITION	Tracked Item Composition	It is the list of item groups or non-tracked items that a tracked item is composed of.
AHL_ITEM_GROUP	Alternate Item Group	A group of similar items where one can be interchanged for another while performing maintenance.
AHL_MAINT_OPERATION	Maintenance Operation	It defines resource and material requirements. It is basic definition of work.
AHL_MAINT_REQUIREMENT	Maintenance Requirement	It is maintenance requirement definition. It defines routes, applicability on item or unit instances. It also defines frequency based on time and counters.
AHL_MAINT_ROUTE	Maintenance Route	It contains set of operations, and defines dispositions, resource and material requirements.
AHL_MAINT_VISIT	Maintenance Visit	It connects an unit or item instance with a block of tasks. It is an organization and department where the maintenance work takes place, and when the work is to be accomplished.
AHL_MAINT_WORKORDER	Maintenance Workorder	Maintenance Workorder with a schedule
AHL_MASTER_CONFIG	Master Configuration	A Master Configuration models the structure of an electromechanical system assembly.



<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
AHL_OSP_ORDER	Outside Service Order	An order that contains the information required to service parts by a third party organization.
AHL_PROD_CLASS	Product Classification	It is the categorization of units or items pertaining to maintenance and usage.
AHL_UNIT_CONFIG	Unit Configuration	An Unit Configuration describes the structure of an assembled electromechanical system.
AHL_UNIT_EFFECTIVITY	Unit Maintenance Plan Schedule	Unit Maintenance Plan with a due date
AHL_UNIT_SCHEDULES	Unit Usage Event	Event describes usage of a configured unit for a specific time period, such as an airplane flight.
AME_ACTION	Approval Action	Approval Action specifies an action to be performed, if the conditions of an approval rule is satisfied. For example, 'Require approvals up to the first three superiors'.
AME_APPROVAL	Approval	Approval
AME_APPROVER_GROUP	Approvals Management Approver Group	A predefined group of approvers who will be assigned to approve actions of specific business processes/transactions.
AME_APPROVER_TYPE	Approver Type	Classification of approvers who can be used in Approvals Management. For example, all HR employees are classified as the approver type as PER in Approvals Management.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
AME_ATTRIBUTE	Approvals Management Attribute	Object to capture business attributes for a transaction which requires approval. For example, INVOICE_AMOUNT can be an attribute which captures the total amount of an invoice.
AME_CONDITION	Approval Rule Condition	Condition based on the Approvals Management attribute that evaluates the approval rules. An example of condition on the attribute INVOICE_AMOUNT can be "INVOICE_AMOUNT > 10,000 USD".
AME_CONFIG_VAR	Approval Configuration Variable	A set of approval configurations which controls certain behavior within Approvals Management.
AME_ITEM_CLASS	Approvals Management Item Class	It is the classification of certain Approval Management objects into different classes like Header, Line Item, Cost Center.
AME_RULE	Approvals Business Rule	Approval Business rule consisting of a set of conditions, when satisfied, will dictate some actions to happen (which will result in a list of approvers).
AME_TRANSACTION_TYPE	Approval Transaction Type	A set of approval attributes, conditions, and rules making up a approval policy.
AMS_BUDGETS	Marketing Budget	It is the budget for Marketing Campaigns, Events, and other marketing activities.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
AMS_CAMPAIGN	Marketing Campaign	Marketing Campaign
AMS_EVENT	Marketing Event	Marketing Event
AMS_LEAD	Sales Lead	Sales Lead
AMS_LIST	Marketing List	Marketing List
AMS_METRIC	Marketing Metric	It is a measurement of marketing operations, such as, number of responses generated by a campaign.
AP_INVOICE	Payables Invoice	Payables Invoice
AP_PAYMENT	Supplier Payment	Supplier Payment
AP_PAYMENT_ADVICE	Payment Advice	Payment Advice
AP_SUPPLIER	Supplier	Supplier
AP_SUPPLIER_CONTACT	Supplier Contact	Supplier Contact
AP_SUPPLIER_SITE	Supplier Site	Supplier Site
AR_ADJUSTMENT	Receivables Invoice Adjustment	Receivables Invoice Adjustment
AR_BILLS_RECEIVABLE	Bills Receivable	Bills Receivable
AR_CHARGEBACK	Chargeback	Chargeback
AR_CREDIT_MEMO	Credit Memo	Credit Memo
AR_CREDIT_REQUEST	Credit Request	Credit Request
AR_DEBIT_MEMO	Debit Memo	Debit Memo
AR_DEPOSIT	Deposit	Deposit

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
AR_INVOICE	Receivables Invoice	Receivables Invoice
AR_PREPAYMENT	Prepayment	Prepayment
AR_RECEIPT	Receivables Receipt	Receivables Receipt
AR_REMITTANCE	Remittance	Remittance
AR_REVENUE	Revenue	Revenue
AR_SALES_CREDIT	Sales Credit	Sales Credit
AR_SALES_TAX_RATE	Sales Tax Rate	Sales Tax Rate
ASN_OPPORTUNITY	Sales Opportunity	Sales Opportunity
ASN_SALES_TEAM	Sales Team	Sales Team on an Opportunity or an Account, or a Lead
ASO_QUOTE	Sales Quote(1)	A sales quote is a business object that contains detailed information on the products, prices, terms, etc. in the solution proposed to potential customers(1).
AS_OPPORTUNITY	Sales Opportunity(1)	Sales Opportunity(1)
BEN_CWB_3RD_PARTY_STOCK_OPTS	Third Party Stock Option	Third Party Stock Options
BEN_CWB_AUDIT	Compensation Workbench Audit	It records every change event within a Compensation Workbench user session. This covers all compensation elements.
BEN_CWB_AWARD	Compensation Workbench Award	It is an employee monetary award. For example, salary raise, salary bonus, or shares.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
BEN_CWB_BUDGET	Compensation Workbench Budget	it is the budget of money or shares available for a manager to distribute including base salaries and bonuses.
BEN_CWB_PERSON	Compensation Workbench Person	Snapshot of a HR person on a specific date, for Compensation Workbench processing.
BEN_CWB_PLAN	Compensation Workbench Plan	It is a Compensation Plan, such as Salary Raise Plan, Bonus Plan or Stock Option Plan.
BEN_CWB_TASK	Compensation Workbench Task	It is the task performed in managing a Compensation Workbench Plan. For example, budgeting, allocation of amounts, submitting work and approval.
BIS_REPORT	BIS Report	BIS Report
BOM_BILL_OF_MATERIAL	Bill of Material	This interface adds, changes, and deletes Bill of Material of any type.
BOM_MFG_ROUTING	Product Manufacturing Routing	A routing defines the step-by-step operations required to produce an assembly in accordance with its Bill of Material.
BOM_PRODUCT_FAMILY	Product Family	Product Family for Planning Purposes
CAC_APPOINTMENT	Appointment	Appointment or Meeting for a given date and time period

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
CAC_BUSINESS_OBJECT_META_DATA	Business Object Meta Data Definition	Metadata definition for a Business Entity. It is used to dynamically link to external business entities. Also it is used for querying entity details, building dynamic LOVs and search pages.
CAC_CAL_TASK	Calendar Task	Task that will appear on User's Calendar as a time Blocking Task or a Todo.
CAC_NOTE	Note	Notes or Comments associated to different Business Objects
CAC_RS_TIME_BOOKING	Resource Time Booking	Time Booking for Person and non Person (e.g. Conference room) Resources
CAC_SCHEDULE	Schedule	Schedule
CAC_SCHEDULE_TEMPLATE	Schedule Template	Schedule Template
CAC_SYNC_SERVER	Calendar Synchronization Server	Calendar server to synchronize calendar entities like Task, Appointments, Contacts etc. to external calendars.
CAC_TASK_TEMPLATE	Calendar Task Template	Calendar Task Template
CCT_ADVANCED_TELEPHONY_SDK	Advanced Telephony SDK	This SDK allows telephony integration with Oracle E-Business Suite using server side integration.
CCT_BASIC_TELEPHONY_SDK	Basic Telephony SDK	This SDK allows telephony integration with Oracle E-Business Suite using client side integration.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
CE_BANK_STATEMENT	Bank Statement	Bank Statement
CE_RECONCILIATION_ITEM	Reconciliation Item	Reconciliation Item
CHV_PLANNING_SCHEDULE	Buyer Forecast	Buyer Forecast
CHV_SHIPPING_SCHEDULE	Buyer Shipment Request	Buyer Shipment Request
CLN_TRADING_PARTNER_COLL	Collaboration Trading Partner	Trading Partner
CLN_TRADING_PARTNER_COLL_EVENT	Trading Partner Collaboration Event	Trading Partner Collaboration Event
CN_COMP_PLANS	Incentive Compensation Plan	Incentive Compensation Plan
CN_INCENTIVES	Incentive Compensation	Variable compensation or rebates that can be monetary or non-monetary rewards for sales people, partners or customers.
CSD_REPAIR_ESTIMATE	Repair Estimate	Repair Estimate shows the total cost for the repair execution, which can include material, labor and expense charge lines.
CSD_REPAIR_LOGISTICS	Repair Logistics	Repair Logistics track the receiving and shipping of the customer item being repaired and also the items being loaned.
CSD_REPAIR_ORDER	Repair Order(1)	Repair Order(1)
CSF_TASK_DEBRIEF	Service Task Debrief	Service task debrief of material, labor and expense

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
CSI_COUNTER	Counters	It provides a mechanism to define and maintain different types of Matrixes. These can be attached to objects in the Oracle E-Business Suite like Installed Base Instances, or Service Contract Lines.
CSI_ITEM_INSTANCE	Item Instance	Install Base Item Instance
CST_DEPARTMENT_OVERHEAD	Manufacturing Department Overhead Rate	Manufacturing Department Overhead Rate
CST_ITEM_COST	Inventory Item Cost	Inventory Item Cost
CST_RESOURCE_COST	Manufacturing Resource Unit Cost	Manufacturing Resource Unit Cost
CS_SERVICE_CHARGE	Service Charge	Service Charge
CS_SERVICE_REQUEST	Service Request	Service Request
CZ_CONFIG	Configuration	Configuration
CZ_CONFIG_MODEL	Configuration Model	Configuration Model
CZ_MODEL_PUB	Configuration Model Publication	Configuration Model Publication
CZ_RP_FOLDER	Configurator Repository Folder	Configurator Repository Folder
CZ_USER_INTERFACE	Configuration Model User Interface	Configuration Model User Interface
DPP_EXECUTION_REQUEST	Execution Integration Request	It is an entity for integration of DPP with other Applications. It is used by event invoked from DPP UI and concurrent programs for integration with external applications like AR, AP, etc.



BUSINESS_ENTITY_CODE	MEANING	DESCRIPTION
DPP_TRANSACTION_APPR OVAL	Transaction Approval Notification	This entity is defined for the AME Approval for DPP transaction. It is referenced in UI on clicking of the <b>Request Approval</b> button in a New DPP transaction.
DPP_XMLG_OUTBOUND	Outbound pre-approval process	It is an entity used by events to trigger preapproval process through Oracle XML Gateway for Price Protection.
EAM_ASSET_ACTIVITY_AS SOCIATION	Maintenance Asset Activity Association	Maintenance Asset Activity Association
EAM_ASSET_ACTIVITY_SU PPRESSION	Asset activity suppression relations	It indicates that an asset preventive maintenance activity is suppressed due to the performance of another activity.
EAM_ASSET_AREA	Maintenance Asset Area	Maintenance Asset Area
EAM_ASSET_ATTRIBUTE_G ROUPS	Maintenance Asset Attribute Group	Maintenance Asset Attribute Group
EAM_ASSET_ATTRIBUTE_V ALUE	Maintenance Asset Attribute Value	Maintenance Asset Attribute Value
EAM_ASSET_METER	Maintenance Asset Meter Association	Maintenance Asset Meter Association
EAM_ASSET_NUMBER	Maintenance Asset Number	Maintenance Asset Number
EAM_ASSET_ROUTE	Maintenance Asset Route	Maintenance Asset Route
EAM_COMPLETE_WO_OPE RATION	Maintenance Work Completion	Maintenance Work Completion
EAM_DEPARTMENT_APPR OVER	Maintenance Department Approver	Maintenance Department Approver - User or responsibility

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
EAM_METER	Meter	Meter
EAM_METER_READING	Meter Reading	Meter Reading
EAM_PARAMETER	Maintenance Setup	Maintenance Setup
EAM_PM_SCHEDULE	Preventive Maintenance Schedule	Preventive Maintenance Schedule
EAM_SET_NAME	Maintenance Set	Maintenance Set
EAM_WORK_ORDER	Asset Maintenance Work Order	Asset Maintenance Work Order
EAM_WORK_REQUEST	Maintenance Work Request	Maintenance Work Request
ECX_CONFIRM_BOD	XML Gateway Confirmation Message	XML Gateway Confirmation Message
ECX_MESSAGE_DELIVERY	XML Gateway Message Delivery	It is used by both Oracle and non Oracle messaging systems to report delivery status. Status information is written to XML Gateway log tables to track and report transaction delivery data.
ECX_TRADING_PARTNER	XML Gateway Message Delivery(1)	It is used by both Oracle and non Oracle messaging systems to report delivery status. Status information is written to XML Gateway log tables to track and report transaction delivery data(1).
ECX_TRANSFORMATION	XML Gateway Transformation	This interface is used to apply a style sheet to an XML message and return the transformed XML message for further processing by the calling environment.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
EC_CODE_CONVERSION	Code Conversion	It converts Oracle's Internal Codes to External System Codes and vice-versa, such as Currency Code, Unit Of Measure.
EC_EDITION_TRANSACTION_LAYOUT	EDI Transaction Layout Definition Report	EDI Transaction Layout Definition Report
EC_INBOUND	Inbound EDI Message	It is an EDI message sent to the system from a trading partner.
EC_OUTBOUND	Outbound EDI Message	It is an EDI message sent from the system to a trading partner.
EC_TP_MERGE	Trading Partner Merge	It indicates a merge of Trading Partners as a result of an account merge in the Trading Community Architecture (TCA).
EDR_EVIDENCE_STORE	E-Records Evidence Store	E-Records Evidence Store
EDR_ISIGN_FILE_UPLOAD	File Upload Approval Request	File Upload Approval Request
EGO_ITEM	Catalog Item	An item that is listed in the Item Catalog.
EGO_USER_DEFINED_ATTR_GROUP	PLM User Defined Attributes	This interface adds, changes, deletes, and queries User-defined attributes for any entity.
ENG_CHANGE_ORDER	Product Change Order	Product or Engineering Change
FA_ASSET	Asset	The interface for adding assets to Oracle Assets.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
FA_CAPITAL_BUDGET	Capital Budget	The interface for uploading capital budgets to Oracle Assets.
FA_LEASE_PAYMENT	Lease Payment	The interface for sending lease payment lines to Oracle Payables.
FEM_ACCOUNT_FACT	Analytic Account Information	Detail level financial account data
FEM_BALANCES_FACT	Analytic Balances	It includes Ledger input and Ledger Profitability processing results.
FEM_FACT_REPOSITORY	Enterprise Analytical Fact Repository	It contains numeric facts (often called measurements) that can be categorized by multiple dimensions. It contains either detail-level facts or facts that have been aggregated.
FEM_STATISTICAL_FACT	Analytic Statistical Information	It contains dimensional numerical measures. These measures are actual statistical values, both derived and empirically obtained.
FEM_TRANSACTION_FACT	Analytic Transaction Information	The information represents counts of events and interactions for financial accounts.
FEM_XDIM_ACTIVITY	Analytic Activity	It describes repeatable tasks in relation to other dimensions. It is defined by an action and acted upon item. Business processes and actions of individuals can be categorized as activities.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
FEM_XDIM_AUXILIARY	Auxiliary Analytic Dimensions	It indicates the "non-foundation" dimensions for the Enterprise Performance Foundation. Unlike Foundation dimensions, they are not employed by calculation engines for value-added processing.
FEM_XDIM_BUDGET	Analytic Budget	It identifies budgets and forecasts.
FEM_XDIM_CAL_PERIOD	Analytic Calendar Period	Analytic Calendar Period
FEM_XDIM_CCTR_ORG	Analytic Organization	It indicates Standard Analytic Organization dimension made up of Company and Cost Center.
FEM_XDIM_CHANNEL	Analytic Channel	It identifies distribution and sales channels.
FEM_XDIM_COMPANY	Company Dimension	Standard Analytic Company dimension
FEM_XDIM_COST_CENTER	Cost Center Dimension	Standard Analytic Cost Center dimension
FEM_XDIM_COST_OBJECT	Analytic Cost Object	A Cost Object is a multidimensional entity that describes a cost.
FEM_XDIM_CUSTOMER	Analytic Customer	It identifies groups or individuals with a business relationship to analytic data.
FEM_XDIM_DATASET	Analytic Dataset	It identifies generic containers for analytic data.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
FEM_XDIM_ENTITY	Analytic Consolidation Entity	It identifies Consolidation, Elimination and Operating Entities for Global Consolidation System Users.
FEM_XDIM_FINANCIAL_ELEMENT	Analytic Financial Element	It identifies categories of amount types for balances, statistics and rates.
FEM_XDIM_GENERIC_FACT_DATA	Analytic User Defined Fact Data	Tables available for storing fact data of user defined dimensionality
FEM_XDIM_GEOGRAPHY	Analytic Geography	It identifies geographic locations.
FEM_XDIM_HIERARCHY	Analytic Dimension Hierarchy	It is organized parent-child relationships of dimension members.
FEM_XDIM_LEDGER	Analytic Ledger	It identifies books of account. It is analogous to a Set of Books.
FEM_XDIM_LEVEL	Analytic Dimension Level	It identifies categories for dimension members.
FEM_XDIM_LINE_ITEM	Analytic Line Item	It identifies general ledger accounts, typically as an extension to Natural Accounts.
FEM_XDIM_NATURAL_ACCOUNT	Analytic Natural Account	It identifies an account within an organization where balances are posted for the five different balance types of revenue, expense, owners equity, asset and liability.
FEM_XDIM_PRODUCT	Analytic Product	It identifies commodities or services offered for sale.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
FEM_XDIM_PROJECT	Analytic Project	It identifies plans and endeavors.
FEM_XDIM_SIC	Analytic Standard Industrial Classification	It identifies official codes of the Standard Industrial Classification system.
FEM_XDIM_SIMPLE	Analytic List of Values only Dimension	Grouping of all Analytic dimensions that have no attributes and serve only as lists of values.
FEM_XDIM_SOURCE_SYSTEM	Analytic Source System	It identifies the point of origin for fact and dimension data.
FEM_XDIM_TASK	Analytic Task	It identifies individual operations and pieces of work.
FEM_XDIM_USER_DIMENSION	Analytic User Defined Dimension	It is the grouping of all customizable analytic attributed dimensions.
FF_FORMULA_FUNCTION	Fast Formula Function	It represents an external procedural call providing arbitrary extensions to core Fast Formula functionality.
FLM_FLOW_SCHEDULE	Flow Schedule	Flow Schedule
FND_APPS_CTX	Oracle E-Business Suite Applications Security Context	Applications context representing current user session
FND_CP_PROGRAM	Concurrent Program	Discrete unit of work that can be run in the concurrent processing system. Typically, a concurrent program is a long-running, data-intensive task, such as generating a report.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
FND_CP_REQUEST	Concurrent Request	It is the request to the concurrent processing system to run a program with a given set of parameter values, an optional schedule to repeat, and optional postprocessing actions.
FND_CP_REQUEST_SET	Concurrent Request Set	A convenient way to run several concurrent programs with predefined print options and parameter values. Request sets group requests into stages that are submitted by the set.
FND_FLEX_KFF	Key Flexfield	Customizable multi-segment fields
FND_FORM	Oracle E-Business Suite Applications Form	A form is a special class of function that you may navigate to them using the Navigator window.
FND_FUNCTION	Oracle E-Business Suite Applications Function	A function is a part of an application functionality that is registered under an unique name for the purpose of providing function security.
FND_FUNC_SECURITY	Function Security	Function security restricts application functionality to authorized users.
FND_GFM	Oracle E-Business Suite Applications File	Generic file manager provides ways to upload/download files and manipulate the file attributes.
FND_LDAP_OPERATIONS	LDAP Directory	Enable Oracle E-Business Suite to performs operations against the integrated OID.



<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
FND_MENU	Oracle E-Business Suite Applications Menu	A hierarchical arrangement of functions and menus of functions that appears in the Navigator.
FND_MESSAGE	Oracle E-Business Suite Applications Message Dictionary	It contains catalog / repository of messages for the entire Oracle E-Business Suite. Message Dictionary facility is used to display and logging from application.
FND_NAVIGATION	Oracle E-Business Suite Applications Navigation	Standard ways of navigating from one page to another within applications
FND_OBJECT_CLASSIFICATION	OATM Object-Tablespace Classification	This entity stores seeded, explicit OATM object-tablespace classifications, which can be further customized.
FND_PROFILE	User Profile	It is a set of changeable options that affects the way the application behaves run time.
FND_RESPONSIBILITY	Responsibility	A responsibility defines the menu structure for a product in Oracle E-Business Suite.
FND_SSO_MANAGER	Single Sign On Manager	Single Sign On and Central Login related APIs
FND_TABLESPACE	Tablespace Model Tablespace	It classifies all storage-related objects. Logical Tablespaces have a 1:1 relation with physical tablespaces.
FND_USER	User	It represents a user of Oracle E-Business Suite.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
FUN_ARAP_NETTING	Payables and Receivables Netting	Payables and Receivables for Netting
FUN_IC_TRANSACTION	IC Manual Transaction	Intercompany transaction will be between one initiator and single/multiple recipients.
FUN_INTERCOMPANY_BATCH	Intercompany Transaction Set	It is intercompany batch containing transactions between legal entities.
FV_BUDGETARY_DISCOUNT	Federal Budgetary Discount	It creates Budgetary Discount Transactions.
FV_BUDGET_JOURNAL	Federal Budget Execution Document	It contains federal budget records imported into federal budgetary tables.
FV_FINANCE_CHARGE	Federal Finance Charge	Federal Finance Charge
FV_IPAC_DISBURSEMENT	IPAC Disbursement	IPAC Disbursement
FV_PRIOR_YEAR_ADJUSTMENT	Prior Year Adjustment	Prior Year Adjustment
FV_TREASURY_DISBURSEMENT	Treasury Disbursement	Treasury Confirmation, Backout and Void Disbursement Transactions
FV_YEAR_END_CLOSE	Federal Year End Closing Information	Federal Year End Closing
GHR_DUTY_STATION	US Federal Workplace Duty Station	US Federal Workplace Duty Station
GHR_EEO_COMPLAINT	US Federal EEO Complaint	US Federal EEO Complaint
GHR_POSITION_DESCRIPTION	Position Description	Position Description
GHR_REQ_FOR_PERSONNEL_ACTION	Request for Personnel Action	Request for Personnel Action

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
GL_ACCOUNTING_SETUP_MANAGER	Accounting Setup Manager	This represents the Accounting Setup of Ledgers and Legal Entities in General Ledger.
GL_ACCOUNT_COMBINATIONS	General Ledger Code Combination	This represents General Ledger Account Combinations Defined Under Chart of Accounts.
GL_BC_PACKETS	Budgetary Fund Control Transaction Packet	Budgetary Fund Control Transaction Packet
GL_BUDGET_DATA	General Ledger Budget Data	General Ledger Budget Data
GL_CHART_OF_ACCOUNTS	Chart of Accounts	Chart of Accounts (COA)
GL_DAILY_RATE	Daily Currency Conversion Rate	Daily Currency Conversion Rate
GL_INTERCOMPANY_TRANSACTION	Intercompany Transaction	Intercompany Transaction
GL_JOURNAL	Journal Entry	Journal Entry
GL_PERIOD	General Ledger Accounting Period	This represents the Accounting Period defined in Accounting Calendar.
GMD_ACTIVITIES_PUB	Product Development Activity	It creates, modifies, or deletes activity information.
GMD_FORMULA	Process Manufacturing Formula	Process Manufacturing Formula
GMD_OPERATION	Process Manufacturing Operation	Process Manufacturing Operation

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
GMD_OUTBOUND_APIS_P UB	Process Manufacturing Quality Outbound Transaction	It is public level Process Manufacturing Quality package containing APIs to export information to third party products.
GMD_QC_SAMPLES	Process Manufacturing Quality Sample	Process Manufacturing Quality Sample
GMD_QC_SPEC	Process Manufacturing Quality Specification	Process Manufacturing Quality Specification
GMD_QC_SPEC_VR	Process Manufacturing Specification Usage Rule	Process Manufacturing Specification Usage Rule
GMD_QC_TESTS_PUB	Process Manufacturing Quality Test	Process Manufacturing Quality Test
GMD_RECIPE	Process Manufacturing Recipe	Process Manufacturing Recipe
GMD_RECIPE_VALIDITY_R ULE	Process Manufacturing Recipe Usage Rule	Process Manufacturing Recipe Usage Rule
GMD_RESULTS_PUB	Process Manufacturing Quality Test Result	Process Manufacturing Quality Test Result
GMD_ROUTING	Process Manufacturing Routing	Process Manufacturing Routing
GMD_STATUS_PUB	Process Manufacturing Product Development Status	It modifies the status for routings, operations, receipts, and validity rules.
GME_BATCH	Process Manufacturing Batch	Process Manufacturing Batch
GME_BATCH_STEP	Process Manufacturing Batch Step	Process Manufacturing Batch Step
GMF_ALLOCATION_DEFIN ITION	Process Manufacturing Expense Allocation Definition	It is the setup data for allocating indirect expenses (indirect overheads) to items.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
GMF_BURDEN_DETAIL	Process Manufacturing Financials Overhead Detail	It indicates overhead costs assigned to items that have been manufactured or purchased.
GMF_ITEM_COST	Process Manufacturing Financials Item Cost	Process Manufacturing Financials Item Cost
GMF_RESOURCE_COST	Process Manufacturing Financials Resource Cost	Process Manufacturing Financials Resource Cost
GMI_ADJUSTMENTS	Process Manufacturing Inventory Adjustment	Process Manufacturing Inventory Adjustment
GMI_API	Process Manufacturing Inventory Setup	It is the Process Manufacturing Inventory transaction to create, modify, delete items, lots, lot conversions.
GMI_ITEM	Process Manufacturing Item	Process Manufacturing Item
GMI_ITEM_LOT_UOM_CON V	Process Manufacturing Item Lot UOM Conversion	Process Manufacturing Item Lot UOM Conversion
GMI_LOT	Process Manufacturing Lot	Process Manufacturing Lot
GMI_OM_ALLOC_API_PUB	Process Manufacturing Sales Order Inventory Allocation	The Allocate OPM Orders API is a business object that can create, modify, or delete OPM reservation (allocation) information for Order Management.
GMI_PICK_CONFIRM_PUB	Process Manufacturing Sales Order Inventory Pick Confirmation	The Pick Confirm API is a business object that pick confirms, or stages the inventory for a Process Move Order Line or a Delivery Detail line.
GMP_CALENDAR_API	Process Planning Shop Calendar	It modifies the Shop Calendar.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
GMP_GENERIC_RESOURCE	Generic Process Manufacturing Resource	Manufacturing resource in Process Manufacturing
GMP_PLANT_RESOURCE	Process Manufacturing Plant Resource	Plant specific manufacturing resource in Process Manufacturing
GMP_RSRC_AVL_PKG	Process Planning Resource Availability	It modifies resource availability.
GMS_AWARD	Project Award Budget	Project Award Budget
HR_AUTHORIA_INTEGRATION_MAP	Authoria Integration Map	Authoria Integration Map
HR_BUDGET	HR Budget	HR Budget
HR_BUSINESS_GROUP	Business Group	Business Group
HR_CALENDAR_EVENT	HR Calendar Event	HR Calendar Event
HR_COST_CENTER	Cost Center	Cost Center
HR_EVENT	HR Bookable Event	HR Bookable Event
HR_HELP_DESK	HR Help Desk Integration	Peoplesoft Help Desk Integration points with the Oracle E-Business Suite HRMS
HR_KI_MAP	Knowledge Integration Map	Knowledge Integration Map
HR_KI_SYSTEM	Knowledge Integration System	Knowledge Integration System
HR_LEGAL_ENTITY	Legal Entity	Legal Entity
HR_LIABILITY_PREMIUM	Liability Premium	Liability Premium
HR_LOCATION	Location	Location

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
HR_MESSAGE_LINE	HRMS Message Line	HRMS Message Line
HR_OPERATING_UNIT	Operating Unit	Operating Unit
HR_ORGANIZATION	HRMS Organization	HRMS Organization
HR_ORGANIZATION_LINK	Organization Link	Organization Link
HR_PAY_SCALE	Pay Scale	Pay Scale
HR_PERSON	HR Person(1)	HR Person(1)
HR_PERSONAL_DELIVERY_METHOD	Personal Delivery Method	Personal Delivery Method
HR_ROLE	HRMS Role	HRMS Role
HR_SALARY_BASIS	Salary Basis	Salary Basis
HR_SELF_SERVICE_TRANSACTION	HR Self Service Transaction	Self Service Transaction
HR_SOC_INS_CONTRIBUTIONS	Social Insurance Contribution	Social Insurance Contribution
HR_SUPER_CONTRIBUTION	Superannuation Contribution	It indicates payment to a fund providing for a person's retirement.
HR_USER_HOOK	HRMS User Hook	HRMS User Hook
HXC_TIMECARD	Timecard	Timecard
HXC_TIMECARD_RECURRING_PERIOD	Timecard Recurring Period	Timecard Recurring Period
HXC_TIME_INPUT_SOURCE	Time Input Source	It indicates how Timecard data was input.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
HXC_TIME_RECIPIENT	Time Recipient Application	An application that receives and processes Time and Labor Data.
HZ_ACCOUNT_CONTACT	Customer Account Contact	A person who is the contact for a customer account.
HZ_ADDRESS	Trading Community Address	It is an address of a trading community member, for example, a customer's or partner's address.
HZ_CLASSIFICATION	Trading Community Classification	It is a categorization of parties, using user-defined or external standards such as the NAICS, NACE, or SIC.
HZ_CONTACT	Trading Community Contact	A person who is a contact for an organization or another person.
HZ_CONTACT_POINT	Contact Point	It is a means of contact, for example, phone or e-mail.
HZ_CONTACT_PREFERENCE	Contact Preference	It is the information about when and how parties prefer to be contacted.
HZ_CUSTOMER_ACCOUNT	Customer Account	A person or organization that the deploying company has a selling relationship with.
HZ_EXTERNAL_REFERENCE	Trading Community External Reference	Management of operational mappings between the trading community database and external source systems.
HZ_GROUP	Trading Community Group	Trading Community Group
HZ_ORGANIZATION	Trading Community Organization	It is a party of type Organization and related information, including financial and credit reports.



<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
HZ_PARTY	Party	A trading community entity, either person or organization, that can enter into business relationships.
HZ_PERSON	Trading Community Person	It is a party of type Person and related information, such as employment and education.
HZ_RELATIONSHIP	Trading Community Relationship	A representation of how two parties are related, based on the role that each party plays with respect to the other.
HZ_RELATIONSHIP_TYPE	Trading Community Relationship Type	A categorization of roles that parties can play in relationships.
IBC_CONTENT_DELIVERY_MANAGER	Content Delivery Manager	Content Delivery Manager class provides APIs for applications to retrieve content items stored in the OCM Content Repository.
IBE_CATALOG_PUNCHOUT	Web Store Catalog Punchout	It is a process of enabling procurement users to choose items available in iStore catalog. The login/logout of procurement users in iStore is transparent to them.
IBE_CONTENT	Web Store Content	Web Store Page Content
IBE_ITEM	Web Store Item	Web Store Product Item
IBE_SALES_ORDER	Web Store Sales Order	Web Store Sales Order
IBE_SECTION	Web Store Section	Navigational Hierarchy for Web content and product
IBE_SESSION_ATTRIBUTES	Web Store Session Attributes	Session Attributes of Users visiting the Web Store

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
IBE_SHOPPING_CART	Web Store Shopping Cart	Web Store Shopping Cart
IBE_SHOPPING_LIST	Web Store Shopping List	Web Store Shopping List
IBE_SITE	Web Store Site	Web Store Site
IBE_TEMPLATE	Web Store Template	Web Store Page Template
IBE_USER	Web Store User	Users, Contacts, Customers
IBW_PAGE_ACCESS_TRACKING	Web Analytics Page Access Tracking	It captures visit and page access data required for Web analytics reporting.
IBY_BANKACCOUNT	External Bank Account	Supplier or Customer Bank Account
IBY_CREDITCARD	Credit Card	Credit Card Payment Instrument
IBY_EXCEPTION	IBY Exception	It is an exception generated by IBY code when an error is encountered.
IBY_FUNDCAPTURE_ORDER	Funds Capture Order	It is a single funds capture request delivered to a payment system by the request payee.
IBY_PAYMENT	IBY Payment	It indicates payment made through IBY to the supplier.
IEO_AGENT	Interaction Center Agent	A person that interacts with a customer during an interaction event.
IEX_COLLECTION_CASE	Collection Case	Collection Case

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
IEX_COLLECTION_DISPUTE	Collection Dispute	A dispute creates a credit memo request in Oracle Receivables to resolve all or part of an invoice that a customer contends is not owed.
IEX_COLLECTION_PROMISE	Collection Promise	Collection Promise
IEX_COLLECTION_SCORE	Collection Score	Collection Score
IEX_COLLECTION_STRATEGY	Collection Strategy	Collection Strategy
IEX_PROMISES	Collection Payment Promise	A promise to pay is a non-binding agreement from the customer to make a payment at a certain date.
IEX_STRATEGY	Receivables Collection Strategy	Strategies are a pre-configured sequence of work items that automate the process of collecting open receivables and support complex collections management activities.
IGC_CONTRACT_COMMITMENT	Contract Commitment	Contract Commitment
IGC_ENCUMBRANCE_JOURNAL	Encumbrance Journal	Encumbrance Journal
IGF_AWARD	Financial Aid Student Award	Financial Aid Student Award
IGF_BASE_RECORD	Financial Aid Student Base Record	Financial Aid Student Base Record
IGF_COA	Student Attendance Cost	Student Attendance Cost
IGF_DL	Financial Aid Direct Loan	Financial Aid Direct Loan

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
IGF_FFELP	Financial Aid FFELP Loan	Financial Aid FFELP Loan
IGF_FWS	Financial Aid Work Study	Financial Aid Work Study
IGF_ISIR	Institutional Student Information Record	Institutional Student Information Record
IGF_PELL	Financial Aid Pell Grant	Financial Aid Pell Grant
IGF_PROFILE	Student Profile Application	Student Profile Application
IGF_TODO	Financial Aid Student Todo Item(1)	Financial Aid Student Todo Item(1)
IGF_VERFN	Financial Aid Verification Item	Financial Aid Verification Item
IGS_ADM_APPLICATION	Admission Application	Admission Application
IGS_ADM_FEE	Admission Fee	Admission Application Fee
IGS_ADV_STAND	Advanced Standing	Advanced Standing
IGS_DA_REQUEST	Degree Audit Request	Degree Audit Request
IGS_INQ_APPLICATION	Prospective Applicant Inquiry	Prospective Applicant Inquiry
IGS_INSTITUTION	Institution	Institution Party
IGS_PARTY_CHARGE	Higher Education Party Charge	Higher Education Party Account Charge Transactions
IGS_PARTY_CREDIT	Higher Education Party Credit	Higher Education Party Account Credit Transactions
IGS_PARTY_REFUND	Higher Education Party Refund	Higher Education Party Account Refund Transactions
IGS_PERSON_ALTERNATE_ID	Alternate Person Identifier	Person Alternate Identifier e.g. SSN, Driver Licence etc

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
IGS_PERSON_CONTACT	Person Contact Information	Person Contact Information
IGS_PREV_EDUCATION	Previous Education	Previous Education
IGS_PROGRAM	Higher Education Program	Higher Education Program
IGS_SPONSORSHIP	Student Sponsor Relationship	Student Sponsor Relationship
IGS_STUDENT_CONCENTRATION	Student Concentration	Student Concentration
IGS_STUDENT_PROGRAM	Student Program Attempt	Student Program Attempt
IGS_STUDENT_UNIT	Student Unit Attempt	Student Unit Attempt
IGS_TODO	Financial Aid Student Todo Item	Financial Aid Student Todo Item
IGS_UNIT	Higher Education Unit	Higher Education Unit
IGW_PROPOSAL	Grants Proposal	Grants Proposal
IGW_PROPOSAL_BUDGET	Grants Proposal Budget	Grants Proposal Budget
INV_ACCOUNTING_PERIOD	Inventory Accounting Period	Status of an inventory accounting period
INV_ALLOCATION	Material Allocation	Inventory Material Allocation
INV_CONSIGNED_DIAGNOSTICS	Consigned Inventory Diagnostics	Set of utilities that identify and communicate inaccuracies in setup data of Consigned Inventory from Supplier feature.
INV_COUNT	Material Count	Material Count
INV_IC_TRANSACTION_FLOW	Inventory Intercompany Invoicing Transaction Flow	It is an execution of the transactions that generate intercompany invoices in Inventory.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
INV_IC_TRANSACTION_FLOW_SETUP	Intercompany Inventory Transaction Flow Setup	Intercompany Inventory Transaction Flow Setup
INV_LOT	Inventory Lot	Inventory Lot
INV_MATERIAL_TRANSACTION	Material Transaction	Inventory Material Transaction
INV_MOVEMENT_STATISTICS	Movement Statistics	Statistics that are associated with the movement of material across the border of two countries.
INV_MOVE_ORDER	Material Move Order	Physical movement of inventory from one location to another within a warehouse or other facility. It does not involve a transfer of the inventory between organizations.
INV_ONHAND	Inventory On Hand Balance	Inventory On Hand Balance
INV_ORGANIZATION_SETUP	Inventory Organization Setup	Inventory Organization Setup
INV_PICK_RELEASE_PUB	Inventory Pick Release	Inventory allocation in support of pick release
INV_POSITION	Inventory Position	It indicates on-hand balance of an Inventory Organization Hierarchy for a particular time bucket including quantity received, quantity issued and ending balance.
INV_REPLENISHMENT	Inventory Replenishment	Inventory Material Replenishment
INV_RESERVATION	Material Reservation	Inventory Material Reservation

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
INV_SALES_ORDERS	Inventory Sales Order	It indicates inventory sales order tracking with references to the order in Oracle Order Management or a third party order management system.
INV_SERIAL_NUMBER	Inventory Serial Number	Inventory Serial Number
INV_SUPPLIER_CONSIGNED_INV	Supplier Consigned Inventory	Goods that physically reside in an inventory organization but are owned by a supplier.
INV_UNIT_OF_MEASURE	Unit Of Measure	Inventory Unit Of Measure
IPM_DOCUMENT	Imaging Document	Electronic documentation to facilitate the entry and completion of transactions in the Oracle E-Business Suite.
IRC_AGENCY	Recruiting Agency	Third party agency authorized to recruit for a Vacancy.
IRC_CANDIDATE_NOTIFY_PREFS	Candidate Recruitment Notification Preferences	Candidate Recruitment Notification Preferences
IRC_CANDIDATE_SAVED_SEARCH	Candidate Recruitment Saved Search	Candidate Recruitment Saved Search
IRC_CANDIDATE_WORK_PREFERENCES	Candidate Recruitment Work Preferences	Candidate Recruitment Work Preferences
IRC_DEFAULT_JOB_POSTING	Default Job Posting	Default Job Posting
IRC_JOB_BASKET	Job Basket	Job Basket
IRC_JOB_OFFER	Job Offer	It contains details of a job to be offered to a Recruitment Candidate.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
IRC_JOB_OFFER_LETTER_TEMPLATE	Job Offer Letter Template	It is a template for a Job Offer letter.
IRC_JOB_OFFER_NOTES	Job Offer Note	Notes for a Job Offer
IRC_JOB_POSTING	Job Posting	Job Posting
IRC_JOB_SEARCH_LOCATION	Job Search Location	It contains locations for the Candidate Recruitment Saved Search or for the Candidate Recruitment Work Preferences.
IRC_JOB_SEARCH_PROF_AREA	Job Search Professional Area	It contains Professional Areas for the Candidate Recruitment Saved Search or for the Candidate Recruitment Work Preferences.
IRC_NOTIFICATION	iRecruitment Notification	Notifications that are sent to recruiter, interviewer and candidate.
IRC_RECRUITING_3RD_PARTY_SITE	Recruiting Third Party Site	Recruiting Third Party Site
IRC_RECRUITING_DOCUMENT	Recruiting Document	Recruiting Document
IRC_RECRUITING_SITE	Recruiting Site	Recruiting Site
IRC_RECRUITING_TEAM	Recruiting Team	Recruiting Team
IRC_RECRUITMENT_CANDIDATE	Recruitment Candidate	Recruitment Candidate
IRC_VACANCY_CONSIDERATION	Vacancy Consideration	Vacancy Consideration



<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
JE_ES_WHT	Spanish Withholding Tax Transaction	Spanish Withholding Tax Transaction stores withholding tax transactions from Payables and other external sources.
JL_BR_AP_BANK_COLLECTION_DOC	Brazilian Payables Bank Collection Document	Brazilian Payables Bank Collection Document
JL_BR_AR_BANK_RETURN_DOC	Brazilian Receivables Bank Return Document	Brazilian Receivables Bank Return Document
JTA_BUSINESS_RULE	Business Rule	Business Rule for Escalation or Auto Notifications
JTA_ESCALATION	Customer Escalation Management	It manages customer's escalation of some key business entities like Service Requests, Tasks, etc.
JTF_RS_DYNAMIC_GROUP	Resource Group (Dynamic)	Dynamic Resource Group (defined using dynamic SQL statements)
JTF_RS_DYNAMIC_GROUP	Resource Dynamic Group	Dynamic Resource Group (defined using dynamic SQL statements)
JTF_RS_GROUP	Resource Group	Grouping of Individual Resources
JTF_RS_GROUP_MEMBER	Resource Group Member	Members within a Group
JTF_RS_GROUP_MEMBER_ROLE	Resource Group Member Role	Roles assigned to members in a group
JTF_RS_GROUP_RELATION	Resource Group Hierarchy	Resource Group Hierarchy Element
JTF_RS_GROUP_USAGE	Resource Group Usage	Functional use of Resource Groups in different applications

BUSINESS_ENTITY_CODE	MEANING	DESCRIPTION
JTF_RS_RESOURCE	Individual Resource	Individual Resource
JTF_RS_RESOURCE_AVAILABILITY	Resource Availability	Whether an individual resource is available (Yes/No) for work assignments at present time.
JTF_RS_RESOURCE_LOV	Resource	Person and non-person resources
JTF_RS_RESOURCE_SKILL	Resource Skill	Resource Skillsets for work assignment
JTF_RS_RESOURCE_SKILL_LEVEL	Resource Skill Level	Skill Levels indicating Novice, Expert, Intermediate for different skills
JTF_RS_ROLE	Person Resource Role	Roles assigned to an individual resource
JTF_RS_ROLE_RELATION	Person Resource Role Hierarchy	Hierarchy of Roles associated with individual resources, groups, and group members.
JTF_RS_SALESREP	Sales Representative	Individual Resources that represent Enterprise Sales Force.
JTF_RS_SALES_GROUP_HIERARCHY	Sales Group Hierarchy	Sales Group Hierarchy
JTF_RS_SRP_TERRITORY	Sales Representative Territory	<p>Territories that are assigned to Sales people.</p> <p><b>Note:</b> These are not to be confused with Territory Manager.</p>
JTF_RS_TEAM	Resource Team	Teams represent collection of people, and groups.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
JTF_RS_TEAM_MEMBER	Resource Team Member	Members of a team that includes individuals, as well as groups.
JTF_RS_TEAM_USAGE	Resource Team Usage	It represents functional use of Resource Teams in different applications.
JTF_RS_UPDATABLE_ATTRIBUT	Updatable Attributes for Resources	Individual resource information that is allowed to be modified.
JTF_RS_WF_EVENT	Resource Business Event	Actions in Resource Manager that raise Workflow Business Events.
JTF_RS_WF_ROLE	Resource Workflow Role	It is the Workflow Role representing Individual, Group, and Team resources.
JTF_RS_WF_USER_ROLE	Resource Workflow User Role	It is the Workflow User Role representing resource roles, group members, or team members.
JTH_INTERACTION	Customer Interaction	It is the communication or attempted communication with a customer party.
JTH_INTERACTION_ACTIVITY	Customer Interaction Activity	A business event that occurs during an interaction with a customer party.
JTH_INTERACTION_MEDIA	Customer Interaction Media	It contains the details of the communication used in an interaction. Call, E-mail, Web, etc.
JTY_TERRITORY	Territory	Territories for sales representatives, service engineers and collections agents

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
LSH_APPLICATIONAREA	Application Area	A collection of objects that define a business application. Objects can be Business Area, Data Mart, Loadset, Program, Report Set, Table, Workflow etc.
LSH_BUSINESSAREA	Business Area	A Business Area acts as an interface with an External Visualization System (EVS). For example, a Discoverer Business Area is an interface with the Oracle Discoverer Visualisation tool.
LSH_DATAMART	Data Mart	A Data Mart stores data exported from the Transactional System and is usually used for Analytical purposes.
LSH_DOMAIN	Life Sciences Data Hub Domain	The top level container that owns Application Areas and is used to store object definitions in the Library. The definitions can be Business Area, Data Mart, Loadset, Program, etc.
LSH_EXECUTIONSETUP	Life Sciences Data Hub Execution Setup Information	It is a defined object that is a component of each LSH executable object instance (Programs, RS, Load Sets, Workflows etc.) whose purpose is to control the execution of the executable object.
LSH_EXECUTION_FWK	Life Sciences Data Hub Execution Job	An entity that provides the surround and the set the rules for the execution of an object. Examples are Job Submission API, Job Log API, etc

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
LSH_GENERIC_OBJECT	Life Science Data Hub Generic Object	Life Science Data Hub Generic Object
LSH_LOADSET	External Data Load Run	A Load Set is an executable object that is used to define the structure and behavior of a Program-like structure that is used for loading data from an outside system.
LSH_MAPPING	Life Science Data Hub Column Mapping	A mapping defines a column level mapping between a Table like object and a View like object.
LSH_OBJ_CLASSIFICATION	Object Classification	An entity that provides the categories and rules for classifying an object.
LSH_OBJ_SECURITY	Life Sciences Data Hub Object Security Policy	An entity that provides a set of rules to define and implement data security on objects.
LSH_OUTPUT	Life Sciences Data Hub Output	This is the actual Output that is generated on execution of an executable object, such as a Report Set output, a Data Mart output, a Program Output.
LSH_PARAMETER	API Parameter	A defined object that acts as a simple scalar variable and is based on a variable, such as an input/output Parameter of a Program, Report Set, etc.
LSH_PARAMETERSSET	Parameter Set	A collection of interrelated Parameters

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
LSH_PLANNEDOUTPUT	Life Science Data Hub Planned Output	A Planned Output is an expected output when an LSH object is executed. It is defined with the executable object.
LSH_PROGRAM	Life Sciences Data Hub Program	A Program is a metadata object that is used to define the structure and behavior of a Program-like structure that is used for processing and/or reporting on set of data.
LSH_REPORTSET	Report Set	A Report Set is a group of reports used to define the structure and behavior of a hierarchical structure that is intended for simultaneously reporting on sets of data.
LSH_SOURCECODE	Software Source Code	A Source Code is the actual program code which is executed when a Program is run.
LSH_TABLE	Metadata Registered Data Object	It is a metadata description of a table-like object (for example a Oracle view or a SAS dataset).
LSH_UTILITY	Life Sciences Data Hub Setup Utility	It is a set of tools and utilities.
LSH_VALIDATION	Life Sciences Data Hub Validation	An entity that provides the rules for validating an object in the application.
LSH_VARIABLE	Life Science Data Hub Variable	A LSH defined object equivalent to a SAS variable or Oracle table column that serves as a source definition for LSH Parameters and Table Columns.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
LSH_WORKAREA	Application Work Area	A container within an Application Area that provides the definer a place to prepare related LSH Definitional objects for release and installation to a LSH schema.
MES_COMPLETION_TRANSACTION	Assembly Completion in MES	Business Entity for Assembly Completion in MES
MES_MATERIAL_TRANSACTION	MES Material Transaction	Business Entity for Material Transaction in MES
MES_MOVE_TRANSACTION	MES Move Transaction	Business Entity for Move Transaction in MES
MES_TIME_ENTRY	Time Entry in MES	Import Time Entry Record in Discrete Manufacturing Execution system
MSC_ATP_ENQUIRY	ATP Enquiry	This interface checks the availability for the item(s) and returns their availability picture.
MSC_FORECAST	Supply Chain Forecast	This interface creates a forecast for supply chain planning.
MSC_NOTIFY_PLAN_OUTPUT	Supply Chain Planned Order	Supply chain planned order
MSC_ON_HAND	Supply Chain Plan On Hand Inventory	This interface creates on hand supply records for supply chain planning.
MSC_PLANNING_SUPPLY_DEMAND	Collaborative Planning Supply / Demand	This interface is used to create any supply / demand records in Collaborative Planning.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
MSC_PURCHASE_ORDER	Supply Chain Plan Purchase Order	This interface creates a purchase order supply for supply chain planning.
MSC_REQUISITION	Supply Chain Plan Requisition	A Purchase Requisition for Supply Chain Planning
MSC_SALES_ORDER	Supply Chain Plan Sales Order	This interface creates a sales order demand for supply chain planning.
MSC_SHIPMENT_NOTICE	Supply Chain Plan Advanced Shipment Notice	This interface creates an inbound intransit supply for supply chain planning.
MSC_WORK_ORDER	Supply Chain Plan Work Order	This interface creates a work order supply for supply chain planning.
NETTING_BATCH	Netting Batch	Netting batch is a set of payables and receivables transactions.
OCM_GET_DATA_POINTS	Credit Review Data Point	It is a list of Data Points (Criterion items against which the credit standing of a organization is reviewed) for a given credit classification, review type, data point category, or subcategory.
OCM_GET_EXTRL_DECSN_PUB	Imported Credit Score and Recommendation	Import score and recommendations from external source
OCM_GUARANTOR_CREDIT_REQUEST	Guarantor Credit Request	It allows user to create Guarantor credit request.



BUSINESS_ENTITY_CODE	MEANING	DESCRIPTION
OCM_RECOMMENDATIONS	Credit Recommendation	It is the recommendation/decision made by reviewer of the credit request. For example, a standard recommendation is "Approve/Reject".
OCM_WITHDRAW_CREDIT_REQUEST	Credit Request Withdrawal	It allows user to withdraw a credit request.
OIE_CREDIT_CARD_TRXN	Credit Card Transaction	Credit Card Transaction
OIE_PCARD_TRXN	Procurement Card Transaction	Procurement Card Transaction
OIR_REGISTRATION	Self Registration of user	Self Registration of external user of the application
OKC_DELIVERABLE	Contract Deliverable	Contract Deliverable
OKC_LIBRARY_ARTICLE	Contract Library Article	Contract Library Article
OKC_LIBRARY_CLAUSE	Contract Library Clause	Contract library clause
OKC_REPOSITORY_CONTRACT	Repository Contract	A contract that handles outside the normal purchasing or sales flows, such as a non-disclosure agreement or a partnership agreement. These contracts are stored in the Contract Repository.
OKC_REP_CONTRACT	Repository Contract(1)	A contract that handles outside the normal purchasing or sales flows, such as a non-disclosure agreement or partnership agreement. These contracts are stored in the Contract Repository(1).
OKE_CONTRACT	Project Contract	Project Contract

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
OKL_ACCOUNT_DISTRIBUTION	Lease Account Distribution	Lease Account Distribution
OKL_ACCOUNT_ID	Lease Account	Lease Account
OKL_AGREEMENT	Lease Agreement	Lease Agreement
OKL_ASSET_MANAGEMENT	Asset Management	Manage portfolios and asset returns
OKL_COLLECTION	Collection	Bill, collect cash and manage collections from customers
OKL_COLLECTION_CASE	Lease Collection Case	Lease Collection Case
OKL_CONTRACT	Lease Contract	Lease Contract
OKL_CONTRACT_LIFECYCLE	Contract Management Lifecycle	It manages revisions, termination and renewals of contracts.
OKL_CONTRACT_PARTY	Lease Contract Party	Lease Contract Party
OKL_CONTRACT_PAYMENT	Lease Contract Payment	Lease Contract Payment
OKL_CONTRACT_TERM	Lease Contract Term	Lease Contract Term
OKL_DISBURSEMENT	Disbursement	Process manually initiated or automated disbursements
OKL_EXECUTE_FORMULA	Lease Formula	Lease Formula
OKL_FINANCIAL_PRODUCT	Lease Contract Financial Product	Financial Product specified in lease contract
OKL_INSURANCE	Lease Insurance	Lease Insurance
OKL_INTEREST	Lease Interest	Lease Interest

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
OKL_INVESTMENT_PROGR AM	Manage Investment Program	It manages investor accounts and investment agreements.
OKL_LATE_POLICY	Lease Late Payment Policy	Lease Late Payment Policy
OKL_LEASE_RATE	Lease Rate Set	Lease Rate Set
OKL_MARKETING_PROGR AM	Marketing Program	It manages internal and partner pricing programs.
OKL_ORIGINATION	Origination	It manages master agreements, author lease and loan contracts.
OKL_REMARKETING	Remarketing	It manages sale of assets to vendors and third parties.
OKL_RESIDUAL_VALUE	Lease Residual Value	Lease Residual Value
OKL_RISK_MANAGEMENT	Risk Management	It manages credit, pricing, approval and insurance policies.
OKL_SALES	Sales	Qualify, quote and manage deal opportunities
OKL_STREAM	Lease Stream	Lease Stream
OKL_TERMINATION_QUOT E	Lease Termination Quote	Lease Termination Quote
OKL_THIRD_PARTY_BILLI NG	Lease Third Party Billing	Lease Third Party Billing
OKL_UNDERWRITING	Manage Underwriting	It manages credit applications and lines.
OKL_VENDOR_RELATIONS HIP	Manage Vendor Relationship	It manages vendor accounts and agreements.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
OXS_AVAILABLE_SERVICE	Service Availability	APIs for retrieving customer service information, specifically, duration of a service, availability of service for a customer and list of services which can be ordered for a customer.
OXS_CONTRACT	Service Contract	Service Contract
OXS_COVERAGE	Service Contract Coverage	Service contract coverage service terms
OXS_IMPORT	Service Contracts Import	Service contracts import is a process of importing the legacy data into the Oracle tables.
OXS_ENTITLEMENT	Service Contract Entitlement	Service contract customer entitled services
ONT_SALES_AGREEMENT	Sales Agreement	This is a business document that outlines the agreement between a Customer and Supplier committing to order and deliver a specified amount or quantity over an agreed period of time.
ONT_SALES_ORDER	Sales Order	Sales Order is a business document containing customer sales order information. This entity is used by several Oracle E-Business Suite applications.
OTA_CATALOG_CATEGORY	Learning Catalog Category	Learning Catalog Category
OTA_CERTIFICATION	Learning Certification	Catalog object that offers learners the opportunity to subscribe to and complete one time and renewable certifications.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
OTA_CHAT	Learning Chat	Scheduled live discussion that enables learners and instructors to exchange messages online.
OTA_CONFERENCE_SERVER	Conference Server	Conference server integrates OLM with Oracle Web Conferencing (OWC) to deliver online synchronous classes.
OTA_COURSE_PREREQUISITE	Course Prerequisite	A course or competency that a learner must or should complete before enrolling in a given class.
OTA_ENROLLMENT_JUSTIFICATION	Learning Enrollment Justification	Each enrollment justification and its associated priority level can determine the order by which enrollees are automatically placed in a class.
OTA_ENROLLMENT_STATUS_TYPE	Learning Enrollment Status Type	It indicates predefined enrollment statuses (Requested, Placed, Attended, Waitlisted, Cancelled).
OTA_FINANCE_HEADER	Learning Finance Header	It is a record of a monetary amount against a class, a learner enrollment, or a resource booking.
OTA_FINANCE_LINE	Learning Finance Line	It is an individual financial transaction within a finance header.
OTA_FORUM	Learning Forum	It represents message board that learners and instructors use to post general learning topics for discussion.
OTA_LEARNER_ENROLLMENT	Learner Enrollment	Learner Enrollment

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
OTA_LEARNING_ANNOUNCEMENT	Learning Announcement	Learning Announcement
OTA_LEARNING_CATALOG_CAT_USE	Learning Catalog Category Usage	Learning Catalog Category Usage
OTA_LEARNING_CLASS	Learning Class	Learning Class
OTA_LEARNING_COURSE	Learning Course	Learning Course
OTA_LEARNING_CROSS_CHARGE	Learning Cross Charge Setup	Learning Cross Charge Setup
OTA_LEARNING_EXTERNAL	Learning External Record	A class or course that a person has attended, not scheduled in the internal learning catalog.
OTA_LEARNING_OFFERING	Learning Offering	Learning Offering
OTA_LEARNING_OFFERING_RESOURCES_CHKLIST	Learning Offering Resource Checklist	Learning Offering Resource Checklist
OTA_LEARNING_PATH	Learning Path	Learning Path
OTA_LEARNING_PATH_CATEGORY	Learning Path Category	Learning Path Category
OTA_LEARNING_PATH_COMPONENT	Learning Path Component	Learning Path Component
OTA_LP_SUBSCRIPTION	Learning Path Subscription	It contains subscriptions for all Learning Paths and Components. For Example, Subscriptions to Catalog Learning Paths and Learning Paths created by Managers from Appraisals, Suitability Matching etc.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
OTA_RESOURCE	Learning Resource	It is a person or an object needed to deliver a class, such as a named instructor or a specific classroom.
OTA_RESOURCE_BOOKING	Learning Resource Booking	Learning Resource Booking
OTA_TRAINING_PLAN	Training Plan	Training Plan
OZF_ACCOUNT_PLAN	Trade Account Plan	Account Plan for Trade Planning and Promotion activities
OZF_BUDGET	Sales and Marketing Budget	It is the budget for Promotional Offer, Marketing Campaigns, Events, and other marketing activities.
OZF_CLAIM	Trade Claim	Claims that customers could be seeking money against, such as Promotional claims, breakages, transportation errors etc.
OZF_INDIRECT_SALES	Indirect Sales	Point of Sales Data, chargebacks etc,
OZF_OFFERS	Promotional Offer	Promotional Offers or Discounts that are given to Customers from a Vendors Sales or Marketing Organization.
OZF_QUOTA	Trade Planning Quota	Quota and Targets for Trade Planning and Promotional Activities
OZF_SOFT_FUND	Partner Fund	Partner Fund Requests
OZF_SPECIAL_PRICING	Special Pricing	Special Pricing Requests

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
OZF_SSD_BATCH	Supplier Ship and Debit Batch	Supplier ship and debit batch is essentially a claim that the distributor submits to the supplier for approval and payment. The batch contains accruals for which the supplier is expected to make payment for.
OZF_SSD_REQUEST	Supplier Ship and Debit Request	An agreement that allows distributor to request a special price from supplier. The ship and debit request will allow specifications with respect to the quantity limits associated to the price reduction, product and period.
PAY_BALANCE	Payroll Balance	Payroll Balance
PAY_BALANCE_ADJUSTMENT	Payroll Balance Adjustment	Payroll Balance Adjustment
PAY_BATCH_ELEMENT_ENTRY	HRMS Batch Element Entry	HRMS Batch Element Entry
PAY_CONTRIBUTION_USAGE	Payroll Contribution Usage	Payroll Contribution Usage
PAY_COST_ALLOCATION	Payroll Cost Allocation	Payroll Cost Allocation
PAY_DEFINED_BALANCE	Payroll Defined Balance	Payroll Defined Balance
PAY_ELEMENT	HRMS Element	HRMS Element
PAY_ELEMENT_CLASSIFICATION	HRMS Element Classification	It describes categories of HRMS Elements such as Earnings, Deductions and Information. These influence subsequent processing.
PAY_ELEMENT_ENTRY	HRMS Element Entry	HRMS Element Entry



<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
PAY_ELEMENT_LINK	HRMS Element Eligibility Criteria	HRMS Element Eligibility Criteria
PAY_EMP_TAX_INFO	Employee Tax Information	Employee Tax Information
PAY_FORMULA_RESULT	Payroll Processing Result Rule	It indicates how the Formula is to be processed and how its result is to be used by the Payroll processes.
PAY_ITERATIVE_RULE	Payroll Iterative Rule	Payroll Iterative Rule
PAY_LEAVE_LIABILITY	Leave Liability	It describes leave type and associated definitions utilized by the Leave Liability process.
PAY_ORG_PAYMENT_METHOD	Organization Payment Method	It is a Payroll Payment Method used by the Organization for employee compensation.
PAY_PAYMENT_ARCHIVE	Payroll Payment Archive	Payroll Payment Archive
PAY_PAYROLL_DEFINITION	Payroll Definition	Payroll Definition
PAY_PAYROLL_EVENT_GROUP	Payroll Event Interpretation Group	Payroll Event Interpretation Group
PAY_PAYROLL_TABLE_RECORD_EVENT	Payroll Table Recordable Event	Payroll Table Recordable Event
PAY_PERSONAL_PAYMENT_METHOD	Personal Payment Method	Personal Payment Method
PAY_PROVINCIAL_MEDICAL	Provincial Medical Account	Provincial Medical Account
PAY_RUN_TYPE	Payroll Run Type	Payroll Run Type
PAY_TIME_DEFINITION	Payroll Time	This holds information about period of time and its usage.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
PAY_USER_DEFINED_TABLE	HRMS User Defined Table	HRMS User Defined Table
PAY_WORKERS_COMPENSATION	Workers Compensation	Workers Compensation
PA_AGREEMENT	Project Customer Agreement	Project Customer Agreement
PA_BILLING_EVENT	Project Billing Event	Project Billing Event
PA_BUDGET	Project Budget	Project Budget
PA_BURDEN_COST	Project Burden Cost	Project Burden Cost
PA_CAPITAL_ASSET	Project Capital Asset	Project Capital Asset
PA_CUSTOMER_INVOICE	Project Customer Invoice	Project Customer Invoice
PA_EXPENDITURE	Project Expenditure	Project Expenditure
PA_EXPENSE_RPT_COST	Project Expense Report Cost	Project Expense Report Cost
PA_FINANCIAL_TASK	Project Financial Task	Project Financial Task
PA_FORECAST	Project Forecast	Project Forecast
PA_IC_TRANSACTION	Project Cross Charge	Project Cross Charge
PA_INTERCOMPANY_INVOICE	Project Intercompany Invoice	Project Intercompany Invoice
PA_INTERPROJECT_INVOICE	Project Interproject Invoice	Project Interproject Invoice
PA_INVENTORY_COST	Project Inventory Cost	Project Inventory Cost
PA_INVOICE	Project Invoice	Project Invoice
PA_LABOR_COST	Project Labor Cost	Project Labor Cost

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
PA_MISCELLANEOUS_COST	Project Miscellaneous Cost	Project Miscellaneous Cost
PA_PAYABLE_INV_COST	Project Supplier Cost	Project Supplier Cost
PA_PERF_REPORTING	Project Reporting	Project Reporting
PA_PROJECT	Project	Project
PA_PROJ_COST	Project Cost	Project Cost
PA_PROJ_DELIVERABLE	Project Deliverable	Project Deliverable
PA_PROJ_FUNDING	Project Funding	Project Funding
PA_PROJ_PLANNING_RESOURCE	Project Planning Resource	Project Planning Resource
PA_PROJ_RESOURCE	Project Resource	Project Resource
PA_RES_BRK_DWN_STRUCTURE	Project Resource Breakdown Structure	Project Resource Breakdown Structure
PA_REVENUE	Project Revenue	Project Revenue
PA_TASK	Project Task	Project Task
PA_TASK_RESOURCE	Project Task Resource	Project Task Resource
PA_TOT_BURDENED_COST	Project Total Burdened Cost	Project Total Burdened Cost
PA_USAGE_COST	Project Asset Usage Cost	Project Asset Usage Cost
PA_WIP_COST	Project Work in Process Cost	Project Work in Process Cost
PA_WORKPLAN_TASK	Project Workplan Task	Project Workplan Task
PER_APPLICANT	Applicant	Applicant
PER_APPLICANT_ASG	Applicant Assignment	Applicant Assignment

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
PER_APPRAISAL	Worker Appraisal	Worker Appraisal
PER_APPRAISAL_PERIOD	Appraisal Period	It defines appraisal period information to be used within a performance plan.
PER_ASSESSMENT	Worker Assessment	Worker Assessment
PER_BF_BALANCE	Third Party Payroll Balance	Third Party Payroll Balance
PER_BF_PAYROLL_RESULTS	Third Party Payroll Results	Third Party Payroll Results
PER_CHECKLIST	Person Task Checklist	It is a checklist containing tasks which can be copied and the copy assigned to an Employee, Contingent Worker or Applicant, e.g. 'New Hire Checklist'.
PER_COLLECTIVE_AGREEMENT	Collective Agreement	Collective Agreement
PER_COLLECTIVE_AGREEMENT_ITEM	Collective Agreement Item	Collective Agreement Item
PER_COMPETENCE	Competence	Competence
PER_COMPETENCE_ELEMENT	Competence Element	Competence Element
PER_COMPETENCE_RATING_SCALE	Competence Rating Scale	Competence Rating Scale
PER_CONFIG_WORKBENCH	HCM Configuration Workbench	It manages enterprise structure configuration workbench wizard for setting up entities such as Locations, Business Groups, Jobs and Positions.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
PER_CONTACT_RELATIONSHIP	Contact Relationship	Contact Relationship
PER_CWK	Contingent Worker	Contingent Worker
PER_CWK_ASG	Contingent Worker Assignment	Contingent Worker Assignment
PER_CWK_RATE	Contingent Worker Assignment Rate	Contingent Worker Assignment Rate
PER_DISABILITY	Disability	Disability
PER_DOCUMENTS_OF_RECORD	Documents of Record	Documents of Record for an Employee, Contingent Worker, Applicant or Contact
PER_EMPLOYEE	Employee	Employee
PER_EMPLOYEE_ABSENCE	Employee Absence	Employee Absence
PER_EMPLOYEE_ASG	Employee Assignment	Employee Assignment
PER_EMPLOYMENT_CONTRACT	Employment Contract	Employment Contract
PER_ESTAB_ATTENDANCES	Schools and Colleges Attended	Schools and Colleges Attended
PER_EX-EMPLOYEE	Ex-Employee	Ex-Employee
PER_GENERIC_HIERARCHY	Generic Hierarchy	Generic Hierarchy
PER_GRADE	Employee Grade	Employee Grade
PER_JOB	Job	Job
PER_JOB_GROUP	Job Group	Job Group

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
PER_MEDICAL_ASSESSMENT	Medical Assessment	Medical Assessment
PER_OBJECTIVE_LIBRARY	Objectives Library	A repository of reusable objectives that can be either created individually or imported from an external source.
PER_ORGANIZATION_HIERARCHY	Organization Hierarchy	Organization Hierarchy
PER_PERFORMANCE_REVIEW	Employee Performance Review	Employee Performance Review
PER_PERF_MGMT_PLAN	Performance Management Plan	It indicates the parameters of the performance management process, including the performance period, population and appraisal periods.
PER_PERSON	HR Person	HR Person
PER_PERSONAL_CONTACT	Personal Contact	Personal Contact
PER_PERSONAL_SCORECARD	Person Scorecard	One worker's objectives for a performance management plan, which provides a goal setting, performance review and scoring basis.
PER_PERSON_ADDRESS	Person Address	Person Address
PER_PHONE	Phone	Phone
PER_POSITION	Position	Position
PER_POSITION_HIERARCHY	Position Hierarchy	Position Hierarchy

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
PER_PREVIOUS_EMPLOYMENT	Previous Employment	Previous Employment
PER_QUALIFICATION	Person Qualification	Person Qualification
PER_RECRUITMENT_ACTIVITY	Recruitment Activity	Recruitment Activity
PER_SALARY_PROPOSAL	Salary Proposal	Salary Proposal
PER_SALARY_SURVEY	Salary Survey	Salary Survey
PER_SCORECARD_SHARING	Scorecard Access	It holds the list of persons and access permissions for a scorecard for which the owner of the scorecard has granted access.
PER_SECURITY_PROFILE	Security Profile	Security Profile
PER_SUPPLEMENTARY_ROLE	HR Supplementary Role	HR Supplementary Role
PER_VACANCY	Vacancy	Vacancy
PER_VACANCY_REQUISITION	Vacancy Requisition	Vacancy Requisition
PER_WORK_COUNCIL_ELECTION	Work Council Election	Work Council Election
PER_WORK_INCIDENT	Work Incident	Work Incident
PN_CUSTOMER_SPACE_ASSIGNMENT	Customer Space Assignment	Customer Space Assignment
PN_EMPLOYEE_SPACE_ASSIGNMENT	Employee Space Assignment	Employee Space Assignment
PN_PROPERTY	Space	A property or component of property, such as a building, land parcel, floor, or office.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
PN_RECOVERABLE_EXPENSE	Property Recoverable Expense	Property Recoverable Expense
PN_VOLUME_HISTORY	Variable Rent Volume History	Variable Rent Volume History
PO_ACKNOWLEDGEMENT	Purchase Order Acknowledgement	Purchase Order Acknowledgement
PO_ADVANCED_SHIP_NOTIFICATION	Advanced Shipment Notification	Advanced Shipment Notification
PO_APPROVAL	Purchase Order Approval	Purchase Order Approval
PO_APPROVAL_HIERARCHY	Purchase Order Approval Hierarchy	Purchase Order Approval Hierarchy
PO_APPROVED_SUPPLIER_LIST	Approved Supplier List	Approved Supplier List
PO_ATTACHMENTS	Procurement Attachments	Procurement Attachments
PO_AUCTION	Auction	Auction
PO_AWARD	Award	Award
PO_BIDDING_ATTRIBUTES	Bidding Attributes	Bidding Attributes
PO_BLANKET_PURCHASE_AGREEMENT	Blanket Purchase Agreement	Blanket Purchase Agreement
PO_BLANKET_RELEASE	Purchasing Blanket Release	A blanket release is issued against a blanket purchase agreement to place the actual order.
PO_CATALOG	Purchasing Catalog	Purchasing Catalog
PO_CATALOG_CATEGORY	Purchasing Catalog Category	Purchasing Catalog Category
PO_CHANGE	Purchase Order Change	Purchase Order Change



<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
PO_CONSUMPTION_ADVICE	Consigned Inventory Consumption Advice	Release or Standard PO for Consigned Consumption
PO_CONTRACT	Purchasing Contract	Purchasing Contract
PO_CONTRACT_PURCHASE_AGREEMENT	Contract Purchase Agreement	Contract Purchase Agreement
PO_CONTRACT_TEMPLATE	Purchasing Contract Template	Purchasing Contract Template
PO_CONTRACT_TERM	Purchasing Contract Term	Purchasing Contract Term (Articles, Deliverables and Contract Documents)
PO_DOCUMENT_APPROVER	Purchasing Document Approver	Purchasing Document Approver
PO_EXPENSE_RECEIPT	Expense Receipt	Expense Receipt
PO_GLOBAL_BLANKET_AGREEMENT	Global Blanket Purchase Agreement	Global Blanket Purchase Agreement
PO_GLOBAL_CONTRACT_AGREEMENT	Global Contract Purchase Agreement	Global Contract Purchase Agreement
PO_GOODS_RECEIPT	Goods Receipt	Goods Receipt
PO_GOODS_RETURN	Goods Return	Goods Return
PO_INTERNAL_REQUISITION	Internal Requisition	Internal Requisition
PO_NEGOTIATION	Sourcing Negotiation	Sourcing Negotiation
PO_PLANNED_PURCHASE_ORDER	Planned Purchase Order	Planned Purchase Order
PO_PLANNED_RELEASE	Planned PO Release	Planned PO Release
PO_PRICE_BREAKS	Sourcing Price Break	Sourcing Price Break

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
PO_PRICE_DIFFERENTIAL	Purchasing Price Differential	Purchasing Price Differential holds the price differentials for the rate based lines for requisition lines, PO lines or Blanket pricebreaks based on the entity type.
PO_PRICE_ELEMENTS	Sourcing Price Element	Sourcing Price Element
PO_PURCHASE_REQUISITION	Purchase Requisition	Purchase Requisition
PO_QUOTE	Sourcing Quote	Sourcing Quote
PO_RECEIPT_CORRECTION	Receipt Correction	Receipt Correction
PO_RECEIPT_TRAVELER	Receipt Traveler	Receipt Traveler
PO_REQUISITION_APPROVAL	Requisition Approval	Requisition Approval
PO_REQ_APPROVAL_HIERARCHY	Requisition Approval Hierarchy	Requisition Approval Hierarchy
PO_RFI	Request for Information	Request for Information
PO_RFQ	Request for Quotation	Request for Quotation
PO_RFQ_RESPONSE	RFQ Response	RFQ Response
PO_SERVICES_RECEIPTS	Services Receipt	Services Receipt
PO_SHIPMENT_AND_BILLING_NOTICE	Shipment / Billing Notice	Shipment / Billing Notice
PO_SOURCING_BID	Sourcing Bid	Sourcing Bid
PO_SOURCING_RULES	Sourcing Rule	Sourcing Rule
PO_SOURCING_RULE_ASSIGNMENTS	Sourcing Rule Assignment	Sourcing Rule Assignment

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
PO_STANDARD_PURCHASE_ORDER	Standard Purchase Order	Standard Purchase Order
PO_SUPPLIER_BANK_ACCOUNT	Supplier Bank Account	Supplier Bank Account
PQH_ADDITIONAL_SECOND_PENSION	Additional Second Pension	Additional Second Pension
PQH_DEFAULT_HR_BUDGET_SET	Default HR Budget Set	Default HR Budget Set
PQH_EMEA_SENIORITY_SITUATION	European Seniority Situation	European Seniority Situation
PQH_EMPLOYEE_ACCOMMODATION	Employee Accommodation	Employee Accommodation
PQH_EMPLOYER_ACCOMMODATION	Employer Provided Accommodation	Employer Provided Accommodation
PQH_FR_CORPS	French CORPS	French CORPS
PQH_FR_SERVICES_VALIDATION	French Services Validation	French Services Validation
PQH_FR_STATUTORY_SITUATION	French Statutory Situation	French Statutory Situation
PQH_GLOBAL_PAY_SCALE	Global Pay Scale	Global Pay Scale
PQH_POS_CTRL_BUSINESS_RULE	Position Control Business Rule	Position Control Business Rule
PQH_POS_CTRL_ROUTING	Position Control Routing	Position Control Routing
PQH_POS_CTRL_TRANSACTION_TEMPLATE	Position Control Transaction Template	Position Control Transaction Template

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
PQH_RBC_RATE_MATRIX	Person Eligibility Criteria Rates Matrix	Rate Matrix stores different criteria value combinations and the rate a person is eligible for if the person's value matches the criteria values.
PQH_REMUNERATION_REGULATION	Remuneration Regulation	Remuneration Regulation
PQH_WORKPLACE_VALIDATION	Workplace Validation Process	Workplace Validation
PQP_PENSION_AND_SAVING_TYPE	Pension and Saving Type	Pension and Saving Type
PQP_VEHICLE_ALLOCATION	Vehicle Allocation	Vehicle Allocation
PQP_VEHICLE_REPOSITORY	Vehicle Repository	Vehicle Repository
PRP_PROPOSAL	Sales Proposal	Sales Proposal
PSP_EFF_REPORT_DETAILS	Employee Effort Report	It summarizes employee's labor distributions over a period of time. It is used to ensure accurate disbursement of labor charges to comply with Office of Management and Budget Guidelines.
PV_OPPORTUNITY	Partner Opportunity Assignment	It supports the assignment of indirect opportunities to partners.
PV_PARTNER_PROFILE	Partner Profiling	It is the extensible attribute model used to capture additional information about a partner and their contacts.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
PV_PROGRAM	Partner Program Management	It represents the partner program management framework which includes the creation/maintenance of partner programs and the associated partner enrollments/memberships into those programs.
PV_REFERRAL	Partner Business Referral	Partner creates referrals to refer business to vendor. If referral results in a sale, the partner gets compensated. Partner register deals with vendor for non-competition purposes.
QA_PLAN	Quality Collection Plan	Quality Collection Plan
QA_RESULT	Quality Result	It indicates collection plan result data collected directly, through transactions or collection import.
QA_SPEC	Quality Specification	A requirement for a characteristic for an item or item category specific to a customer or supplier.
QOT_QUOTE	Sales Quote	Sales Quote
QP_PRICE_FORMULA	Price Formula	Price Formula
QP_PRICE_LIST	Price List	Price List
QP_PRICE_MODIFIER	Price Modifier	Price Modifier
QP_PRICE_QUALIFIER	Price Qualifier	Price Qualifier
REPAIR_ORDER	Repair Order	Repair Order

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
RLM_CUM	Supplier Shipment Accumulation	It is used to track the total shipments made by the supplier for a particular customer item, based on CUM management setup.
RLM_SCHEDULE	Customer Demand Schedule	It refers to customers production material release.
RRS_SITE	Site	It indicates the spatial location of an actual or planned structure or set of structures (as a building, business park, communication tower, highway or monument).
SOA_DIAGNOSTICS	Diagnostics for Oracle E-Business Suite Integrated SOA Gateway	Diagnostics for Oracle E-Business Suite integrated SOA Gateway
UMX_ACCT_REG_REQUESTS	User Account Request	It represents requests made for user accounts, needed to gain system access.
UMX_ROLE	Security Role	It represents a set of permissions in the security system. Roles are assigned to users and can be defined in role inheritance hierarchies. A Responsibility is a special type of role.
UMX_ROLE_REG_REQUESTS	Security Role Request	It represents requests made for roles (as defined in the security system) to gain access to a secured part of the system.
WF_ENGINE	Workflow Item	It indicates a workflow Item including processes, functions, notifications and event activities.

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
WF_EVENT	Business Event	Business Event
WF_NOTIFICATION	Workflow Notification	Workflow Notification
WF_USER	Workflow Directory User	Workflow Directory User
WF_WORKLIST	Workflow Worklist Content	Approve workflow entities (Expense Reports, PO Request, HR Offer, HR Vacancy)
WIP_ACCOUNTING_CLASS	WIP Accounting Class	WIP Accounting Class
WIP_COMPLETION_TRANSACTION	WIP Assembly Completion	Business Entity for Assembly Completion in WIP
WIP_EMPLOYEE_LABOR_RATE	WIP Employee Labor Rate	WIP Employee Labor Rate
WIP_MATERIAL_TRANSACTION	WIP Material Transaction	Business Entity for Material Transaction in WIP
WIP_MOVE_TRANSACTION	WIP Shopfloor Move	WIP Shopfloor Move
WIP_PARAMETER	Work in Process Setup	Work in Process Setup
WIP_PRODUCTION_LINE	Production Line	Production Line
WIP_REPETITIVE_SCHEDULE	Repetitive Schedule	Repetitive Schedule
WIP_RESOURCE_TRANSACTION	WIP Resource Process Flow	WIP Resource Transaction
WIP_SCHEDULE_GROUP	WIP Schedule Group	WIP Schedule Group
WIP_SHOPFLOOR_STATUS	Shopfloor Status	Shopfloor Status
WIP_WORK_ORDER	Work Order	Job/Work Order

BUSINESS_ENTITY_CODE	MEANING	DESCRIPTION
WMS_CONTAINER	Warehouse Management License Plate	Warehouse Container and License Plate Management
WMS_DEVICE_CONFIRMATION_PUB	Dispatch Task	It contains status update for dispatch task. For example, an ASRS task, a Carousel task, a Pick to Light system task.
WMS_DEVICE_INTEGRATION	Warehouse Device	Warehouse Device
WMS_EPC_PUB	Electronic Product Code	It stores Electronic Product Codes such as GTIN, GID, SSCC, etc.
WMS_INSTALL	Warehouse Management System Installation Check	<p>This API has two purposes:</p> <ul style="list-style-type: none"> <li>• This API checks if WMS product is installed in the system, without which some flags are hidden on forms.</li> <li>• The API also returns if an organization is wms enabled.</li> </ul>
WMS_LABEL	Label Printing	It holds information to support the printing of shipping, package, container, item and serial labels.
WMS_LICENSE_PLATE	License Plate	It is an identifier of a container instance used by shipping, warehouse management and shop floor management.
WMS_RFID_DEVICE	Warehouse Management Radio Frequency Identification	Warehouse Management Radio Frequency Identification Integration



<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
WMS_SHIPPING_TRANSACTION	Warehouse Management Shipping Transaction	Warehouse Management truck loading and shipping
WSH_CONTAINER_PUB	Container	Vessel in which goods and material are packed for shipment.
WSH_DELIVERY	Delivery	Group of Shipment Lines
WSH_DELIVERY_LINE	Delivery Line	Shipment Line
WSH_EXCEPTIONS_PUB	Shipping Exception	Exceptions automatically logged for Shipping Entities such as Change Quantity, Cancel Shipment in OM, etc. Exception behavior defined as "Error", "Warning" or "Information Only".
WSH_FREIGHT_COSTS_PUB	Freight Costs	The cost of transportation services for the Shipper. For example, amount Shipper will pay carrier for transportation services.
WSH_PICKING_BATCHES_PUB	Pick Release	The process of releasing delivery lines to warehouse for allocation and picking.
WSH_TRIP	Trip	It describes a planned or historical departure of shipment from a location.
WSH_TRIP_STOPS_PUB	Trip Stop	The physical location through which a Trip will pass where goods are either dropped off or picked up.
WSM_INV_LOT_TXN	Inventory Lot Transaction	Lot based Inventory Transactions
WSM_LOT_BASED_JOB	Lot Based Job	Lot Based Job / WIP Lot

<b>BUSINESS_ENTITY_CODE</b>	<b>MEANING</b>	<b>DESCRIPTION</b>
WSM_LOT_MOVE_TXN	Lot Move Transaction	Lot based jobs shopfloor move transactions
WSM_WIP_LOT_TXN	WIP Lot Transaction	Lot based WIP transactions
XDP_SERVICE_ORDER	Service Fulfillment Order	An order for one or more services, which need to be provisioned by Service Fulfillment Manager. The provisioning of these services often involve systems outside Oracle E-Business Suite.
XLA_JOURNAL_ENTRY	Subledger Accounting Journal Entry	Subledger Accounting Journal Entry comprising of a Header, Line and Distribution
XNB_ADD_BILLSUMMARY	Bill Summary Processing	This is used for inserting, creating, or populating new Bill Summary records into Oracle E-Business Suite from external Billing systems.
XNB_ADD_GROUPSALERO RDER	Billing System Sales Order Lines Group	All the Sales order lines information is generated as one XML Message and published to third party billing application.
XNB_ADD_SALESORDER	Billing System Sales Order Addition	Sales order information is generated as XML Message and published to third party billing application.
XNB_SYNC_ACCOUNT	Billing System Customer Account Synchronization	An account information is generated as XML Message and published to third party billing application.
XNB_SYNC_ITEM	Billing System Inventory Item Synchronization	Catalog information is generated as XML Message and published to third party billing application.

BUSINESS_ENTITY_CODE	MEANING	DESCRIPTION
XTR_BANK_BALANCE	Bank Account Balance	Bank Account Balance
XTR_DEAL_DATA	Treasury Deal	Treasury Deal
XTR_MARKET_DATA	Market Rate	Financial Market Rates Data
XTR_PAYMENT	XTR Payment	Treasury Payment represents the payments that are being made.
ZX_DATA_UPLOAD	Imported Tax Content	This entity code is used in all the programs of Oracle E-Business Suite Tax Content Upload Request Set.

#### Example: Create Customer

```

/*#
_ *This interface creates a customer. It calls the
_ *customer hub API that creates a 'party' to create a
_ *party of type 'customer'.
_ *@rep:scope public
_ *@rep:product OM
_ *@rep:displayname Create Customer
_ *@rep:category BUSINESS_ENTITY OM_CUSTOMER
_ *@rep:lifecycle active
_ *@rep:compatibility S
_ */

```

## Composite Service - BPEL Annotation Guidelines

This section describes what you should know about Integration Repository annotations for Composite Services - BPEL.

#### Annotating Composite Services - BPEL

- You should annotate BPEL projects in \*.bpel files.
- Before annotating, make sure that no comments beginning with /\*# are present. The "slash-star-pound" characters are used to set off repository annotations, and will result in either an error or undesirable behavior if used with normal comments.
- To annotate, open the .bpel file in text editor to edit the file.
- In the .bpel file, place the annotations within the comments section in beginning of the file.

- Enter meaningful description that covers the condition under which the business event is raised and the UI action that invokes the business event.
- Define product codes in FND\_APPLICATION.
- Use existing business entities for your composite services - BPEL processes. For the list of existing business entities, see Business Entity Annotation Guidelines, page A-37.
- Interface name in <BPEL\_PROCESS\_NAME>.bpel should be defined as 'oracle.apps' + product\_code + '<BPEL\_PROCESS\_NAME>.
- If BPEL process name is "BPEL\_PROCESS\_NAME", then
  - A BPEL Process Jar file should be created with <prod>\_pbel\_<BPEL\_PROCESS\_NAME>.jar.
  - <prod>\_pbel\_<BPEL\_PROCESS\_NAME>.jar file should be placed under \$product\_top/patch/115/jar/bpel.
  - <prod>\_pbel\_<BPEL\_PROCESS\_NAME>.jar file should be unzipped under \$product\_top/patch/115/jar/bpel.
  - BPEL file for <BPEL\_PROCESS\_NAME>.jar should be present under \$product\_top/patch/115/jar/bpel/<prod>\_pbel\_<BPEL\_PROCESS\_NAME>.bpel.
  - BPEL File Name should not be changed from <BPEL\_PROCESS\_NAME>.bpel.
  - WSDL file for <BPEL\_PROCESS\_NAME> should be present under \$product\_top/patch/115/jar/bpel/<prod>\_pbel\_<BPEL\_PROCESS\_NAME>.bpel.
  - WSDL File Name should not be changed from <BPEL\_PROCESS\_NAME>.wsdl.
  - Standalone Parser should be run on annotated \$product\_top/patch/115/jar/bpel/<prod>\_pbel\_<BPEL\_PROCESS\_NAME>.bpel/bpel<BPEL\_PROCESS\_NAME>.bpel.
  - \$product\_top/patch/115/jar/bpel/<prod>\_pbel\_<BPEL\_PROCESS\_NAME>.bpel/bpel<BPEL\_PROCESS\_NAME>\_bpel.ildt should be loaded into Integration Repository.
- During the execution of a standalone parser, arcs file location of \*.bpel file should be patch/115/jar/bpel.

#### Annotations for Composite Services - BPEL - Syntax

The annotations for composite services - BPEL are:

```

/*#
 * This is a bpel file for creating invoice.
 * @rep:scope public
 * @rep:displayname Create Invoice
 * @rep:lifecycle active
 * @rep:product inv
 * @rep:compatibility S
 * @rep:interface oracle.apps.inv.CreateInvoice
 * @rep:category BUSINESS_ENTITY INVOICE_CREATION
 */

```

Refer to General Guidelines for Annotations, page A-1 in Integration Repository for details of element definitions.

### Required Annotations

Follow the links below to view syntax and usage of each annotation.

- Must begin with description sentence(s)
- rep:displayname, page A-117
- rep:scope, page A-115
- rep:product, page A-116
- rep:category BUSINESS\_ENTITY, page A-125

### Optional Annotations

- link, page A-120
- see, page A-121
- rep:lifecycle, page A-119
- rep:compatibility, page A-119
- rep:ihelp, page A-122
- rep:metalink, page A-123
- rep:doccd, page A-124

### Template

You can use the following template when annotating composite - BPEL files:

```

.
.
.
/*#
 * <Put your long bpel process description here
 * it can span multiple lines>
 * @rep:scope <scope>
 * @rep:displayname <display name>
 * @rep:lifecycle <lifecycle>
 * @rep:product <product or pseudoproduct short code>
 * @rep:compatibility <compatibility code>
 * @rep:interface <oracle.apps.[product_code].[bpel_process_name]>
 * @rep:category BUSINESS_ENTITY <entity name>
 */
.
.
.

```

### Example

Here is an example of an annotated composite - BPEL file:

```

////////////////////////////////////
Oracle JDeveloper BPEL Designer

Created: Tue Oct 30 17:10:13 IST 2007
Author:  jdole
Purpose: Synchronous BPEL Process
/*#
 * This is a bpel file for creating invoice.
 * @rep:scope public
 * @rep:displayname Create Invoice
 * @rep:lifecycle active
 * @rep:product PO
 * @rep:compatibility S
 * @rep:interface oracle.apps.po.CreateInvoice
 * @rep:category BUSINESS_ENTITY INVOICE
 */

////////////////////////////////////

-->
<process name="CreateInvoice">
  targetNamespace="http://xmlns.oracle.com/CreateInvoice"
  xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"

  xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"

  xmlns:xp20="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.Xpath20"

  xmlns:ns4="http://xmlns.oracle.com/pcbpel/adapter/file/ReadPayload/"
  xmlns:ldap="http://schemas.oracle.com/xpath/extension/ldap"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ns5="http://xmlns.oracle.com/bpel/workflow/xpath"
  xmlns:client="http://xmlns.oracle.com/CreateInvoice"

  xmlns:ns6="http://xmlns.oracle.com/bpel/services/IdentityService/xpath"
  xmlns:ora="http://schemas.oracle.com/xpath/extension"

  xmlns:ns1="http://xmlns.oracle.com/soapprovider/plsql/AR_INVOICE_API_PUB_
2108/CREATE_SINGLE_INVOICE_1037895/"

  xmlns:ns3="http://xmlns.oracle.com/soapprovider/plsql/AR_INVOICE_API_PUB_
2108/APPS/BPEL_CREATE_SINGLE_INVOICE_1037895/AR_INVOICE_API_PUB-24CREATE
_INV/"
  xmlns:ns2="http://xmlns.oracle.com/pcbpel/adapter/appscontext/"
  xmlns:bpelx="http://schemas.oracle.com/bpel/extension"

  xmlns:orcl="http://www.oracle.com/XSL/Transform/java/oracle.tip.pc.servi
ces.functions.ExtFunc">

  <!--
  //////////////////////////////////
  PARTNERLINKS
    List of services participating in this BPEL process
  //////////////////////////////////
  -->
  <partnerLinks>
    <!--
    The 'client' role represents the requester of this service. It is
    used for callback. The location and correlation information
    associated

```

```

with the client role are automatically set using WS-Addressing.
-->
<partnerLink name="client" partnerLinkType="client:CreateInvoice"
    myRole="CreateInvoiceProvider"/>
<partnerLink name="CREATE_SINGLE_INVOICE_1037895"
    partnerRole="CREATE_SINGLE_INVOICE_1037895_ptt_Role"

partnerLinkType="ns1:CREATE_SINGLE_INVOICE_1037895_ptt_PL"/>
<partnerLink name="ReadPayload" partnerRole="SynchRead_role"
    partnerLinkType="ns4:SynchRead_plt"/>
</partnerLinks>
<!--
////////////////////////////////////
VARIABLES
    List of messages and XML documents used within this BPEL process
////////////////////////////////////
-->
<variables>
<!--Reference to the message passed as input during initiation-->
    <variable name="inputVariable"
        messageType="client:CreateInvoiceRequestMessage"/>
<!--Reference to the message that will be returned to the requester-->
    <variable name="outputVariable"
        messageType="client:CreateInvoiceResponseMessage"/>
    <variable
name="Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
        messageType="ns1:Request"/>
    <variable
name="Invoke_1_CREATE_SINGLE_INVOICE_1037895_OutputVariable"
        messageType="ns1:Response"/>
    <variable name="Invoke_2_SynchRead_InputVariable"
        messageType="ns4:Empty_msg"/>
    <variable name="Invoke_2_SynchRead_OutputVariable"
        messageType="ns4:InputParameters_msg"/>
</variables>
<!--
////////////////////////////////////
ORCHESTRATION LOGIC
    Set of activities coordinating the flow of messages across the
    services integrated within this business process
////////////////////////////////////
-->
<sequence name="main">
    <!--Receive input from requestor. (Note: This maps to operation
defined in CreateInvoice.wsdl)-->
    <receive name="receiveInput" partnerLink="client"
        portType="client:CreateInvoice" operation="process"
        variable="inputVariable" createInstance="yes"/>
    <!--Generate reply to synchronous request-->
    <assign name="SetHeader">
        <copy>
            <from expression="'operations'">
            <to
variable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
                part="header"

query="/ns1:SOAHeader/ns2:ProcedureHeaderType/ns2:Username"/>
        </copy>
        <copy>
            <from expression="'Receivables, Vision Operations (USA)'">
            <to

```



```

variable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
    part="header"

query="/ns1:SOAHeader/ns2:ProcedureHeaderType/ns2:Responsibility"/>
    </copy>
    <copy>
        <from expression="'204'">
        <to
variable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
    part="header"
    query="/ns1:SOAHeader/ns2:ProcedureHeaderType/ns2:ORG_ID"/>
    </copy>
    <copy>
        <from expression="'Receivables, Vision Operations (USA)'">
        <to
variable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
    part="header"

query="/ns1:SOAHeader/ns1:SecurityHeader/ns1:ResponsibilityName"/>
    </copy>
    </assign>
    <invoke name="InvokeReadPayload" partnerLink="ReadPayload"
        portType="ns4:SynchRead_ptt" operation="SynchRead"
        inputVariable="Invoke_2_SynchRead_InputVariable"
        outputVariable="Invoke_2_SynchRead_OutputVariable"/>
    <assign name="SetPayload">
        <copy>
            <from variable="Invoke_2_SynchRead_OutputVariable"
                part="InputParameters" query="/ns3:InputParameters"/>
            Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
            part="body" query="/ns1:SOARequest/ns3:InputParameters"/>
        </copy>
    </assign>
    <assign name="SetDate">
        <copy>
            <from expression="xp20:current-date()">
            <to to
variable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"
    part="body"

query="/ns1:SOARequest/ns3:InputParameters/ns3:P_TRX_HEADER_TBL/ns3:P_TR
X_HEADER_TBL_ITEM/ns3:TRX_DATE"/>
    </copy>
    </assign>
    <invoke name="Invoke_1" partnerLink="CREATE_SINGLE_INVOICE_1037895"
        portType="ns1:CREATE_SINGLE_INVOICE_1037895_ptt"
        operation="CREATE_SINGLE_INVOICE_1037895"

inputVariable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_InputVariable"

outputVariable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_OutputVariable"/>
    <assign name="AssignResult">
        <copy>
            <from
variable="Invoke_1_CREATE_SINGLE_INVOICE_1037895_OutputVariable"
    part="body"

query="/ns1:SOAResponse/ns3:OutputParameters/ns3:X_MSG_DATA"/>
    <to variable="outputVariable" part="payload"
        query="/client:CreateInvoiceProcessResponse/client:result"/>
    </copy>

```

```

</assign>
  <reply name="replyOutput" partnerLink="client"
    portType="client:CreateInvoice" operation="process"
    variable="outputVariable"/>
</sequence>
</process>

```

## Glossary of Annotations

The following is a list of currently supported annotation types and details about their recommended use.

**<description sentence(s)>**

Annotation Type	<description sentence(s)>
Syntax	Does not require a tag.

Annotation Type	<description sentence(s)>
Usage	<p data-bbox="971 306 1463 369">Defines a user-friendly description of what the interface or method does.</p> <p data-bbox="971 394 1463 520">Start the description with a summary sentence that begins with a capital letter and ends with a period. Do not use all capitals) and do not capitalize words that are not proper nouns.</p> <p data-bbox="971 546 1414 606">An example of a good beginning sentence could be as follows:</p> <p data-bbox="971 632 1425 724">"The Purchase Order Data Object holds the purchase order data including nested data objects such as lines and shipments."</p> <p data-bbox="971 749 1451 1003">In general, a good description has multiple sentences and would be easily understood by a potential customer. An exception to the multiple sentence rule is cases where the package-level description provides detailed context information and the associated method-level descriptions can therefore be more brief (to avoid repetitiveness).</p> <p data-bbox="971 1029 1435 1056">A bad example would be: "Create an order."</p> <p data-bbox="971 1081 1455 1142">This description is barely usable. A better one would be:</p> <p data-bbox="971 1167 1446 1228">"Use this package to create a customer order, specifying header and line information."</p> <p data-bbox="971 1253 1463 1346">You can use the &lt;br&gt; tag for forcing a new line in description. The following is an example on how to force a new line in the description:</p> <p data-bbox="971 1371 1463 1432">The following is an example on how to force a new line in the description:</p> <p data-bbox="971 1457 1463 1803">FEM_BUDGETS_ATTR_T is an interface table for loading and updating Budget attribute assignments using the Dimension Member Loader. &lt;br&gt; These attribute assignments are properties that further describe each Budget. &lt;br&gt; When loading Budgets using the Dimension Member Loader, identify each new member in the FEM_BUDGETS_B_T table while providing an assignment row for each required attribute in the FEM_BUDGETS_ATTR_T table.</p>

Annotation Type	<description sentence(s)>
Example	<pre> /*#  * This is a sample description. Use  * standard English capitalization and  * punctuation. Write descriptions  * carefully. </pre>
Required	Required for all interfaces that have @rep:scope public.
Default	If not set, the value is defaulted from the Javadoc or PL/SQL Doc of the interface or method.
Level	Interface (class) and API (method).
Multiple Allowed	No. Use only one per each program element (class or method).
Data Model	<p>The full description text goes into:</p> <pre> FND_OBJECTS.DESCRPTION FND_FORM_FUNCTIONS.DESCRPTION </pre> <p>The first sentence of the description text goes into:</p> <pre> FND_OBJECTS_TL.DESCRPTION FND_FORM_FUNCTIONS_TL.DESCRPTION </pre>
Comments	<p>Optionally, you can use the following HTML tags in your descriptions:</p> <pre> &lt;body&gt; &lt;p&gt; &lt;strong&gt; &lt;em&gt; &lt;ul&gt; &lt;li&gt; &lt;h1&gt; &lt;h2&gt; &lt;h3&gt; </pre> <p>&lt;pre&gt; for multiple code samples (should be enclosed by &lt;code&gt; tags)</p>

## @rep:scope

Annotation Type	@rep:scope
Syntax	<code>@rep:scope public   private   internal</code>
Usage	Indicates where to publish the interface, if at all.
Example	<p><code>@rep:scope public</code> means publish everywhere.</p> <p><b>Note:</b> Public interfaces are displayed on the customer-facing UI.</p> <p><code>@rep:scope private</code> means that this interface is published to the Integration Repository but restricted for use by the owning team.</p> <p><code>@rep:scope internal</code> means publish within the company.</p>
Required	Required for all interfaces.
Default	None.
Level	Interface (class) and API (method).
Multiple Allowed	No. Use only one per each program element (class or method).
Data Model	<code>FND_OBJECTS.SCOPE</code> <code>FND_FORM_FUNCTIONS.SCOPE</code>
Comments	Private and internal interfaces won't appear on the customer-facing UI, but will appear on the Oracle-internal UI.

### **@rep:product**

Annotation Type	@rep:product
Syntax	@rep:product <i>StringShortCode</i>
Usage	Specifies the product shortname of the interface, as defined in the ARU system.
Example	@rep:product PO
Required	Required for all interfaces.
Default	None.
Level	Interface (class) only.
Multiple Allowed	No. Use only one per interface.
Data Model	FND_OBJECTS.PRODUCT
Comments	<p>If the interface belongs to a pseudoproduct, then use the pseudoproduct's shortname.</p> <p>If your team has an abbreviated name that is different from the application shortname, then use it instead of the application shortname. For example, use "GL" instead of "SQLGL".</p>

### **@rep:implementation**

Annotation Type	@rep:implementation
Syntax	@rep:implementation <i>StringClassName</i>
Usage	Specifies the implementation class name of the interface.
Example	@rep:implementation oracle.apps.po.server.PurchaseOrdersAmImpl

Annotation Type	@rep:implementation
Required	Required for Java only.
Default	None.
Level	Interface (class).
Multiple Allowed	No. Use only one per interface.
Data Model	FND_OBJECTS.IMPLEMENTATION_NAME
Comments	

#### **@rep:displayname**

Annotation Type	@rep:displayname
Syntax	@rep:displayname <i>StringName</i>
Usage	Defines a user-friendly name for the interface.
Example	@rep:displayname Purchase Order Summary
Required	Required for all interfaces that have @rep:scope public.
Default	None.
Level	Interface (class) and API (method).
Multiple Allowed	No. Use only one per each program element (class or method).
Data Model	FND_OBJECTS_TL.DISPLAY_NAME  FND_FORM_FUNCTIONS_TL.USER_FUNCTION NAME

Annotation Type	@rep:displayname
Comments	<p data-bbox="873 306 1159 338"><b>Display Name Guidelines</b></p> <p data-bbox="873 363 1349 457">These guidelines apply to display names for all technologies (interfaces, classes, methods, parameters, XMLG maps, and so on).</p> <p data-bbox="873 480 1295 541">Display names must meet the following criteria:</p> <ul data-bbox="878 564 1365 1339" style="list-style-type: none"> <li data-bbox="878 564 1344 632">• Be mixed case. Do not use all capitals or all lower case.</li> <li data-bbox="878 667 1300 762">• Be singular rather than plural. For example, use "Customer" instead of "Customers".</li> <li data-bbox="878 798 1336 865">• Be fully qualified and representative of your business area.</li> <li data-bbox="878 900 1192 932">• Not have underscores (_).</li> <li data-bbox="878 968 1192 999">• Not end with a period (.).</li> <li data-bbox="878 1035 1317 1066">• Not be the same as the internal name.</li> <li data-bbox="878 1102 1365 1169">• Not begin with a product code or product name.</li> <li data-bbox="878 1205 1365 1339">• Not contain obvious redundancies such as "Package", "API", or "APIs". As you write your display names, do consider the UI where the display name will be seen.</li> </ul> <p data-bbox="873 1375 1333 1533">For example, use 'Promise Activity' as the display name, instead of IEX_PROMISES_PUB. The reason is that IEX_PROMISES_PUB contains underscores and is the same as the internal name.</p> <p data-bbox="873 1560 1357 1682">Use 'Process Activity' as the display name, instead of 'Workflow Process Activity APIs'. This is because it begins with a product name and ends with "APIs".</p>



### **@rep:lifecycle**

Annotation Type	@rep:lifecycle
Syntax	@rep:lifecycle active   deprecated   obsolete   planned
Usage	Indicates the lifecycle phase of the interface.
Example	<p>@rep:lifecycle active means the interface is active.</p> <p>@rep:lifecycle deprecated means the interface has been deprecated.</p> <p>@rep:lifecycle obsolete means the interface is obsolete and must not be used.</p> <p>@rep:lifecycle planned means the interface is planned for a future release. This is used for prototypes and mockups.</p>
Required	Optional.
Default	The default value is active.
Level	Interface (class) and API (method).
Multiple Allowed	No. Use only one per each program element (class or method).
Data Model	FND_OBJECTS.LIFECYCLE FND_FORM_FUNCTION.LIFECYCLE
Comments	The parsers will validate that this annotation is in sync with the "@deprecated" Javadoc annotation.

### **@rep:compatibility**

Annotation Type	@rep:compatibility
Syntax	@rep:compatibility S   N

Annotation Type	@rep:compatibility
Usage	<p>S indicates the lifecycle phase of the interface.</p> <p>N indicates that backward compatibility is not assured.</p>
Example	@rep:compatibility S
Required	Optional.
Default	Conditional. The value is defaulted to S for @rep:scope public. Otherwise, the value is defaulted to N.
Level	Interface (class) and API (method).
Multiple Allowed	No. Use only one per each program element (class or method).
Data Model	<p>FND_OBJECTS.COMPATIBILITY</p> <p>FND_FORM_FUNCTION.COMPATIBILITY</p>
Comments	<p>Integration Repository UI only flags for @rep:compatibility N.</p> <p>@rep:compabitility S is not shown or highlighted in the UI.</p>

## @link

Annotation Type	@link
	<p><b>Note: This is supported only for a destination of Java.</b></p>
Syntax	{@link package.class#member label}
Usage	Provides a link to another interface or method.
Example	<pre>{@link #setAmounts(int,int,int,int) Set Amounts}</pre>

Annotation Type	@link
	<b>Note: This is supported only for a destination of Java.</b>
Required	Optional.
Default	None.
Level	Interface (class) and API (method).
Multiple Allowed	Yes.
Data Model	N/A
Comments	<p>This is the standard Javadoc "@link" annotation, where the linked items are embedded as hyperlinks in the description that displays in the UI.</p> <p>Take note of the following rules: Public APIs must not link to private or internal APIs. @link annotations must not link to documents that are not accessible by the Integration Repository viewer.</p>

## @see

Annotation Type	@see
Syntax	@see StringLocator
Usage	Provides a link to another interface or method.
Example	@see #setAmounts(int,int,int,int)
Required	Optional.
Default	None.
Level	Interface (class) and API (method).

Annotation Type	@see
Multiple Allowed	Yes.
Data Model	FND_CHILD_ANNOTATIONS.VALUE
Comments	<p>This is the standard Javadoc "@see" annotation.</p> <p>The linked items will display on the UI under a "See Also" heading.</p> <p><b>Usage in PL/SQL Code: @see package#procedure</b></p>
<b>@rep:ihelp</b>	
Annotation Type	@rep:ihelp
Syntax	<p>When used as a separate child annotation on a single line:</p> <pre>@rep:ihelp &lt;product_shortname&gt;/@&lt;help_target&gt; #&lt;help_target&gt; &lt;link_text&gt;</pre> <p>When used as an inline annotation, add curly braces:</p> <pre>{@rep:ihelp &lt;product_shortname&gt;/@&lt;help_target&gt; #&lt;help_target&gt; &lt;link_text&gt;}</pre>
Usage	<p>Provides a link to an existing Oracle E-Business Suite HTML online help page.</p> <p>product_shortname is the product short name.</p> <p>help_target is the help target that was manually embedded in the file by the technical writer, such as, "jtfaccsum_jsp," "aolpo," "overview," "ast_aboutcollateral".</p>
Example	@see #setAmounts(int,int,int,int)
Required	Optional.

Annotation Type	@rep:ihelp
Default	None.
Level	Interface (class) and API (method).
Multiple Allowed	Yes.
Data Model	
Comments	The @rep:ihelp annotation is slightly preferred over the similar @rep:doccd annotation, as it is both local and more specific.

#### **@rep:metalink**

Annotation Type	@rep:metalink
Syntax	<p>When used as a separate child annotation on a single line:</p> <pre>@rep:metalink &lt;bulletin_number&gt; &lt;link_text&gt;</pre> <p>When used as an inline annotation, add curly braces:</p> <pre>{@rep:metalink &lt;bulletin_number&gt; &lt;link_text&gt;}</pre>
Usage	Provides a link to an existing My Oracle Support (formerly Oracle <i>MetaLink</i> ) Knowledge Document.
Example	<pre>@rep:metalink 123456.1 See My Oracle Support Knowledge Document 123456.1</pre>
Required	Optional.
Default	None.
Level	Interface (class) and API (method).

Annotation Type	@rep:metalink
Multiple Allowed	Yes.
Data Model	
Comments	

#### **@rep:doccd**

Annotation Type	@rep:doccd
Syntax	<p>When used as a separate child annotation on a single line:</p> <pre>@rep:doccd &lt;file_name&gt; &lt;link_text&gt;</pre> <p>When used as an inline annotation, add curly braces:</p> <pre>{@rep:doccd &lt;file_name&gt; &lt;link_text&gt;}</pre>
Usage	Forms a link to a PDF guide on the documentation CD.
Example	<pre>{@rep:doccd 115xyzug.pdf See the Oracle Sample Product Users Guide}</pre>
Required	Optional.
Default	None.
Level	Interface (class) and API (method).
Multiple Allowed	Yes.
Data Model	

Annotation Type	@rep:doccd
Comments	<ul style="list-style-type: none"> <li>PDF guides on the doc CD should follow the naming standard for PDF file names. Regardless of what the file name is, once an @rep:doccd annotation exists in source code, teams should make every effort to keep that file name stable and enduring.</li> </ul> <p>If the PDF has been zipped to reduce its file size, then use @rep:doccd &lt;file_name.zip&gt; &lt;link_text&gt;.</p> <ul style="list-style-type: none"> <li>The @rep:ihelp annotation is slightly preferred over the similar @rep:doccd annotation, as it is both local and more specific.</li> </ul>

### @rep:category

Annotation Type	@rep:category
Syntax	<pre>@rep:category StringLookupType StringLookupCode SequenceNumber</pre> <p>The <i>StringLookupCode</i> is what you write in your source file. This is a short form. It cannot have any blank spaces within it also it has to be in UPPERCASE only (Numbers are allowed). Meaning a look up code like "ABC D" or "Abcd" is wrong. It has to be something like the following: "ABCD" or "ABC_D" or "ABC2". The Look up code has to be registered with the librarian along with the actual display name. During run time iRep UI resolves the Look up code against the library table and paints the display name in the UI</p> <p>See Annotation Syntax, page A-1 for details about this annotation's syntax.</p>
Usage	Specifies the business category of the interface.

Annotation Type	@rep:category
Example	<pre>@rep:category BUSINESS_ENTITY PO_PURCHASE_ORDER</pre> <p>PO_PURCHASE_ORDER is your string lookup code and your display name for example could be "Purchase Order".</p> <p>In this case register PO_PURCHASE_ORDER and its desired displayed name with the librarian.</p> <pre>@rep:category STANDARD_READY 3A4_IN</pre> <p>Where 3A4_IN IS YOUR Look up code and it resolves to 'ROSETTANET:02.02.00:PIP3A4-Request Purchase Order' when displayed through UI. Both 3A4_IN and its display name (ROSETTANET:02.02.00:PIP3A4-Request Purchase Order) have to be registered with the Librarian. Follow a similar process for the rest of the category types listed below.</p> <pre>@rep:category OPEN_INTERFACE AP_INVOICE_LINES_INTERFACE 1</pre> <p>Where AP_INVOICE_LINES_INTERFACE is your string lookup code.</p> <pre>@rep:category MISC_EXTENSIONS &lt;FND_CODE&gt;</pre>
Required	<p>BUSINESS_ENTITY is mandatory for all interfaces. If the methods belonging to a class ALL have the same business entity, you only need to annotate the class. However, if the methods belonging to a class have heterogeneous business entities, then you have to annotate each of the methods appropriately.</p> <p>See Business Entity Annotation Guidelines, page A-37 for additional details.</p> <p>OPEN_INTERFACE is mandatory for CPs that are part of Open Interfaces.</p>
Default	Methods default to the value set on the class.
Level	Interface (class) and API (method).
Multiple Allowed	Yes.
Data Model	<pre>FND_LOOKUP_ASSIGNMENTS.LOOKUP_TYPE FND_LOOKUP_ASSIGNMENTS.LOOKUP_CODE</pre>



Annotation Type	@rep:category
Comments	<p>You are encouraged to use the <code>rep:category</code> annotation liberally in your code. While certain categories are required (notably <code>BUSINESS_ENTITY</code> and <code>OPEN_INTERFACE</code>), You are encouraged to create your own categories (consult the repository librarian) and utilize this annotation to track product-specific information.</p> <p><code>STANDARD_READY</code> specifies where Oracle is standard-ready but cannot claim standard compliance because a third-party process is responsible for compliance.</p> <p>When using <code>MISC_EXTENSIONS</code>, the value for <code>&lt;FND_CODE&gt;</code> can be any of the following:</p> <ul style="list-style-type: none"> <li>• <code>ALL_USER_EXIT</code></li> <li>• <code>HR_USER_HOOKS</code></li> <li>• <code>HR_DATAPUMP</code></li> <li>• <code>PA_CLIENT_EXTENSION</code></li> </ul> <p>The Stringlookup code cannot be more than 30 characters in length. For example the String lookup value like "ROSETTANET:02.02.00:PIP3A7" cannot be more than 30 characters in length. The development teams have to register both developer key (<code>lookup_code</code>) and display name (meaning) with the librarian similar to the open interface process and then use the developer key inside the annotation.</p>

#### **@rep:usestable**

Annotation Type	@rep:usestable
Syntax	<pre>@rep:usestable &lt;table or view name&gt; &lt;sequence&gt; &lt;direction flag&gt;</pre>

Annotation Type	@rep:usestable
Usage	<p>Used when annotating concurrent programs to identify associated open interface tables or views.</p> <p>&lt;table or view name&gt; is the name of the table or view.</p> <p>&lt;sequence&gt; is an integer used to tell the UI the display order of the different pieces. By convention, in the rep:category OPEN_INTERFACE, page A-125 annotation, you will have used 1 for the concurrent program. Here in the rep:usestable annotations, order the input tables: list master (header) tables before detail (lines) tables. Finally, put any output views or tables at the end of the sequence.</p> <p>&lt;direction flag&gt; is optional and specifies one of the following: IN (default), OUT, or BOTH.</p>
Example	@rep:usestable SampleTable 3 IN
Required	Only if the concurrent program is part of an open interface.
Default	None.
Level	Interface.
Multiple Allowed	Yes.

#### **@rep:standard**

Annotation Type	@rep:standard
Syntax	<p><i>@rep:standard StringType StringVersionNumber StringSpecName</i></p> <p>In the following example @rep:standard OAG 7.2 Process_PO_001 StringType is OAG, StringVersionNumber is 7.2 and StringSpecName is Process_PO_001</p> <p>See Annotation Syntax, page A-1 for details about this annotation's syntax.</p>

Annotation Type	@rep:standard
Usage	Specifies the business standard name. This annotation is reserved for where Oracle is compliant with industry standards.
Example	In the example <code>@rep:standard RosettaNet 02.02.00 'Pip3B12-Shipping Order Confirmation'</code> , the <code>StringSpecName</code> is enclosed in Single Quotes because the spec name has empty spaces. It is not necessary to have these quotes if the <code>StringSpecName</code> does not have any empty spaces like the following example <code>@rep:standard RosettaNet 02.02.00 Pip3B12-PurchaseOrderConfirmation</code> .
Required	Optional.
Default	Methods default to the value set on the class.
Level	Documents and data rows.
Multiple Allowed	No. Use only one per each program element (class or method).
Data Model	<code>FND_OBJECTS.STANDARD</code> <code>FND_OBJECTS.STANDARD_VERSION</code> <code>FND_OBJECTS.STANDARD_SPEC</code>
Comments	

### **@rep:interface**

Annotation Type	@rep:interface
Syntax	<code>@rep:interface StringClassName</code> where the <code>StringClassName</code> syntax is <code>transactiontype:subtype</code> . Refer to the example below.
Usage	Specifies the interface name for technologies where parsing tools can't easily introspect the interface name.

Annotation Type	<b>@rep:interface</b>
Example	The StringClassName is always transactiontype:subtype  @rep:interface PO:POC
Required	Optional.
Default	None.
Level	Interface only.
Multiple Allowed	No. Use only one per interface.
Data Model	FND_OBJECTS.OBJECT_NAME
Comments	Used in technologies where there isn't a strong native definition of the interface, such as XML Gateway and EDI.

#### **@param**

Annotation Type	<b>@param</b>
Syntax	@param paramName paramDescription
Usage	Specifies the name and description of a method, procedure, or function parameter (IN, OUT, or both).
Example	@param PONumber The purchase order number.
Required	Optional.
Default	None.
Level	Methods, procedures, and functions.
Multiple Allowed	Yes.
Data Model	FND_PARAMETERS.PARAMETER_NAME FND_PARAMETERS.PARAMETER_DESCRIPTION

Annotation Type	@param
Comments	For convenience, Java annotations are also supported.

### @return

Annotation Type	@return
Syntax	<code>@return StringDescription</code>
Usage	Specifies the description of a method or function return parameter.
Example	<code>@return The purchase order status.</code>
Required	Optional.
Default	None.
Level	Methods, procedures, and functions.
Multiple Allowed	Yes.
Data Model	<code>FND_PARAMETERS.PARAMETER_NAME</code> <code>FND_PARAMETERS.PARAMETER_DESCRIPTION</code>
Comments	For convenience, Java annotations are also supported.

### @rep:paraminfo

Annotation Type	@rep:paraminfo
Syntax	<code>@rep:paraminfo {@rep:innertype typeName}</code> <code>{@rep:precision value} {@rep:required}</code>

Annotation Type	<b>@rep:paraminfo</b>
Usage	<pre data-bbox="724 306 911 333">rep:paraminfo</pre> <p data-bbox="724 359 1365 453">The rep:paraminfo annotation must come immediately in the line following the parameter's @param or @return annotation it is describing.</p> <pre data-bbox="724 474 911 501">rep:innertype</pre> <p data-bbox="724 527 1365 585">Optional inline annotation to describe the inner type of generic objects such as collections.</p> <pre data-bbox="724 606 911 634">rep:precision</pre> <p data-bbox="724 659 1365 718">Optional inline annotation to specify the parameter precision. Used for Strings and numbers.</p> <pre data-bbox="724 739 894 766">rep:required</pre> <p data-bbox="724 791 1365 852">Optional inline annotation to indicate that a not null must be supplied. This is only needed for non-PL/SQL technologies.</p>

Annotation Type	@rep:paraminfo
Example	<pre> /**  * Acknowledges purchase orders, including whether  * the terms have been accepted or not. You can  * also provide updated line item pricing and  * shipment promise dates with the acknowledgement.  *  * @param purchaseOrders list of purchase order objects  * @paraminfo {@rep:innertype oracle.apps.po.PurchaseOrderAcknowledgements SDO} {@rep:required}  *  * @rep:scope public  * @rep:displayname Receive Purchase Order Items  */ public void acknowledgePurchaseOrders(DataList purchaseOrders);  /**  * Gets the price for a purchase order line item.  *  * @param poNumber purchase order unique identifier  * @paraminfo {@rep:precision 10} {@rep:required}  * @param lineNumber purchase order line unique identifier  * @paraminfo {@rep:precision 10} {@rep:required}  * @return the item price for the given purchase order line  * @paraminfo {@rep:precision 10}  *  * @rep:scope public  * @rep:displayname Get Purchase Order Line Item Price  */ public Number getItemPrice(Number poNumber, Number lineNumber); </pre>
Required	Optional.
Default	None.

Annotation Type	@rep:paraminfo
Level	Methods only.
Multiple Allowed	Yes. Multiple values can be assigned for different parameters.
Data Model	FND_OBJECT_TYPE_MEMBERS.TYPE
Comments	

#### **@rep:businesssevent**

Annotation Type	@rep:businesssevent
Syntax	@rep:businesssevent BusinessEvent
Usage	Indicates the name of the business event raised by this method.
Example	@rep:businesssevent oracle.apps.wf.notification.send
Required	Optional.
Default	Defaulted in file types where the business event can be derived.
Level	Methods only.
Multiple Allowed	Yes.
Data Model	FND_CHILD_ANNOTATIONS.VALUE
Comments	Make sure to use this annotation at every instance where you raise a business event. Note that business events themselves do not require an annotation.



### **@rep:direction**

Annotation Type	@rep:direction
Syntax	@rep:direction <OUT   IN>
Usage	Indicates whether the interface is outbound or inbound.
Example	@rep:direction OUT
Required	Required for EDI and XML Gateway annotations only.
Default	None.
Level	Interface.
Multiple Allowed	No.
Data Model	
Comments	

### **@rep:service**

Annotation Type	@rep:service
Syntax	@rep:service
Usage	Indicates that a Java file is a business service object (as opposed to a normal Java API) Use this tag as is in your Java file. Refer to the example section below. It takes no parameters

Annotation Type	@rep:service
Example	<pre> /**  * The Purchase Order service lets you to  * view, update, acknowledge and  * approve purchase orders. It also lets you  * receive items, and obtain  * pricing by line item.  *  * @see  * oracle.apps.fnd.framework.toolbox.tutorial.P  * urchaseOrderSDO  * @see  * oracle.apps.fnd.framework.toolbox.tutorial.P  * urchaseOrderAcknowledgementsSDO  * @see  * oracle.apps.fnd.framework.toolbox.tutorial.P  * urchaseOrderReceiptsSDO  *  * @rep:scope public  * @rep:displayname Purchase Order Service  * @rep:implementation  * oracle.apps.fnd.framework.toolbox.tutorial.s  * erver.PurchaseOrderSAMImpl  * @rep:product PO  * @rep:category BUSINESS_ENTITY  * PO_PURCHASE_ORDER  * <b>@rep:service</b>  */ </pre>
Required	Required for Java files at the class level.
Default	None.
Level	Class.
Multiple Allowed	No.
Data Model	
Comments	

#### **@rep:servicedoc**

Annotation Type	@rep:servicedoc
Syntax	@rep:servicedoc

Annotation Type	@rep:servicedoc
Usage	Indicates that a Java file is an SDO (as opposed to a normal Java API). Use this tag as is in your java file. Refer to the example section below. It takes no parameters.
Example	<pre>/**  * The Purchase Order Data Object holds the  * purchase order data including  * nested data objects such as lines and  * shipments.  *  * @see  * oracle.apps.fnd.framework.toolbox.tutorial.P  * urchaseOrderLineSDO  *  * @rep:scope public  * @rep:displayname Purchase Order Data  * Object  * @rep:product PO  * @rep:category BUSINESS_ENTITY  * PO_PURCHASE_ORDER  * @rep:servicedoc  */</pre>
Required	Required for Java files at the class level.
Default	None.
Level	Class.
Multiple Allowed	No.
Data Model	
Comments	Developers do not need to enter this annotation because it is automatically generated.

#### **@rep:synchronicity**

Annotation Type	@rep:synchronicity
Syntax	@rep:synchronicity <SYNCH or ASYNCH>
Usage	Specifies synchronous or asynchronous behavior.

Annotation Type	<b>@rep:synchronicity</b>
Example	<code>@rep:synchronicity SYNCH</code>
Required	Optional.
Default	Is defaulted based on module type. For example, ASYNCH for XML Gateway and SYNCH for Business Service Object.
Level	Class or method.
Multiple Allowed	No.
Data Model	
Comments	

#### **@rep:appscontext**

Annotation Type	<b>@rep:appscontext</b>
Syntax	<code>@rep:appscontext &lt;NONE, APPL, RESP, USER, NLS, or ORG&gt;</code>
Usage	Specifies the context required to execute the method.
Example	<code>@rep:appscontext USER</code>
Required	Optional.
Default	NONE
Level	Method.
Multiple Allowed	No, only one allowed per method.
Data Model	
Comments	

### **@rep:comment**

Annotation Type	@rep:comment
Syntax	@rep:comment <comment>
Usage	This annotation is skipped by the parsers. It is for use by product teams when a non-published comment is desired.
Example	@rep:comment This is a sample comment.
Required	Optional.
Default	None.
Level	Any.
Multiple Allowed	
Data Model	
Comments	

### **@rep:primaryinstance**

Annotation Type	@rep:primaryinstance
Syntax	@rep:primaryinstance
Usage	To indicate the primary instance of an overloaded method or procedure.
Example	
Required	Required for all overloaded methods and procedures.
Default	None.
Level	Method or procedure.

Annotation Type	<b>@rep:primaryinstance</b>
Multiple Allowed	No.
Data Model	
Comments	The primary instance's display name and description will be used in the browser UI when a list of methods is displayed. The non-primary instances (such as, the overloads) should have descriptions that emphasize how they differ from the primary (such as, "This variant allows specification of the org_id."). The non-primary display names and descriptions will only be displayed when viewing the details of the overloaded interface.

#### **@rep:usesmap**

Annotation Type	<b>@rep:usesmap</b>
Syntax	<code>@rep:usesmap &lt;map_name&gt; &lt;sequence_number&gt;</code>
Usage	<p>To indicate the E-Commerce Gateway maps that are associated with a concurrent program.</p> <p><code>&lt;map_name&gt;</code> where <code>map_name</code> is the default map name.</p> <p><code>&lt;sequence_number&gt;</code> is an integer used to tell the UI the display order of the different pieces.</p>
Example	<code>@rep:usesemap SampleMap 2</code>
Required	Optional.
Default	None.
Level	Any.
Multiple Allowed	Yes.
Data Model	

Annotation Type	@rep:usesmap
Comments	The default map name has the following naming convention "EC_XXXX_FF" where XXXX is the 4-letter acronym for your transaction.





---

## Configuring Server Connection

### Overview

If your Web services are exposed and invoked through BPEL PM, to successfully deploy BPEL processes to Oracle Application Server, you must first establish the necessary server connection information used at run time in the background.

How to configure the server connection is described in the following sections:

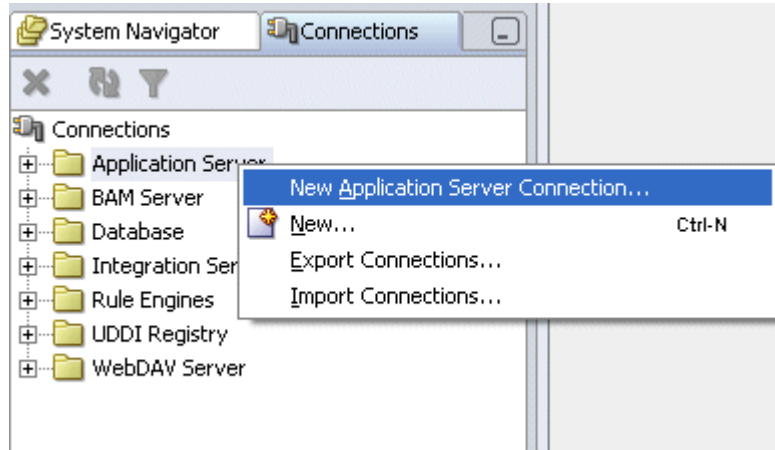
- Creating the Application Server Connection, page B-1
- Creating the Integration Server Connection, page B-6

### Creating the Application Server Connection

You must establish a connectivity between the design-time environment and the server you want to deploy it to. In order to establish such a connectivity, you must create the application server connection.

Use the following steps to create the application server connection:

1. From Oracle JDeveloper (for example, Oracle JDeveloper 10.1.3.3), select **View > Connection Navigator** to open the Connections tab.
2. Right-click on the Application Server and select **New Application Server Connection**.

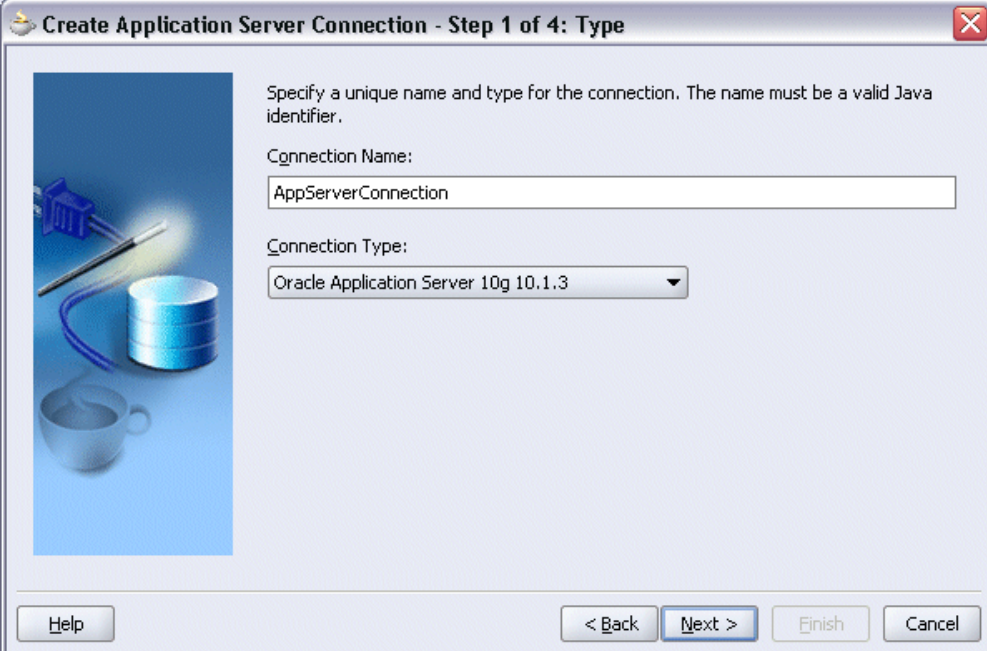


This opens the Create Application Server Connection wizard. Click **Next** in the Welcome page of the wizard.

3. Enter the connection name and select **Oracle Application Server 10g 10.1.3** as the connection type.

Click **Next**.

**Enter the Server Name and Type**



**Create Application Server Connection - Step 1 of 4: Type**

Specify a unique name and type for the connection. The name must be a valid Java identifier.

Connection Name:  
AppServerConnection

Connection Type:  
Oracle Application Server 10g 10.1.3

Help    < Back    **Next >**    Finish    Cancel

4. Enter a valid username (such as oc4jadmin) and the password information (such as welcome) and click **Next**.

### Enter Server Authentication Information



The dialog box is titled "Create Application Server Connection - Step 2 of 4: Authentication". It features a blue-themed icon on the left showing a network cable, a database cylinder, and a coffee cup. The main text area contains the instruction: "Specify a username and password to authenticate the connection. To bypass authentication at runtime, select Deploy Password." Below this, there are two text input fields: "Username:" with the value "oc4jadmin" and "Password:" with the value "\*\*\*\*\*". A checkbox labeled "Deploy Password" is present and unchecked. At the bottom, there are four buttons: "Help", "< Back", "Next >" (which is highlighted with a blue border), "Finish", and "Cancel".

Specify a username and password to authenticate the connection. To bypass authentication at runtime, select Deploy Password.

Username:  
oc4jadmin

Password:  
\*\*\*\*\*

☐ Deploy Password

Help < Back Next > Finish Cancel

5. Select **Single Instance** radio button. Enter appropriate values for the Server connection host name, port, and OC4J instance information.

Click **Next**.

### Specifying Server Connection Information

**Create Application Server Connection - Step 3 of 4: Connection**

Please provide the host name and OPMN port for the OPMN instance and the name of the OC4J instance, component, or group being managed by OPMN. This information is used to assemble an URL used to create a JMX connection to the server.

Connect To: ☒ Single Instance ☐ Group

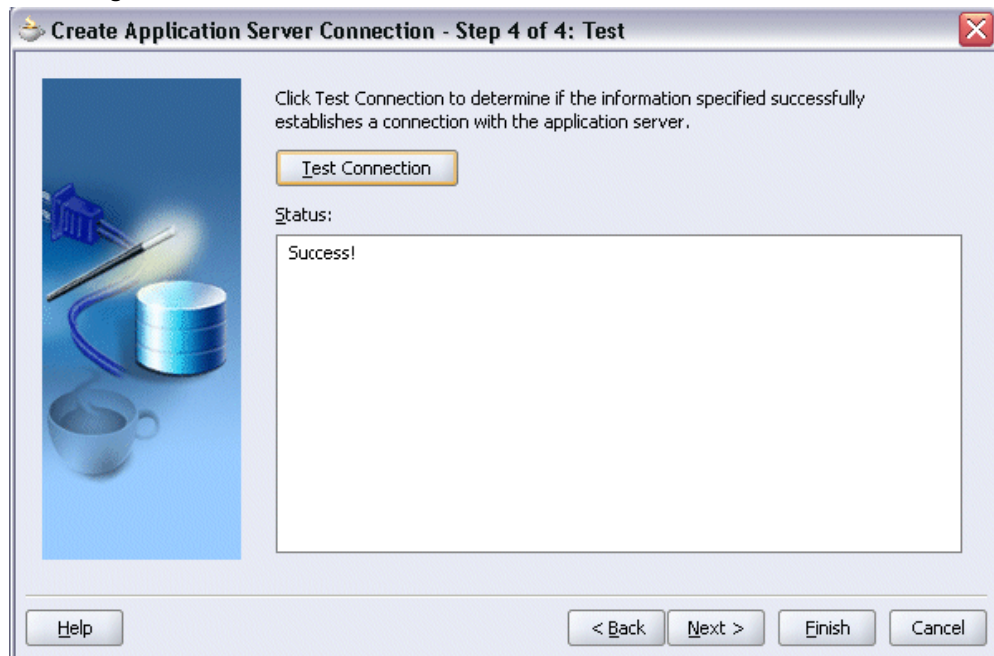
Host Name:  OPMN Port:

OC4J Instance Name:

Help < Back Next > Finish Cancel

6. Click **Test Connection** to validate your server configuration. You should find "Success!" populated in the Status window.

### Validating the Server Connection

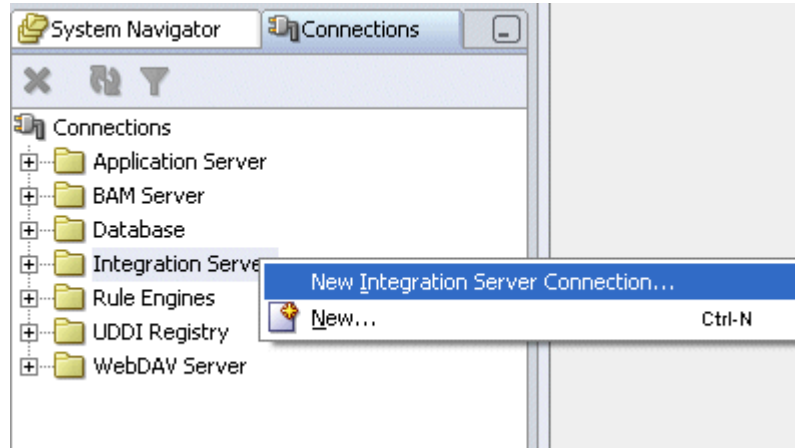


Click **Finish**.

## Creating the Integration Server Connection

Use the following steps to create the integration server connection:

1. From Oracle JDeveloper, select **View > Connection Navigator** to open the Connections tab.
2. Right-click on the Integration Server and select **New Integration Server Connection**.

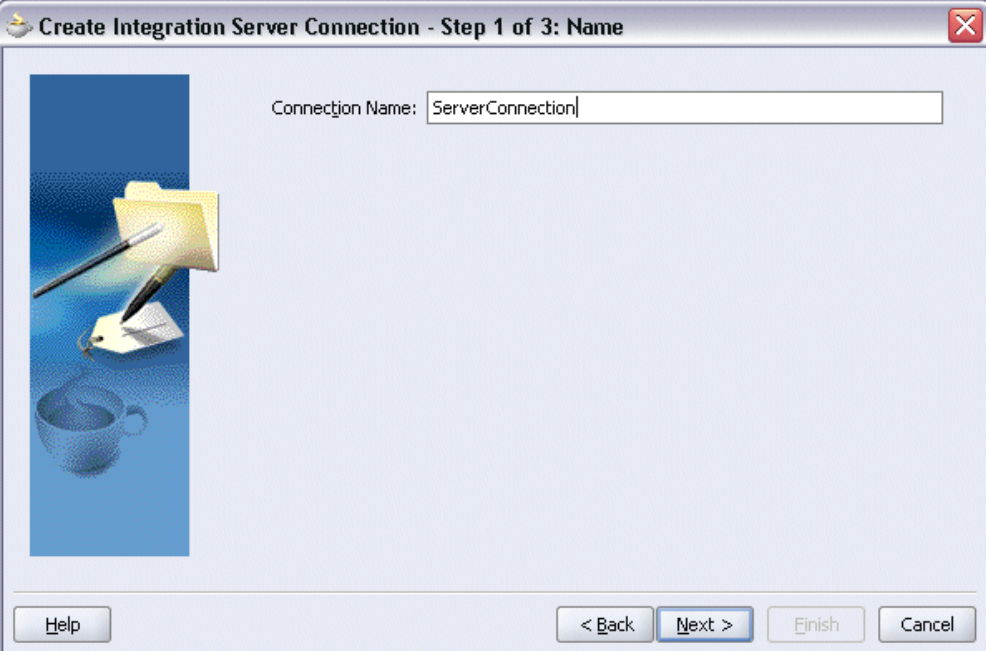


This opens the Create Integration Server Connection wizard. Click **Next** in the Welcome page of the wizard.

3. Enter the connection name.

Click **Next**.

**Enter the Connection Name**



Create Integration Server Connection - Step 1 of 3: Name

Connection Name:

Help < Back Next > Finish Cancel

4. Select the Application Server name you just created from the drop-down list. The Host Name field will be populated automatically based on your selection. Enter HTTP port number.  
Click **Next**.



**Enter Integration Server Connection Information**



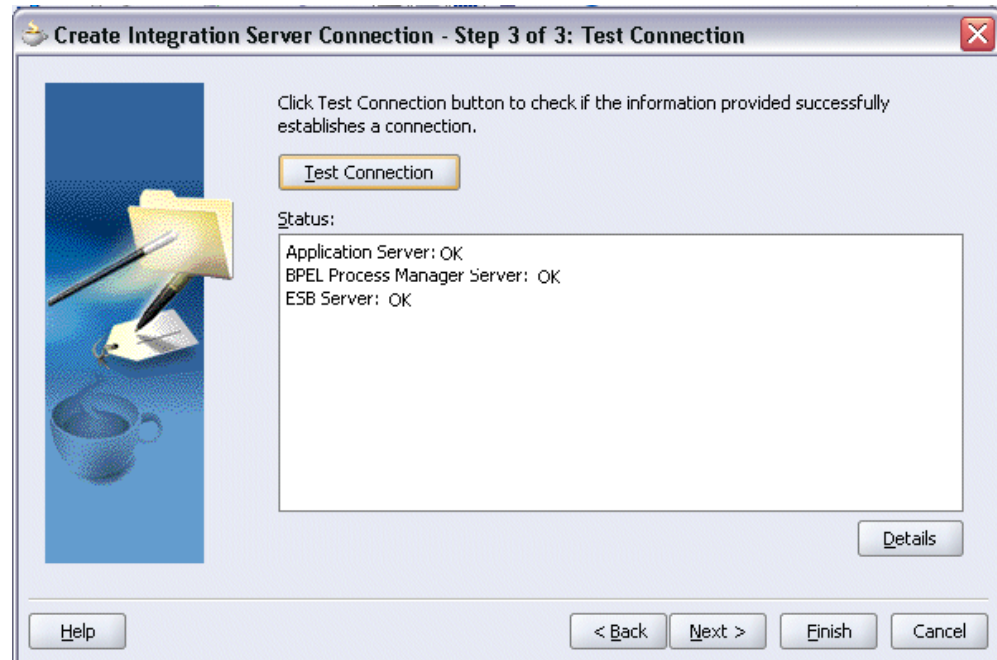
The screenshot shows a Windows-style dialog box titled "Create Integration Server Connection - Step 2 of 3: Connection". On the left is a blue vertical panel with a graphic of a folder, a pen, a notepad, and a coffee cup. The main area contains the following fields and controls:

- Application Server:** A dropdown menu showing "AppServerConnection" with a pencil icon and a green plus icon to its right.
- Host Name:** A text box containing "localhost".
- Port Number:** A text box containing "12345".
- ☒ Add host name to the list of proxy exceptions

At the bottom, there are four buttons: "Help", "< Back", "Next >" (which is highlighted with a blue border), "Finish", and "Cancel".

5. Click **Test Connection** to validate your integration server connection. You should find success messages populated in the Status window.

### Validating the Integration Server Connection



Click **Finish**.

---

## Sample Payload

### Sample Payload for Creating Supplier Ship and Debit Request

The following information shows the sample payload in the  
`InputCreateSDRequest.xml` file:

```

<?xml version="1.0" encoding="UTF-8"?>
  <cre:InputParameters
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:cre="http://xmlns.oracle.com/apps/ozf/soapprovider/plsql/ozf_sd_req
      uest_pub/create_sd_request/">
    <cre:P_API_VERSION_NUMBER>1.0</cre:P_API_VERSION_NUMBER>
    <cre:P_INIT_MSG_LIST>T</cre:P_INIT_MSG_LIST>
    <cre:P_COMMIT>F</cre:P_COMMIT>
    <cre:P_VALIDATION_LEVEL>100</cre:P_VALIDATION_LEVEL>
    <cre:P_SDR_HDR_REC>
      <cre:REQUEST_NUMBER>SDR-CREATE-BPEL1</cre:REQUEST_NUMBER>

    <cre:REQUEST_START_DATE>2008-08-18T12:00:00</cre:REQUEST_START_DATE>
      <cre:REQUEST_END_DATE>2008-10-18T12:00:00</cre:REQUEST_END_DATE>>
      <cre:USER_STATUS_ID>1701</cre:USER_STATUS_ID>
      <cre:REQUEST_OUTCOME>IN_PROGRESS</cre:REQUEST_OUTCOME>
      <cre:REQUEST_CURRENCY_CODE>USD</cre:REQUEST_CURRENCY_CODE>
      <cre:SUPPLIER_ID>601</cre:SUPPLIER_ID>
      <cre:SUPPLIER_SITE_ID>1415</cre:SUPPLIER_SITE_ID>
      <cre:REQUESTOR_ID>100001499</cre:REQUESTOR_ID>
      <cre:ASSIGNEE_RESOURCE_ID>100001499</cre:ASSIGNEE_RESOURCE_ID>
      <cre:ORG_ID>204</cre:ORG_ID>
      <cre:ACCRUAL_TYPE>SUPPLIER</cre:ACCRUAL_TYPE>
      <cre:REQUEST_DESCRIPTION>Create</cre:REQUEST_DESCRIPTION>

    <cre:SUPPLIER_CONTACT_EMAIL_ADDRESS>sdr.supplier@testing.com</cre:SUPPLI
      ER_CONTACT_EMAIL_ADDRESS>

    <cre:SUPPLIER_CONTACT_PHONE_NUMBER>2255</cre:SUPPLIER_CONTACT_PHONE_NUMB
      ER>
      <cre:REQUEST_TYPE_SETUP_ID>400</cre:REQUEST_TYPE_SETUP_ID>
      <cre:REQUEST_BASIS>Y</cre:REQUEST_BASIS>
      <cre:USER_ID>1002795</cre:USER_ID>
    </cre:P_SDR_HDR_REC>
    <cre:P_SDR_LINES_TBL>
      <cre:P_SDR_LINES_TBL_ITEM>
        <cre:PRODUCT_CONTEXT>PRODUCT</cre:PRODUCT_CONTEXT>
        <cre:INVENTORY_ITEM_ID>2155</cre:INVENTORY_ITEM_ID>
        <cre:ITEM_UOM>Ea</cre:ITEM_UOM>
        <cre:REQUESTED_DISCOUNT_TYPE>%</cre:REQUESTED_DISCOUNT_TYPE>
        <cre:REQUESTED_DISCOUNT_VALUE>20</cre:REQUESTED_DISCOUNT_VALUE>
        <cre:COST_BASIS>200</cre:COST_BASIS>
        <cre:MAX_QTY>200</cre:MAX_QTY>
        <cre:APPROVED_DISCOUNT_TYPE>%</cre:APPROVED_DISCOUNT_TYPE>
        <cre:APPROVED_DISCOUNT_VALUE>20</cre:APPROVED_DISCOUNT_VALUE>
        <cre:APPROVED_MAX_QTY>200</cre:APPROVED_MAX_QTY>
        <cre:VENDOR_APPROVED_FLAG>Y</cre:VENDOR_APPROVED_FLAG>
        <cre:PRODUCT_COST_CURRENCY>USD</cre:PRODUCT_COST_CURRENCY>
        <cre:END_CUSTOMER_CURRENCY>USD</cre:END_CUSTOMER_CURRENCY>
      </cre:P_SDR_LINES_TBL_ITEM>
    </cre:P_SDR_LINES_TBL>
    <cre:P_SDR_CUST_TBL>
      <cre:P_SDR_CUST_TBL_ITEM>
        <cre:CUST_ACCOUNT_ID>1290</cre:CUST_ACCOUNT_ID>
        <cre:PARTY_ID>1290</cre:PARTY_ID>
        <cre:SITE_USE_ID>10479</cre:SITE_USE_ID>
        <cre:CUST_USAGE_CODE>BILL_TO</cre:CUST_USAGE_CODE>
        <cre:END_CUSTOMER_FLAG>N</cre:END_CUSTOMER_FLAG>
      </cre:P_SDR_CUST_TBL_ITEM>
      <cre:P_SDR_CUST_TBL_ITEM>
        <cre:CUST_ACCOUNT_ID>1287</cre:CUST_ACCOUNT_ID>

```

```
<cre:PARTY_ID>1287</cre:PARTY_ID>
  <cre:SITE_USE_ID>1418</cre:SITE_USE_ID>
  <cre:CUST_USAGE_CODE>CUSTOMER</cre:CUST_USAGE_CODE>
  <cre:END_CUSTOMER_FLAG>Y</cre:END_CUSTOMER_FLAG>
  <cre:P_SDR_CUST_TBL>
</cre:P_SDR_CUST_TBL>
</cre:InputParameters>
```

## Sample Payload for Inbound Process Purchase Order XML Transaction

The following information shows the sample payload in the `order_data_xmlg.xml` file:

```

<?xml version="1.0">
  <PROCESS_PO_007> <!--xmlns="http://TargetNamespace.com/ServiceName"-->
    <CNTROLAREA>
      <BSR>
        <VERB>PROCESS</VERB>
        <NOUN>PO</NOUN>
        <REVISION>007</REVISION>
      </BSR>
    <SENDER>
      <LOGICALID/>
      <COMPONENT>BPOL</COMPONENT>
      <TASK>POISSUE</TASK>
      <REFERENCEID>refid</REFERENCEID>
      <CONFIRMATION>2</CONFIRMATION>
      <LANGUAGE>ENG</LANGUAGE>
      <CODEPAGE>US7ASCII</CODEPAGE>
      <AUTHID>APPS</AUTHID>
    </SENDER>
    <DATETIME qualifier="CREATION">
      <YEAR>2002</YEAR>
      <MONTH>10</MONTH>
      <DAY>09</DAY>
      <HOUR>16</HOUR>
      <MINUTE>45</MINUTE>
      <SECOND>47</SECOND>
      <SUBSECOND>356</SUBSECOND>
      <TIMEZONE>-0800</TIMEZONE>
    </DATETIME>
  </CNTROLAREA>
  <DATAAREA>
    <PROCESS_PO>
      <POORDERHDR>
        <DATETIME qualifier="DOCUMENT">
          <YEAR>2002</YEAR>
          <MONTH>10</MONTH>
          <DAY>09</DAY>
          <HOUR>16</HOUR>
          <MINUTE>40</MINUTE>
          <SECOND>34</SECOND>
          <SUBSECOND>000</SUBSECOND>
          <TIMEZONE>+0100</TIMEZONE>
        </DATETIME>
        <OPERAMT qualifier="EXTENDED" type="T">
          <VALUE>107.86</VALUE>
          <NUMOFDEC>6</NUMOFDEC>
          <SIGN>+</SIGN>
          <CURRENCY>USD</CURRENCY>
          <UOMVALUE>1</UOMVALUE>
          <UOMNUMDEC>0</UOMNUMDEC>
          <UOM>Ea</UOM>
        </OPERAMT>
        <POID>refid</POID>
        <POTYPE>Mixed</POTYPE>
        <CONTRACTS/>
        <DESCRIPTN/>
        <NOTES index="1"/>
        <USERAREA/>
        <PARTNER>
          <NAME index="1"/>
          <ONETIME>0</ONETIME>
          <PARTNRID/>

```

```

<PARTNRTYPE>SoldTo</PARTNRTYPE>
  <PARTNRIDX>BWSANJOSE</PARTNRIDX>
  </PARTNER>
</POORDERHDR>
<POORDERLIN>
  <QUANTITY qualifier="ORDERED">
    <VALUE>1</VALUE>
    <NUMOFDEC>0</NUMOFDEC>
    <SIGN>+</SIGN>
    <UOM>Ea</UOM>
  </QUANTITY>
  <OPERAMT qualifier="UNIT" type="T">
    <VALUE>107.86</VALUE>
    <NUMOFDEC>6</NUMOFDEC>
    <SIGN>+</SIGN>
    <CURRENCY>USD</CURRENCY>
    <UOMVALUE>1</UOMVALUE>
    <UOMNUMDEC>0</UOMNUMDEC>
    <UOM>Ea</UOM>
  </OPERAMT>
  <POLINENUM>1</POLINENUM>
  <ITEMRV/>
  <NOTES index="1"/>
  <ITEM>LAP-DLX</ITEM>
  <POLINESCHD>
    <DATETIME qualifier="NEEDELV">
      <YEAR>2002</YEAR>
      <MONTH>10</MONTH>
      <DAY>09</DAY>
      <HOUR>00</HOUR>
      <MINUTE>00</MINUTE>
      <SECOND>00</SECOND>
      <SUBSECOND>000</SUBSECOND>
      <TIMEZONE>+0100</TIMEZONE>
    </DATETIME>
    <QUANTITY qualifier="ORDERED">
      <VALUE>1</VALUE>
      <NUMOFDEC>0</NUMOFDEC>
      <SIGN>+</SIGN>
      <UOM>Ea</UOM>
    </QUANTITY>
    <PSCLINENUM>1</PSCLINENUM>
    <USERAREA/>
  </POLINESCHD>
</POORDERLIN>
</PROCESS_PO>
</DATAAREA>
</PROCESS_PO_007>

```





---

# Understanding Basic BPEL Process Creation

## Overview

To design a composite service, integration developers use a Web service composition language BPEL to specify the invocation sequence through Oracle BPEL Process Manager (PM). This composite service has its own WSDL definition and endpoint through the creation of a Partner Link which allows a business event, for example, to be published to the Oracle BPEL Process Manager or to interact with a partner service.

To efficiently utilize a BPEL process in orchestrating a meaningful business flow, the basic concept of creating a BPEL process is discussed in this section.

## Understanding BPEL Business Processes

A BPEL process specifies the exact order in which participating Web services should be invoked either sequentially or in parallel. In a typical scenario, a BPEL process receives a request. To fulfill it, the process invokes the involved Web services and then responds to the original requestor. Because the BPEL process communicates with other Web services, it heavily relies on the WSDL description of the Web services invoked by the composite services.

For example, a BPEL process consists of steps that are placed in the exact order that will be invoked. And these steps are called 'activities'. Each activity represents basic construct and is used for a common task, such as use `<invoke>` activity to invoke a Web service; use `<reply>` to generate a response for synchronous operations.

### Key Activities and Message Patterns

In supporting Web services and message exchanges over the Web, there are many communication patterns to model the processes. For example, a basic request - response pattern can be used in a synchronous or asynchronous way. A *synchronous request - response* pattern is one that waits for a response before continuing on; while an *asynchronous request - response* pattern does not wait for a response before continuing on

which allows operations to occur in parallel until the response is ready. Another pattern can be *request only* operation that does not require response at all.

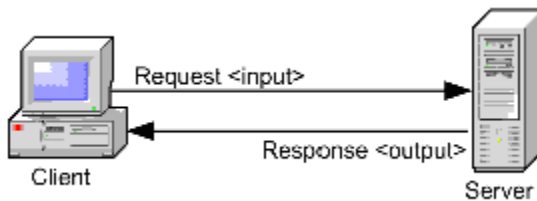
Based on the message patterns, appropriate activities representing various actions can be orchestrated in a meaningful way and enable the services.

To have better explanation about the message patterns and some key activities that are frequently used in a BPEL process, the following two message patterns are used to further describe how to build a simple BPEL process:

### **Synchronous Request - Response BPEL Process**

For synchronous request-response service type, a consumer or client sends a request to a service, and receives an immediate reply.

#### ***Synchronous Request - Response Message Pattern***

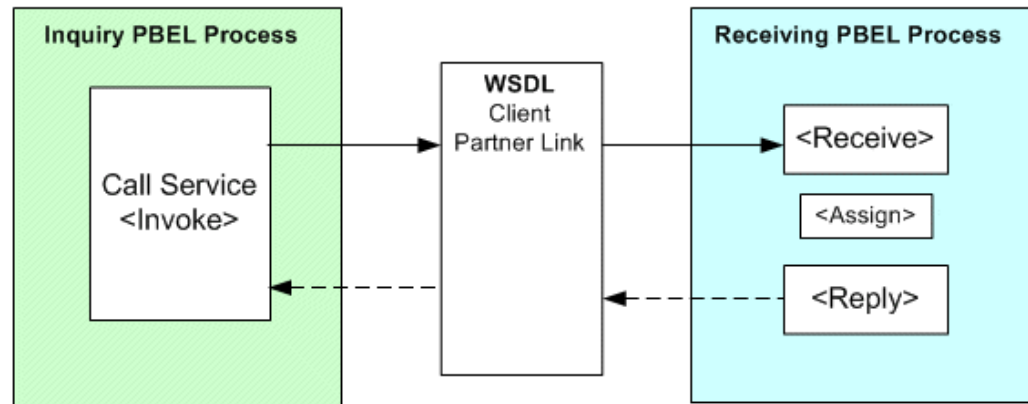


Many online banking tasks are programmed in request-response mode. For example, a request for an account balance is executed as follows:

- A customer (the client) sends a request for an account balance to the Account Record Storage System (the server).
- The Account Record Storage System (the server) sends a reply to the customer (the client), specifying the dollar amount in the designated account.

The synchronous request-response interaction pattern interpreted in a BPEL process can be illustrated in the following diagram:

### ***Synchronous Request-Response Interaction Pattern in BPEL***



In the above diagram, there are two BPEL processes involved to complete the synchronous request-response service:

- **BPEL Process as Client (Request)**

When the BPEL process is on the client side of a synchronous transaction, it needs an **Invoke** activity to send the request and receive the reply and a **PartnerLink** to carry information between the inquiry BPEL process and a Web service.

- **BPEL Process as Service (Response)**

When the BPEL process is on the service side of a synchronous transaction, it needs a **Receive** activity to accept the incoming request, and a **Reply** activity to return either the requested information or an error message (a fault). Additionally, it requires a **PartnerLink** to carry information between the Web service and the inquiry BPEL process.

**Note:** Sometime, an Assign activity is placed before a Reply activity to take received data as an input variable and assign it the Reply activity as an output variable in respond to the request.

### **Partner Link**

A partner link defines the location and the role of the Web services that the BPEL process connects to in order to perform tasks, as well as the variables used to carry information between the Web service and the BPEL process. A partner link is required for each Web service that the BPEL process calls.

### **Invoke Activity**

An Invoke activity opens a port in the BPEL process to send and receive data. It uses this port to submit the required data and receive the response.

In the account balance inquiry example, the Invoke activity submits the account number

entered by the customer to the server and receives dollar amount in return as the account balance. For synchronous callbacks, the Java rule function supports the feature through Business Event System. Thus, only one port is needed for both the send and receive functions.

### **Receive Activity**

A Receive activity waits for an incoming request data as an input variable. For example, the Receive activity accepts the account balance inquiry by taking the account number as an input variable to the server.

### **Reply Activity**

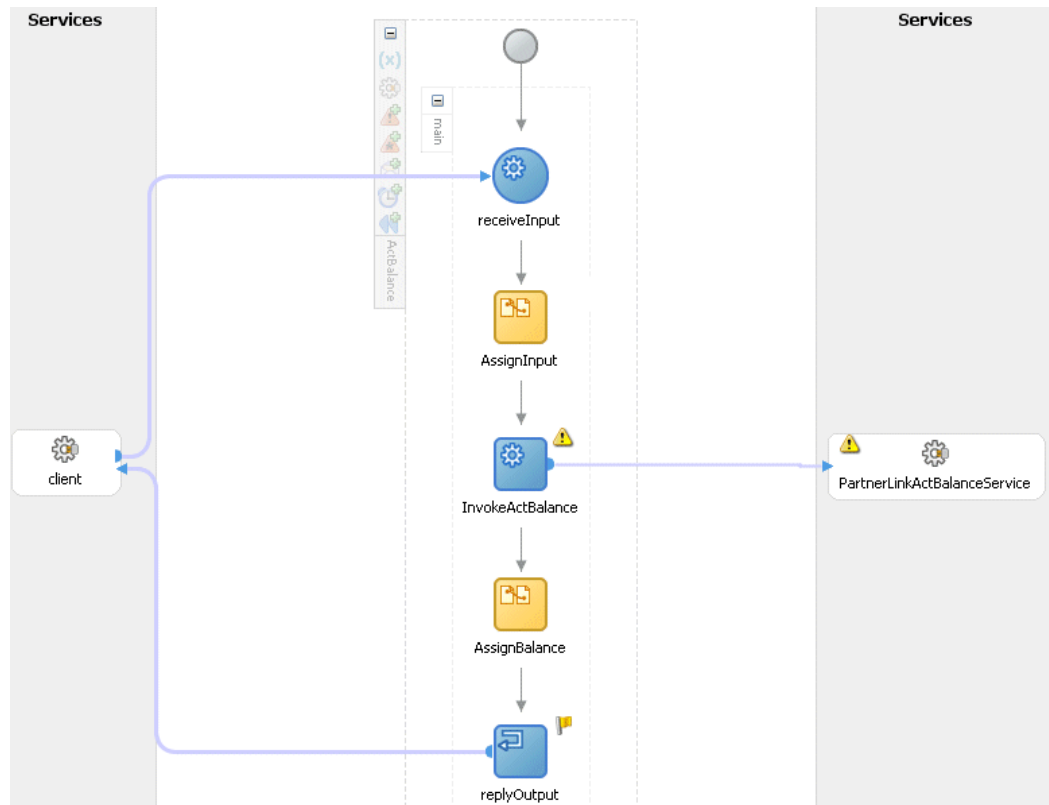
A Reply activity enables the business process to send a response message in reply to a message that was received through a Receive activity. For example, the Reply activity takes the account balance from the server as an output variable and sends it back to the requestor.

The combination of a receive and a reply forms a request-response operation.

## **Orchestrating a Synchronous BPEL Process**

The composite BPEL process design flow should be orchestrated with necessary BPEL components or activities so as to successfully invoke a synchronous Web service.

Use the account balance inquiry as an example. The service invocation sequence can be orchestrated in Oracle JDeveloper BPEL Designer as shown in the following diagram:



1. A client sends the request by entering account number for the balance inquiry.
2. The Receive activity receives it as an input variable.
3. The Assign activity takes the account number and passes it to the Invoke activity.
4. The Invoke activity submits the account number entered by the client to a `ActBalanceService` Partner Link and receives dollar amount in return as the account balance.
5. The Assign activity then takes the dollar amount as an input variable and passes it to the Reply activity.
6. The Reply activity takes the account balance and replies to the requestor.

The above composite service - BPEL process requires the following tasks at the design time:

- Adding a Partner Link
- Adding an Invoke Activity
- Adding a Receive Activity

- Adding a Reply Activity
- Adding an Assign Activity

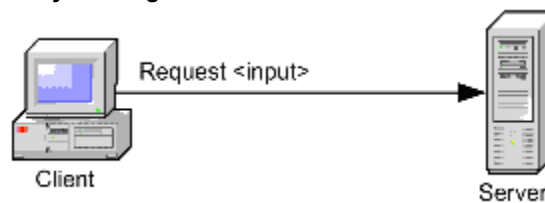
Once a Partner Link is successfully added to a synchronous BPEL project, a WSDL description URL that corresponds to the ActBalanceService business event service with appropriate event payload will be automatically generated. This ActBalanceService Partner Link serves as a bridge to communicate information between the service and the synchronous BPEL project.

**Note:** The generated WSDL URL of a BPEL process can also be used in defining an event subscription if the BPEL process is used for service invocation through the Business Event System.

### One-Way BPEL Process

For One-way or request only service type, there is only one input element which is a client's request for a service and no response is expected.

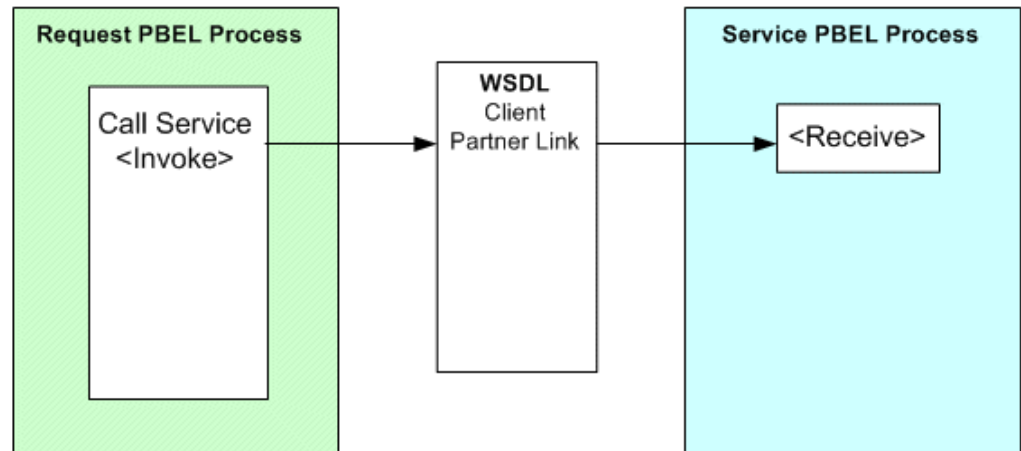
#### *Request Only Message Pattern*



For example, a stock symbol sends updated price to the stock quote service when the price change using the request only operation. The server updates the stock price but no response is sent back.

This type of interaction pattern interpreted in a BPEL process can be illustrated in the following diagram:

### ***One-Way (Request Only) Interaction Pattern in BPEL***

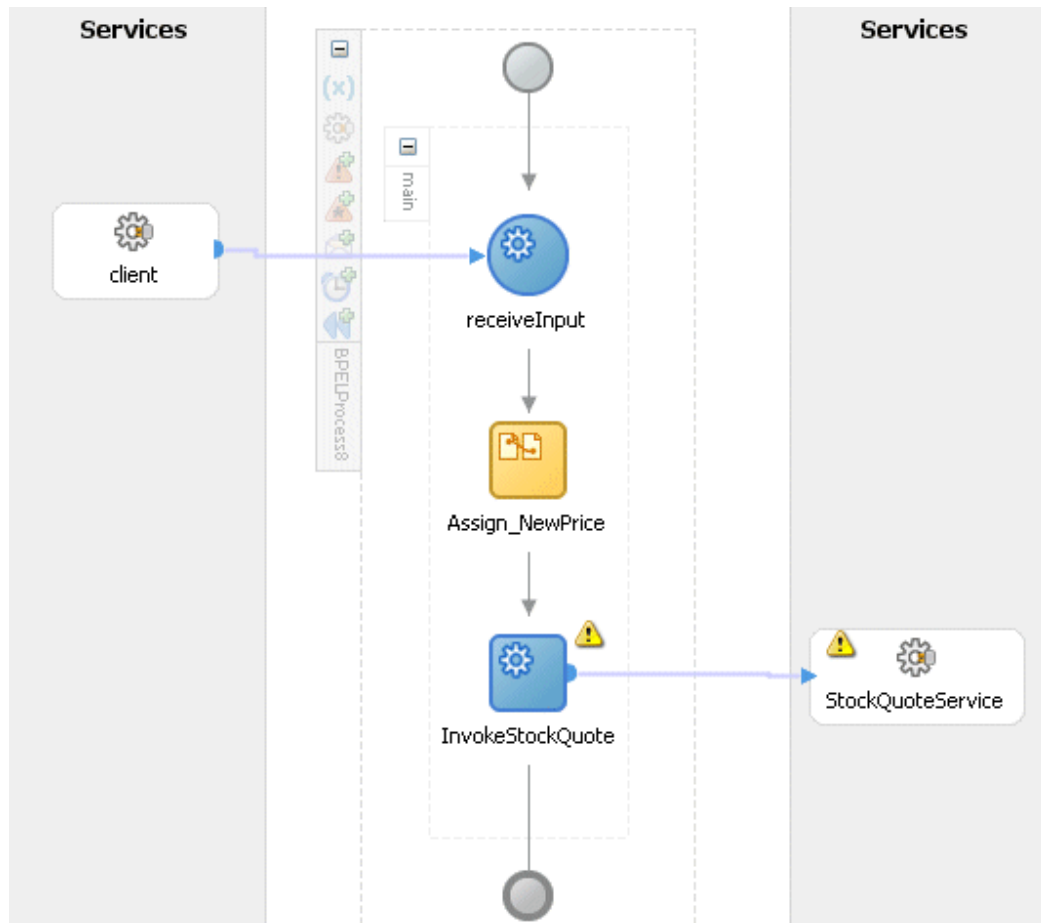


The following two BPEL processes are involved in this type of service:

- **BPEL Process as Client (Request)**  
As the client, the BPEL process needs a valid **PartnerLink** and an **Invoke** activity with the target service and the message. As with all partner activities, the WSDL file defines the interaction.
- **BPEL Process as Service**  
To accept a message from the client's request, the BPEL process needs a **Receive** activity only.

In the stock quote update example, the Invoke activity submits a stock symbol along with a market price to the StockQuote service in a server. The Receive activity takes the new price as an input and update the StockQuote service.

The following diagram illustrates the service invocation sequence for the stock update example in Oracle JDeveloper BPEL Designer:



1. A client sends the stock symbol and updated price.
2. The Receive activity receives it as an input variable.
3. The Assign activity takes the new price and passes it to the Invoke activity.
4. The Invoke activity submits a stock symbol along with an updated price to the `StockQuoteService` Partner Link in a server.

This composite service - BPEL process requires the following tasks at the design time:

- Adding a Partner Link
- Adding an Invoke Activity
- Adding a Receive Activity
- Adding an Assign Activity

Like synchronous request - response operation, once a Partner Link is successfully added to a BPEL project, a WSDL description URL that corresponds to the service with



appropriate event payload will be automatically generated.



---

# Glossary

## **Agent**

A named point of communication within a system.

## **Agent Listener**

A type of service component that processes event messages on inbound agents.

## **BPEL**

Business Process Execution Language (BPEL) provides a language for the specification of executable and abstract business processes. By doing so, it extends the services interaction model and enables it to support business transactions. BPEL defines an interoperable integration model that should facilitate the expansion of automated process integration in both the intra-corporate and the business-to-business spaces.

## **Business Event**

See Event.

## **Concurrent Manager**

An Oracle E-Business Suite component that manages the queuing of requests and the operation of concurrent programs.

## **Concurrent Program**

A concurrent program is an executable file that performs a specific task, such as posting a journal entry or generating a report.

## **Event**

An occurrence in an internet or intranet application or program that might be significant to other objects in a system or to external agents.

## **Event Activity**

A business event modelled 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 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 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.

**Function**

A PL/SQL stored procedure that can define business rules, perform automated tasks within an application, or retrieve application information. The stored procedure accepts standard arguments and returns a completion result.

**Integration Repository**

Oracle Integration Repository is the key component or user interface for Oracle E-Business Suite Integrated SOA Gateway. This centralized repository stores native packaged integration interface definitions and composite services.

**Interface Type**

Integration interfaces are grouped into different interface types.

**Loose Coupling**

Loose coupling describes a resilient relationship between two or more systems or organizations with some kind of exchange relationship. Each end of the transaction makes its requirements explicit and makes few assumptions about the other end.

**Lookup Code**

An internal name of a value defined in a lookup type.

**Lookup Type**

A predefined list of values. Each value in a lookup type has an internal and a display name.

**Message**

The information that is sent by a notification activity. A message must be defined before it can be associated with a notification activity. A message contains a subject, a priority, a body, and possibly one or more message attributes.

**Message Attribute**

A variable that you define for a particular message to either provide information or prompt for a response when the message is sent in a notification. You can use a predefined item type attribute as a message attribute. Defined as a 'Send' source, a message attribute gets replaced with a runtime value when the message is sent. Defined as a 'Respond' source, a message attribute prompts a user for a response when the message is sent.

**Notification**

An instance of a message delivered to a user.

**Notification Worklist**

A Web page that you can access to query and respond to workflow notifications.

**Operation**

An abstract description of an action supported by a service.

**Port**

A port defines an individual endpoint by specifying a single address for a binding.

**Port Type**

A port type is a named set of abstract operations and abstract messages involved.

**Process**

A set of activities that need to be performed to accomplish a business goal.

**Service**

A service is a collection of related endpoints.

**Service Component**

An instance of a Java program which has been defined according to the Generic Service Component Framework standards so that it can be managed through this framework.

**SOA**

Service-oriented Architecture (SOA) is an architecture to achieve loose coupling among interacting software components and enable seamless and standards-based integration in a heterogeneous IT ecosystem.

**SOAP**

Simple Object Access Protocol (SOAP) is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment. It uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols.

**Subscription**

See Event Subscription.

**Web Services**

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

**Workflow Engine**

The Oracle Workflow component that implements a workflow process definition. The Workflow Engine manages the state of all activities for an item, automatically executes functions and sends notifications, maintains a history of completed activities, and detects error conditions and starts error processes. The Workflow Engine is implemented in server PL/SQL and activated when a call to an engine API is made.

**WSDL**

Web Services Description Language (WSDL) is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint.

**WS-Addressing**

WS-Addressing is a way of describing the address of the recipient (and sender) of a message, inside the SOAP message itself.

**WS-Security**

WS-Security defines how to use XML Signature in SOAP to secure message exchanges, as an alternative or extension to using HTTPS to secure the channel.

---

# Index

## B

---

### business events

- annotate, 10-17
- download, 10-14
- Upload file to the database, 10-20
- Upload iLDT files, 10-21
- validate, 10-19

### Business Service Objects Design Tasks

- Adding an Assign Activity, 7-18
- Adding an Invoke Activity, 7-15
- Adding a Partner Link for File Adapter, 7-9
- Creating a New BPEL Project, 7-5
- Creating a Partner Link, 7-7

## C

---

### Composite Services

- download, 9-2
- modify, 9-4
- overview, 9-1
- view, 9-2

### connection information

- Application Server Connection, B-1
- Integration Server Connection, B-6

### create and upload custom interfaces

- business events, 10-14
- composite service validation, 10-13
- creation, 10-4, 10-9
- View and Administer composite services, 10-13

### create and use custom interfaces

- create steps, 10-2

overview, 10-1

use custom interfaces, 10-22

### create BPEL

- message pattern, D-1
- One-Way, D-6
- Synchronous Request-Response, D-2

### create custom interfaces

- composite services, 10-8
- interface types, 10-2

### Creating BPEL Using Business Events

- Assign, 5-23
- Create a New BPEL Project, 5-5
- Create a Partner Link AQ Adapter, 5-6
- Create a Partner Link File Adapter, 5-15
- invoke, 5-21
- receive, 5-13

### Creating Invoker Event Subscription

- Creating Error Subscription, 11-15
- Creating Subscription with 'Invoke Web Service', 11-8

## D

---

### deploy and test bpel

- deploy bpel, 7-24
- test bpel, 7-25

### Deploy and Test Concurrent Program

- deploy bpel, 6-24, 6-25

### Deploy and Test Custom BPEL

- deploy bpel, 10-47
- test bpel, 10-49

### Deploy and Test Event BPEL

- deploy bpel, 5-26

- test bpel, 5-27
- deploy and test Java Forms bpel
  - deploy, 8-27
  - test, 8-28
- Deploy and Test PL/SQL BPEL
  - deploy bpel, 3-38
  - test bpel, 3-39
- Discovering and Viewing Integration Interfaces
  - overview, 2-1
  - review details, 2-6
  - review WSDL details, 2-10
  - search and view interfaces, 2-1
  - SOAP Messages, 2-19

## E

---

- Extensibility
  - addWSSecurityHeader, 11-49
  - invokeService, 11-49
  - postInvokeService, 11-48
  - preInvokeService, 11-48
  - setInputParts, 11-50

## I

---

- Integration Repository Annotation Standards
  - annotation glossary, A-112
  - business entity, A-37
  - business event, A-31
  - composite service - BPEL, A-105
  - concurrent program, A-17
  - guidelines, A-1
  - Java, A-4
  - PL/SQL, A-11
  - XML Gateway, A-19
- Invoke Web service
  - example, 11-33
- Invoke Web Services
  - Calling Back to Oracle E-Business Suite With Web Service Responses, 11-28
- Invoke Web services through Oracle Workflow
  - overview, 11-1
  - Web Service Invocation Using SIF, 11-2
- Invoking Web service steps
  - Creating a receive Event, 11-17
  - Creating Invoke and Receive Events, 11-6
  - creating invoker local and error event subscriptions, 11-8

## O

---

- Oracle E-Business Suite Integrated SOA Gateway
  - component features, 1-3
  - Major Features, 1-2
  - Overview, 1-1

## S

---

- Sample Payload
  - Inbound Purchase Order, C-3
  - Supplier Ship and Debit Request, C-1
- Synchronous Request-Response
  - Orchestrate Synchronous BPEL, D-4

## T

---

- Testing Service Invocation
  - Command Lines, 11-42
  - Test Business Event Page, 11-38
  - Troubleshooting Web Service Invocation Failure
    - Concurrent Manager (CM) Tier JVM, 11-47
    - OACORE OC4J, 11-43
    - Standalone JVM, 11-47

## U

---

- Understanding SOAP Messages
  - SOAP Header for Applications Context, 2-25
  - SOAP Header for XML Gateway Messages, 2-28
  - SOAP Messages Through SOA Provider, 2-33
  - SOAP Messages Through Web Service Provider, 2-35
  - SOAP Security Header, 2-21
- use custom interfaces
  - design tasks, 10-23
  - overview, 10-22
  - run-time tasks, 10-46
- Using Business Events
  - deploy and test bpel, 5-25
  - overview, 5-1
  - using Business Events, 5-1
- Using Business Service Objects
  - deploy and test bpel, 7-24
  - overview, 7-1



- using Business Service Objects WSDL design time, 7-1
- Using Concurrent Program
  - design tasks, 6-1
  - Overview, 6-1
  - run-time tasks, 6-23
- Using Concurrent Program design tasks
  - Adding a Partner Link for File Adapter, 6-10
  - Assign activities, 6-19
  - Creating a New BPEL Project, 6-5
  - Creating a Partner Link, 6-7
  - Invoke activities, 6-15
- Using custom WSDL
  - Add an Assign activity, 10-39
  - Add an Invoke activity, 10-36
- Using Custom WSDL
  - Adding a Partner Link for File Adapter, 10-31
  - Create a New BPEL Project, 10-26
  - Create a Partner Link, 10-28
- Using Java APIs for Forms Services
  - Add a Partner Link for File Adapter, 8-11
  - Adding Assign Activities, 8-19
  - Add Invoke Activities, 8-15
  - Create a New BPEL Project, 8-6
  - Create a Partner Link, 8-7
    - using Java Forms Services, 8-1
- Using Java Forms Services
  - deploy and test bpel, 8-27
  - overview, 8-1
- Using PL/SQL
  - deploy and test bpel, 3-37
  - overview, 3-1
  - using PL/SQL WSDL, 3-1
- Using PL/SQL WSDL
  - Add an Assign activity, 3-25
  - Add an Invoke activity, 3-21
  - Adding a Partner Link for File Adapter, 3-12
  - Create a New BPEL Project, 3-6
  - Create a Partner Link, 3-8
- Using XML Gateway
  - deploy bpel, 4-52
  - overview, 4-1
  - test bpel, 4-53
  - using XML Gateway Inbound, 4-2
- using XML Gateway Inbound
  - using XML Gateway Inbound design time, 4-2
- Using XML Gateway Inbound by SOA Provider
  - Assign, 4-21
  - Creating a New BPEL Project, 4-7
  - Creating a Partner Link, 4-9, 4-12
  - Invoke, 4-18
- Using XML Gateway Inbound SOA Provider
  - Run-time tasks, 4-26
- Using XML Gateway Inbound SOA Provider Run-Time Tasks
  - deploy, 4-27, 4-28
- Using XML Gateway outbound
  - using XML Gateway outbound, 4-30
- Using XML Gateway Outbound
  - deploy and test bpel, 4-51
- Using XML Gateway outbound design task
  - Add an Assign Activity, 4-49
  - Add an Invoke Activity, 4-47
  - Add a Partner Link for File Adapter, 4-41
  - Adding a Receive Activity, 4-39
  - create a new BPEL project, 4-32
  - create a Partner Link for AQ Adapter, 4-33
  - overview, 4-30

## W

---

- Web service invocation
  - consideration, 11-54
  - Extending Seeded Java Rule Function, 11-48
  - Testing Web Service Invocation, 11-37
  - Troubleshooting Web Service Invocation Failure, 11-43
- Web Service Invocation Using SIF
  - invoking Web services, 11-30
  - message patterns, 11-3
  - metadata definition, 11-5
  - Supporting WS-Security, 11-26
  - Web Service Input Message Parts, 11-21

