

## **Oracle® E-Business Suite**

Integrated SOA Gateway Developer's Guide

Release 12.1

**Part No. E12065-02**

April 2009

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.....	1-2

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

Overview.....	2-1
Searching and Viewing Integration Interfaces.....	2-1
Reviewing Interface Details.....	2-4
Reviewing WSDL Element Details.....	2-6

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

Overview.....	3-1
Using WSDL in Creating a BPEL Process at Design Time.....	3-1
Creating a New BPEL Project.....	3-5
Creating a Partner Link for the Web Service.....	3-7
Adding a Partner Link for File Adapter.....	3-10
Adding Invoke Activities.....	3-15
Adding Assign Activities.....	3-18
Deploying and Testing a BPEL Process.....	3-25
Deploying the BPEL Process.....	3-26
Testing the BPEL Process.....	3-27

## 4 Using XML Gateway Inbound and Outbound Interfaces

<b>Overview.....</b>	<b>4-1</b>
<b>Using XML Gateway Inbound Services.....</b>	<b>4-2</b>
Using XML Gateway Inbound Services at Design Time.....	4-2
Creating a New BPEL Project.....	4-6
Creating a Partner Link.....	4-8
Adding Partner Links for File Adapter.....	4-10
Adding Invoke Activities.....	4-18
Adding Assign Activities.....	4-20
Deploying and Testing the BPEL Process at Run Time.....	4-30
Deploying the BPEL Process.....	4-30
Testing the BPEL Process.....	4-32
<b>Using XML Gateway Outbound Through Subscription Model.....</b>	<b>4-35</b>
Using a XML Gateway Outbound in Creating a BPEL Process at Design Time.....	4-36
Creating a New BPEL Project.....	4-37
Creating a Partner Link for AQ Adapter.....	4-38
Adding a Receive Activity.....	4-44
Adding a Partner Link for File Adapter.....	4-46
Adding an Invoke Activity.....	4-52
Adding an Assign Activity.....	4-54
Deploying and Testing the BPEL Process at Run Time.....	4-56
Deploying the BPEL Process.....	4-57
Testing the BPEL Process.....	4-58

## 5 Using Business Events Through Subscription Model

<b>Overview.....</b>	<b>5-1</b>
<b>Using a Business Event in Creating a BPEL Process at Design Time.....</b>	<b>5-1</b>
Creating a New BPEL Project.....	5-4
Creating a Partner Link for AQ Adapter.....	5-5
Adding a Receive Activity.....	5-12
Adding a Partner Link for File Adapter.....	5-14
Adding an Invoke Activity.....	5-20
Adding an Assign Activity.....	5-22
<b>Deploying and Testing a BPEL Process at Run Time.....</b>	<b>5-24</b>
Deploying a BPEL Process.....	5-25
Testing the BPEL Process.....	5-26

## 6 Using Concurrent Programs

<b>Overview.....</b>	<b>6-1</b>
----------------------	------------



<b>Using Concurrent Program WSDL in Creating a BPEL Process at Design Time</b> .....	6-1
Creating a New BPEL Project.....	6-4
Creating a Partner Link for the Web Service.....	6-6
Adding a Partner Link for File Adapter.....	6-9
Adding Invoke Activities.....	6-14
Adding Assign Activities.....	6-18
<b>Deploying and Testing the BPEL Process at Run Time</b> .....	6-23
Deploying the BPEL Process.....	6-23
Testing the BPEL Process.....	6-24

## 7 Using Business Service Objects

<b>Overview</b> .....	7-1
<b>Using Business Service Object WSDL in Creating a BPEL Process at Design Time</b> .....	7-1
Creating a New BPEL Project.....	7-4
Creating a Partner Link.....	7-6
Adding a Partner Link for File Adapter.....	7-9
Adding an Invoke activity.....	7-14
Adding an Assign activity.....	7-17
<b>Deploying and Testing the BPEL Process at Run Time</b> .....	7-19
Deploying the BPEL Process.....	7-19
Testing the BPEL Process.....	7-20

## 8 Using Composite Services - BPEL

<b>Overview</b> .....	8-1
<b>Viewing Composite Services</b> .....	8-2
<b>Downloading Composite Services</b> .....	8-2
<b>Modifying and Deploying BPEL Processes</b> .....	8-4

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

<b>Oracle Workflow and Service Invocation Framework Overview</b> .....	9-1
<b>Web Service Invocation Using Service Invocation Framework</b> .....	9-2
Understanding Message Patterns in WSDL.....	9-3
Defining Web Service Invocation Metadata.....	9-5
Step 1: Creating a Web Service Invoker Business Event.....	9-6
Step 2: Creating Local and Error Event Subscriptions to the Invoker Event.....	9-8
Step 3: Creating a Receive Event and Subscription (Optional).....	9-17
Understanding Web Service Input Message Parts.....	9-21
Supporting WS-Security.....	9-26
<b>Calling Back to Oracle E-Business Suite With Web Service Response</b> .....	9-28

Invoking Web Services.....	9-30
Managing Errors.....	9-36
Testing Web Service Invocation.....	9-37
Troubleshooting Web Service Invocation Failure.....	9-43
Extending Seeded Java Rule Function.....	9-48
Other Invocation Usage Considerations.....	9-54

## **A Understanding Basic BPEL Process Creation**

Overview.....	A-1
---------------	-----

## **Glossary**

## **Index**

---

# Send Us Your Comments

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

### **Part No. E12065-02**

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

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

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

Note: Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the new Applications Release Online Documentation CD available on 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 EBS integration interfaces.
- B2B, A2A and BP integrations.

This documentation assumes familiarity with Oracle Applications. 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 Applications, we suggest you attend one or more of the Oracle Applications training classes available through Oracle University.

See Related Information Sources on page x for more Oracle Applications 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 Composite Services - BPEL**
- 9 Working With Oracle Workflow Business Event System to Invoke Web Services**
- A Understanding Basic BPEL Process Creation**
- Glossary**

## Related Information Sources

This book is included on the Oracle Applications Documentation Library, which is supplied in the Release 12 Media Pack. You can download soft-copy documentation as PDF files from the Oracle Technology Network at <http://otn.oracle.com/documentation>, or you can purchase hard-copy documentation from the Oracle Store at

<http://oraclestore.oracle.com>. The Oracle Applications Release 12 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 Applications documentation, use only the latest Release 12 versions of those guides.

### **Online Documentation**

All Oracle Applications documentation is available online (HTML or PDF).

- **Online Help** - Online help patches (HTML) are available on My Oracle Support.
- **PDF Documentation** - See the Oracle Applications Documentation Library for current PDF documentation for your product with each release. The Oracle Applications Documentation Library is also available on My Oracle Support and is updated frequently.
- **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 Applications product. This information helps you convert data from your existing applications and integrate Oracle Applications data with non-Oracle applications, and write custom reports for Oracle Applications 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 Applications data.

#### **Oracle Applications 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 Applications 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 Applications CRM System Administrator's Guide**

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

#### **Oracle Applications Developer's Guide**

This guide contains the coding standards followed by the Oracle Applications

development staff. It describes the Oracle Application Object Library components needed to implement the Oracle Applications user interface described in the *Oracle Applications 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 Applications. In addition, this guide has information for customizations in features such as concurrent programs, flexfields, messages, and logging.

### **Oracle Applications Flexfields Guide**

This guide provides flexfields planning, setup, and reference information for the Oracle Applications implementation team, as well as for users responsible for the ongoing maintenance of Oracle Applications 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 Applications 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 Applications Installation Guide: Using Rapid Install**

This book is intended for use by anyone who is responsible for installing or upgrading Oracle Applications. It provides instructions for running Rapid Install either to carry out a fresh installation of Oracle Applications 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 Applications Multiple Organizations Implementation Guide**

This guide describes the multiple organizations concepts in Oracle Applications. It describes in detail on setting up and working effectively with multiple organizations in Oracle Applications.

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

This guide covers the use of OracleAS Adapter in developing integrations between Oracle applications and trading partners.

Please note that this guide is in the Oracle Application Server 10g Documentation Library.

### **Oracle Applications System Administrator's Guide Documentation Set**

This documentation set provides planning and reference information for the Oracle Applications System Administrator. *Oracle Applications 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 Applications 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 Applications System Administrator's Guide - Security* describes User Management, data security, function security, auditing, and security configurations.

### **Oracle Applications User's Guide**

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

### **Oracle Applications User Interface Standards for Forms-Based Products**

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

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

This manual contains information on implementing, administering, and developing diagnostics tests in the Oracle E-Business Suite 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 service enablement process (based on the service-oriented architecture) for both native packaged public integration interfaces and composite services (BPEL type). It also describes how to invoke Web services from Oracle E-Business Suite by employing the Oracle Workflow Business Event System; how to manage Web service security; and how to 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 Applications 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 Applications. 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 User 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 ADI 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 that you can then upload. Use this guide to implement Oracle Web ADI and for information on defining 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 Applications-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. 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.

### **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 Applications 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.

## **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 Applications Data**

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

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

Because Oracle Applications tables are interrelated, any change you make using an

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

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

---

# 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 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 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
- Enforce function security and role-based access control security to allow only authorized users to execute administrative functions
- 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

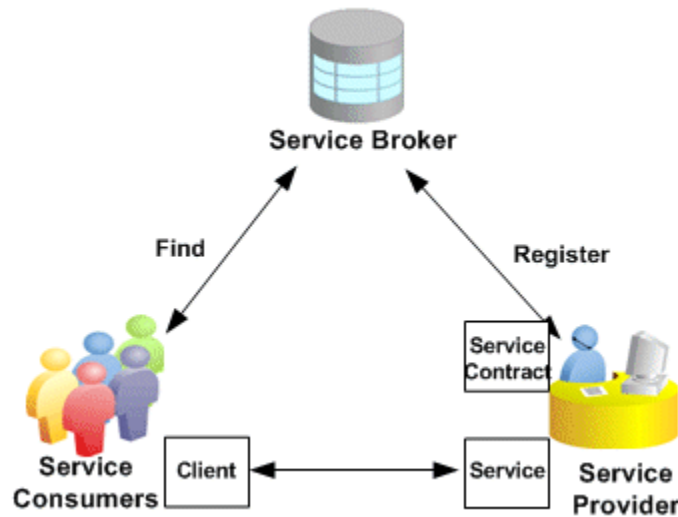
The better understand Oracle E-Business Suite Integrated SOA Gateway and its key components, this section describes some key features and the definition of each component.

### 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)
- 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 9-2.



## **SOA Monitor**

SOA Monitor is an audit and managing tool that allows all SOAP request and response messages received by SOA Provider and Web Service Provider to be logged and displayed (if the SOA Monitor 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 2-4
- Reviewing WSDL Element Details, page 2-6

## 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.

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 2-4.

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 view 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.
5. 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.
6. 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.

- 7. Select an interface type from the search result to view the interface details. You will find the appropriate interface information displayed in the Scope field.

## 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

ORACLE<sup>®</sup> Integration Repository

Home Logout Preferences Help Diagnostics

Integration Repository

View By: Interface Type

Business Event

Business Service Object

Composite - BPEL

Concurrent Program

EDI

Interface View

Java

Open Interface

PL/SQL

Applications Technology

- Application Object Library
  - Common Application Calendar
  - Workflow
  - XML Gateway
- Discrete Manufacturing
- Financial Globalizations Suite
- Financial Receivables Suite
- Financials
- Human Resources Suite
- Marketing and Sales Suite
- Order Management Suite
- Process Manufacturing
- Procurement
- Product Lifecycle Management
- Projects Suite
- Public Sector
- Sales Suite
- Service Suite
- Supply Chain Management
- University

Integration Repository >

PL/SQL Interface : Application Context APIs

Search Printable Page

Internal Name: FND\_GLOBAL

Type: PL/SQL

Product: Application Object Library

Status: Active

Online Help: See related online help

Scope: Public

Interface Source: Oracle

Full Description

Application context related APIs. The server-side package FND\_GLOBAL returns the values of system globals, such as the login/signon or "session" type of values. You should not use FND\_GLOBAL routines in your forms (that is on the client side). On the client side, most of the procedures in the FND\_GLOBAL package are replaced by a user profile option with the same (or a similar) name. You should use FND\_PROFILE routines in your forms instead.

Web Service - SOA Provider

Web Service Status: Generated

WSDL: [http://wss60066rems.us.oracle.com:8040/webservices/SOAPProvider/plsql/fnd\\_global/?wsdl](http://wss60066rems.us.oracle.com:8040/webservices/SOAPProvider/plsql/fnd_global/?wsdl)

Source Information

Source File: patch/115/sql/AFSCGBLS.pls

Source Version: 120.4.12000000.2

Source Product: FND

Procedures and Functions

Name	Internal Name	Status	Description
<a href="#">Get_Conc_Appl_ID</a>	FROG_APP_ID	Active	Returns concurrent program Application ID.
<a href="#">Get_Conc_Login_ID</a>	CONC_LOGIN_ID	Active	Returns concurrent program login ID.
<a href="#">Get_Conc_Program_ID</a>	CONC_PROGRAM_ID	Active	Returns concurrent program ID.
<a href="#">Get_Conc_Request_ID</a>	CONC_REQUEST_ID	Active	Returns concurrent Request ID.
<a href="#">Get_Login_ID</a>	LOGIN_ID	Active	Returns login ID(unique per signon).
<a href="#">Get_User_ID</a>	USER_ID	Active	Returns user id.
<a href="#">Initialize Globals</a>	APPS_INITIALIZE	Active	Sets up global variables and profile values in a database session.

**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.

The Web Service Status field also appears in the region indicating whether the service is deployed or not. If the generated Web service (with the Web Service Status value as 'Generated') has been successfully deployed, an deployed WSDL link appears in the region with the Web Service Status field marked as 'Deployed'.

- If a Web service with 'Generated' status, click the WSDL link to view the generated WSDL information.
- If a Web service with 'Deployed' status, click the deployed WSDL link to view the deployed WSDL information.

For more information on how to view WSDL file, see Reviewing WSDL Element Details, page 2-6.

**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)

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, and creating security grants. 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*

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 Applications.

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 WSDL link (or deployed 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 shows the Oracle Integration Repository interface. On the left is a tree view of the repository structure, including categories like Business Event, Business Service Object, Composite - BPEL, Concurrent Program, EDI, Interface View, Java, Open Interface, and PL/SQL. The main area displays details for the 'PL/SQL Interface: Invoice Creation'. It includes a search bar, a 'Printable Page' button, and a table of metadata: Internal Name (AR\_INVOICE\_API\_PUB), Type (PL/SQL), Product (Receivables), Status (Active), Business Entity (Receivables Invoice), and Meta Link (See OracleMetalink note 236938.1). Below this is a 'Full Description' section with text about the Invoice Creation API. At the bottom, a 'Web Service - SOA Provider' section shows the Web Service Status as 'Deployed' and the WSDL URL: [https://rws60066rems.us.oracle.com:4483/webservices/SOAProvider/plsql/ar\\_invoice\\_api\\_pub/?wsdl](https://rws60066rems.us.oracle.com:4483/webservices/SOAProvider/plsql/ar_invoice_api_pub/?wsdl).

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

The screenshot shows a Microsoft Internet Explorer browser window with the address bar displaying the WSDL URL: [https://rws60066rems.us.oracle.com:4483/webservices/SOAProvider/plsql/ar\\_invoice\\_api\\_pub/?wsdl](https://rws60066rems.us.oracle.com:4483/webservices/SOAProvider/plsql/ar_invoice_api_pub/?wsdl). The browser window title is 'https://rws60066rems.us.oracle.com:4483/webservices/SOAProvider/plsql/ar\_invoice\_api\_pub/?wsdl - Microsoft Internet Explorer'. The main content area displays the WSDL XML document, which includes definitions for the 'AR\_INVOICE\_API\_PUB' namespace, target namespaces, and a message named 'CREATE\_INVOICE\_Input\_Msg'.

**Note:** The http:// address in the new window has the exact WSDL URL information appeared in the interface details page. This address can be copied and used directly while creating a partner link for the invocation of the interface that is exposed as Web service in a BPEL process.

Detailed information about how to use WSDL file in creating a BPEL process, see each individual interface chapter described in this book.

## 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>
```

**Note:** You can import one WSDL document into another. For example, if you have established a WSDL document as a repository for Messages, you can use this import feature to across different Web services. Once you do this, you can use elements defined in the imported WSDL as you would locally defined elements.

For example, a corresponding WSDL document for the Invoice Creation API that is exposed as a Web service appears in a new window.

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.

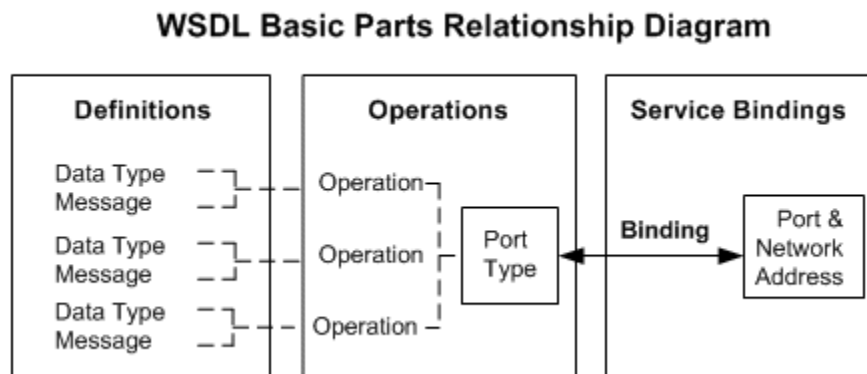
```
<definitions name="AR_INVOICE_API_PUB"
targetNamespace="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_in
voice_api_pub/"
xmlns:tns="http://xmlns.oracle.com/apps/ar/soapprovider/plsql/ar_invoice_
api_pub/"
xmlns="http://schemas.xmlsoap.org/wSDL/"
xmlns:soap="http://schemas.xmlsoap.org/wSDL/soap/"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



## 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 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\_SIN.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://rws60066rems.us.oracle.com:4483/webservices/SOAP
rovider/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://rws60066rems.us.oracle.com:4483/webservices/SOAP
rovider/plsql/ar_invoice_api_pub/APPS_XX_BPEL_CREATE_SINGLE_INVOICE_AR_I
NVOICE_API_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 Name, Responsibility Application Name, Security Group, and NLS Language complex types listed under the "SOAHeader" as shown below are used in passing values that may be required to embed application context into SOAP envelopes for Web service authorization.

For more information, see Other Web Service Input Message Parts, page 9-25.

```

...
<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="ResponsibilityName" minOccurs="0" type="string"/>
                <element name="ResponsibilityApplName" minOccurs="0"
type="string"/>
                <element name="SecurityGroupName" minOccurs="0"
type="string"/>
                <element name="NLSLanguage" minOccurs="0" type="xsd: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 9-21.

## 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.

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 Name, Responsibility Application Name, Security Group, and NLS Language 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.



```

<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://rws60066rems.us.oracle.com:4483/webservices/SOAPProvi
der/plsql/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://rws60066rems.us.oracle.com:4483/webservices/SOAPProvi
der/plsql/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>

```

## 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://rws60066rems.us.oracle.com:4483/webservices/SOAPProvide
r/plsql/ar_invoice_api_pub/" />
  </port>
</service>
```

---

## 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 Applications. APIs are stored procedures that let you update or retrieve data from Oracle Applications.

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.

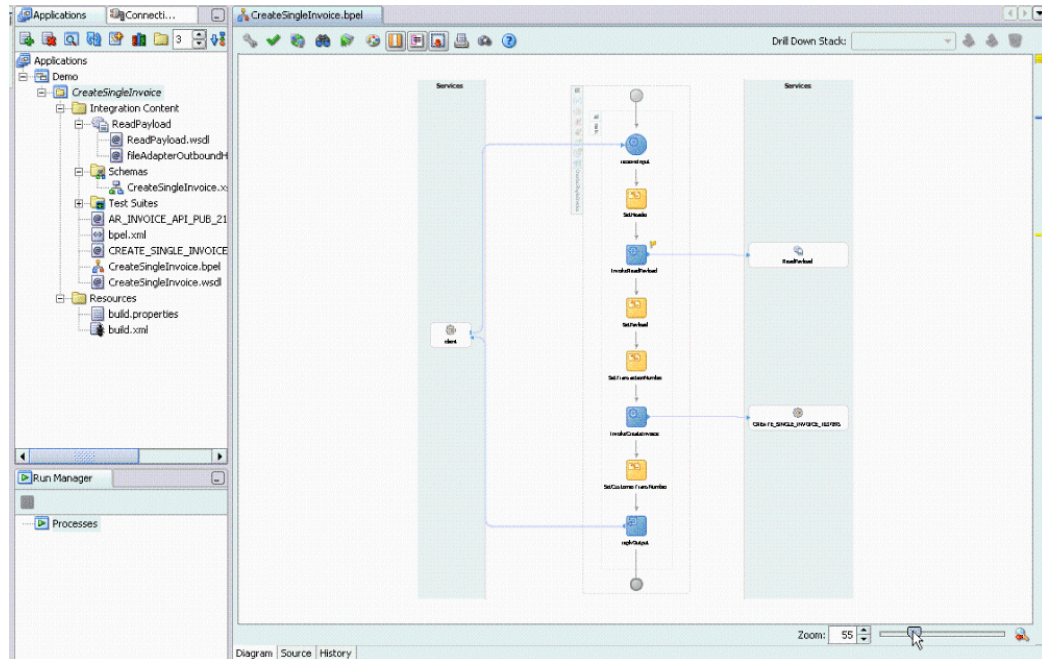
To better understand how each individual Web service represented in WSDL URL can be used in inserting or updating application data, detailed design-time and run-time tasks in creating and deploying a BPEL process are discussed in this chapter.

### Using WSDL in Creating a BPEL Process at Design Time

#### BPEL Process Scenario

Take PL/SQL Invoice Creation API `AR_INVOICE_API_PUB` as an example to explain the BPEL process creation.

When an invoice creation request is received, the invoice creation information including payload and invoice header will be read and passed to create a single invoice. Once the invoice is created, the transaction number will then be returned to the requestor.



After deploying the BPEL process, you should find an invoice is created in the Oracle Applications. The invoice number (transaction number) should be the same as the payload input value.

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

Before performing the design-time tasks for PL/SQL Web services, 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 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 AR\_INVOICE\_API\_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, click **Deploy** in the Web Service - SOA Provider region of the interface details page to deploy the service.

Once the service is successfully deployed, a confirmation message appears on top of the interface details page.

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



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 embed application context into SOAP envelopes for Web service authorization. These SOAHeader elements are *Responsibility Name*, *Responsibility Application Name*, *Security Group Name*, and *NLS Language*.

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 *Assign a SOAHeader Activity*, page 6-21.

### 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 3-5

Use this step to create a new BPEL project called `CreateSingleInvoice.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 3-7

Use this step to create an invoice in Oracle Applications by using the Invoice Creation API `AR_INVOICE_API_PUB` exposed as Web service.

3. Add a Partner Link for File Adapter, page 3-10

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

4. Add Invoke activities, page 3-15

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 `CreateSingleInvoice` partner link to initiate the single invoice creation with payload and transaction details received from the Assign activities.

5. Add Assign activities, page 3-18

Use this step to configure Assign activities in order to pass invoice header details, payload information and transaction number to appropriate Invoke activities to facilitate single invoice creation. At the end, pass the transaction 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 A-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, 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; for example, `CreateSingleInvoice`.

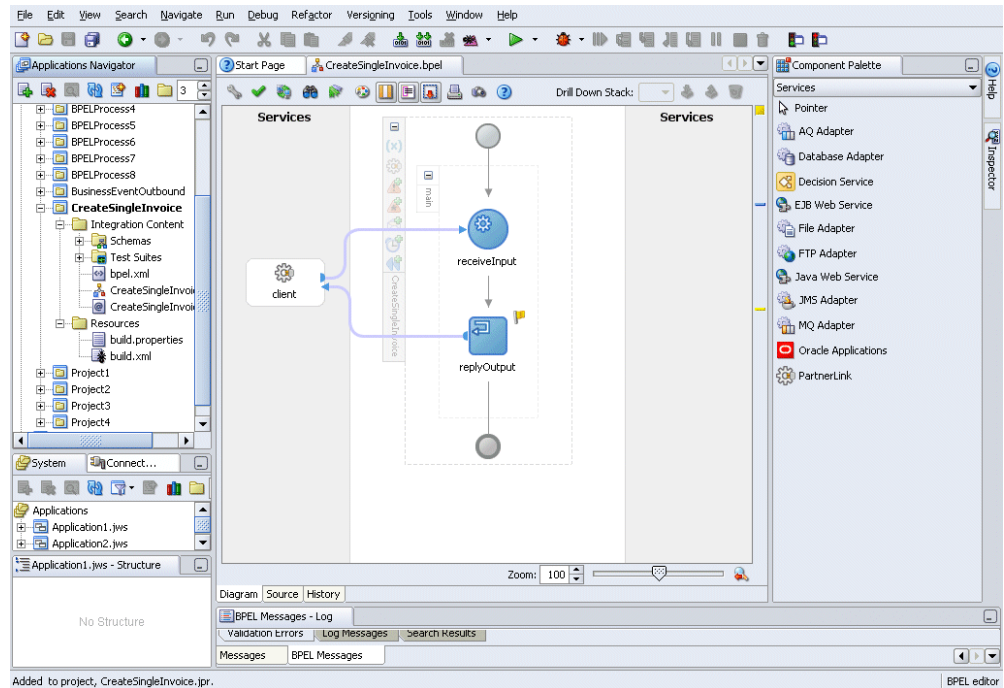
**Note:** SOA Provider does not support service creation for PL/SQL stored procedures or packages which have '\$' character in parameter type names. The presence of \$ in the name would cause the XSD generation to fail.

8. From the Template list, select **Synchronous BPEL Process**, 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, `CreateSingleInvoice.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 `Create_Single_Invoice`.

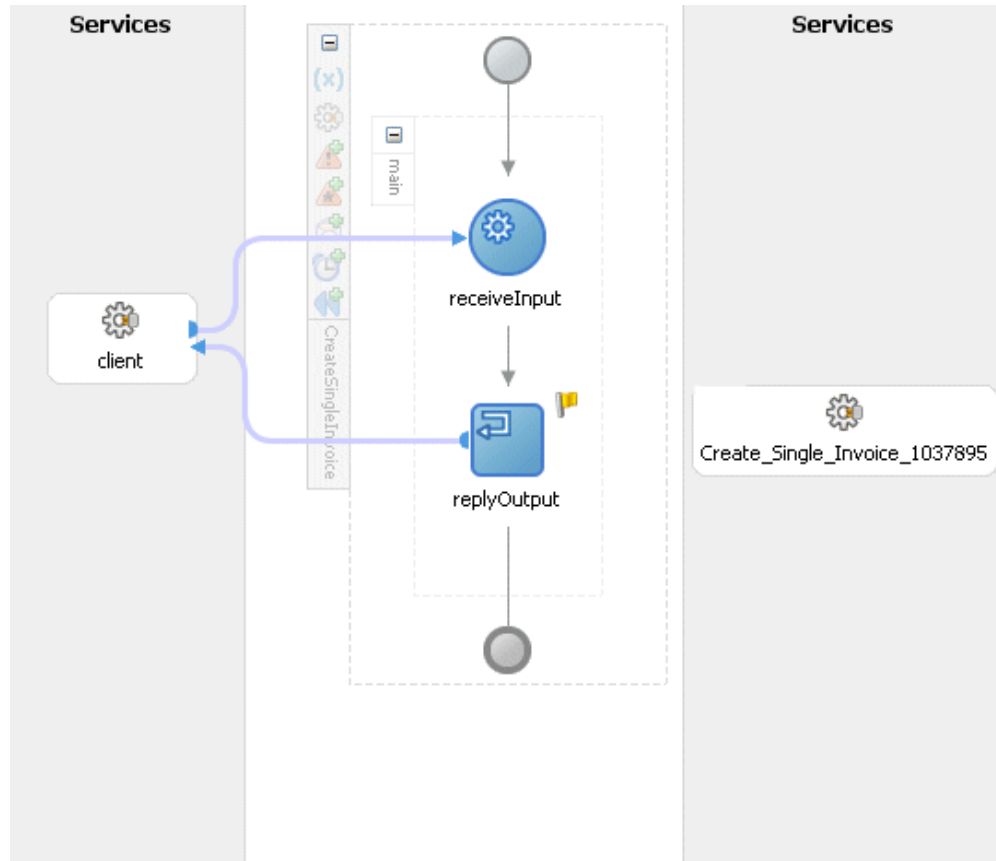
**To create a partner link for `Create_Single_Invoice` Web service:**

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 `AR_INVOICE_API_PUB` 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 value populated automatically.

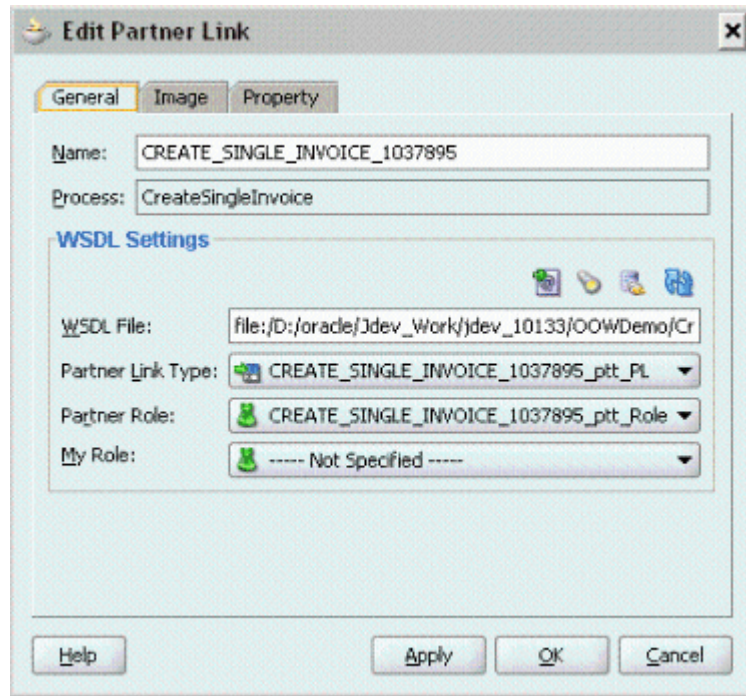
The partner link is created with the required WSDL settings, and is represented in the BPEL project by a new icon in the border area of the process diagram.

#### 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

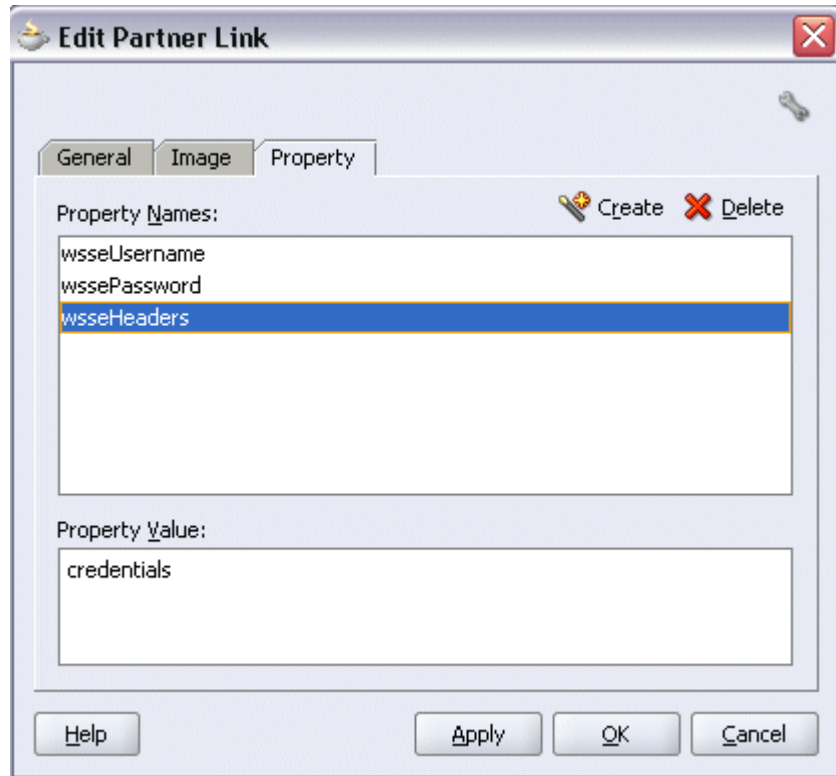


5. Select the Property tab and click the **Create Property** icon to select the following properties from the property name drop-down list in order to pass the security header along with the SOAP request:

- wsseUsername  
Specify the username 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; for example, `ReadPayload`. 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

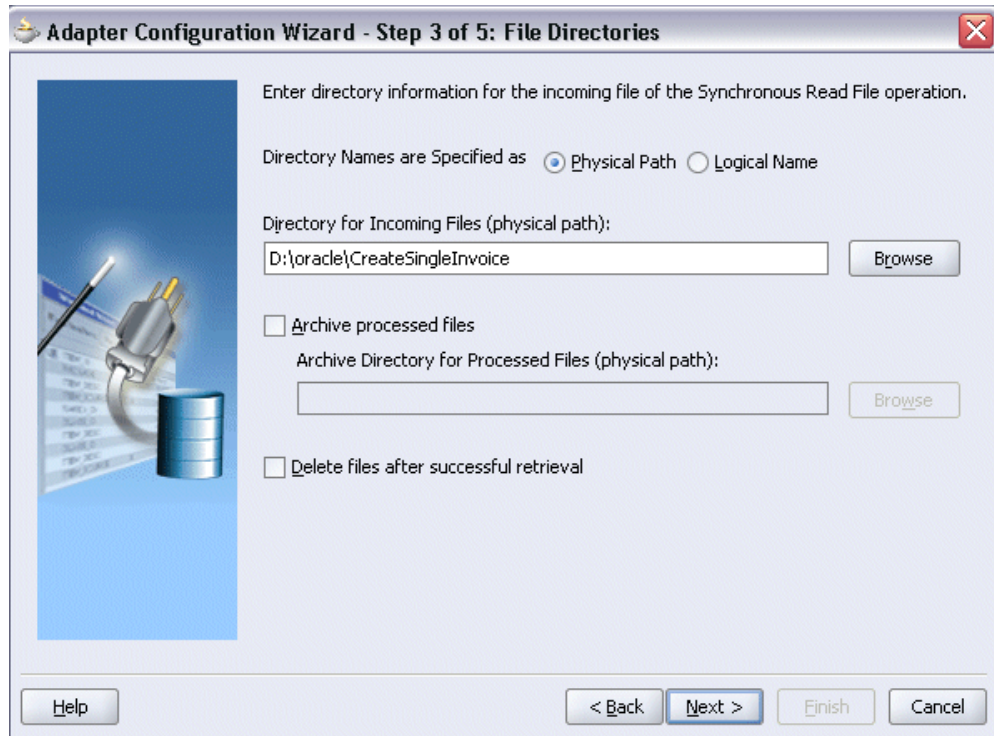
Operation Name:

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



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):

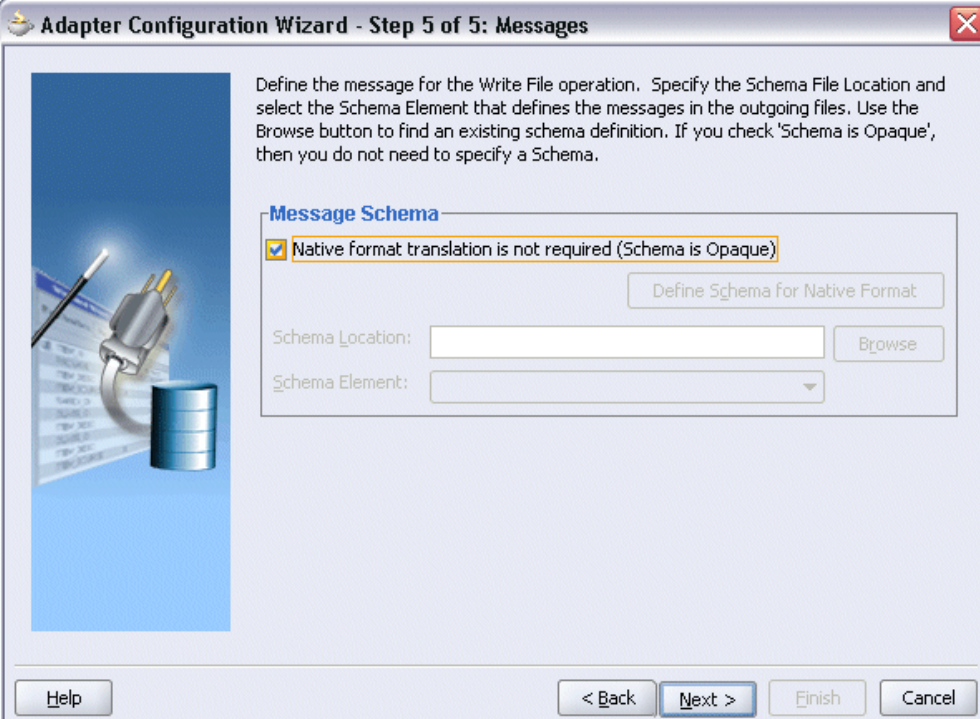
☐ Archive processed files  
Archive Directory for Processed Files (physical path):

☐ Delete files after successful retrieval

6. Select **Physical Path** radio button and 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 'Input.xml'. Click **Next**. The Messages dialog box appears.
8. Select **Native format translation is not required (Schema is Opaque)** check box.



### Specifying Message Schema



The image shows a screenshot of the 'Adapter Configuration Wizard - Step 5 of 5: Messages' dialog box. On the left is a vertical panel with a blue background and an illustration of a database cylinder and a cable. The main area contains instructions: '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 is a 'Message Schema' section with a checked checkbox 'Native format translation is not required (Schema is Opaque)'. To the right of the checkbox is a disabled button 'Define Schema for Native Format'. Below are two fields: 'Schema Location:' with a text box and a 'Browse' button, and 'Schema Element:' with a dropdown menu. At the bottom are buttons for 'Help', '< Back', 'Next >', 'Finish', 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

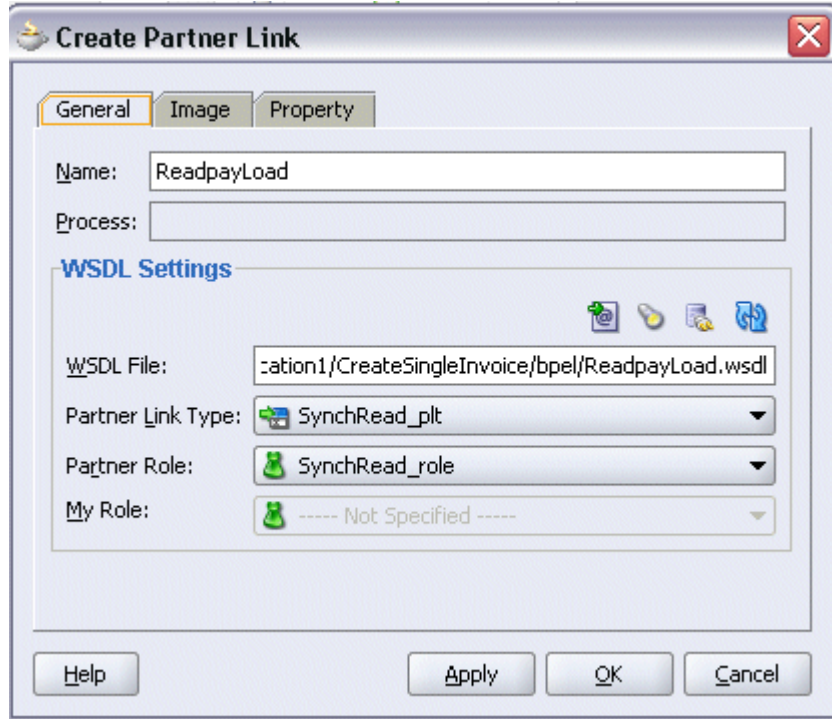
Schema Location:  Browse

Schema Element:

Help < Back Next > Finish Cancel

9. Click **Next**, 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:** cation1/CreateSingleInvoice/bpel/ReadpayLoad.wsdl
  - Partner Link Type:** SynchRead\_plt (dropdown menu)
  - Partner Role:** SynchRead\_role (dropdown menu)
  - My Role:** ----- Not Specified ----- (dropdown menu)

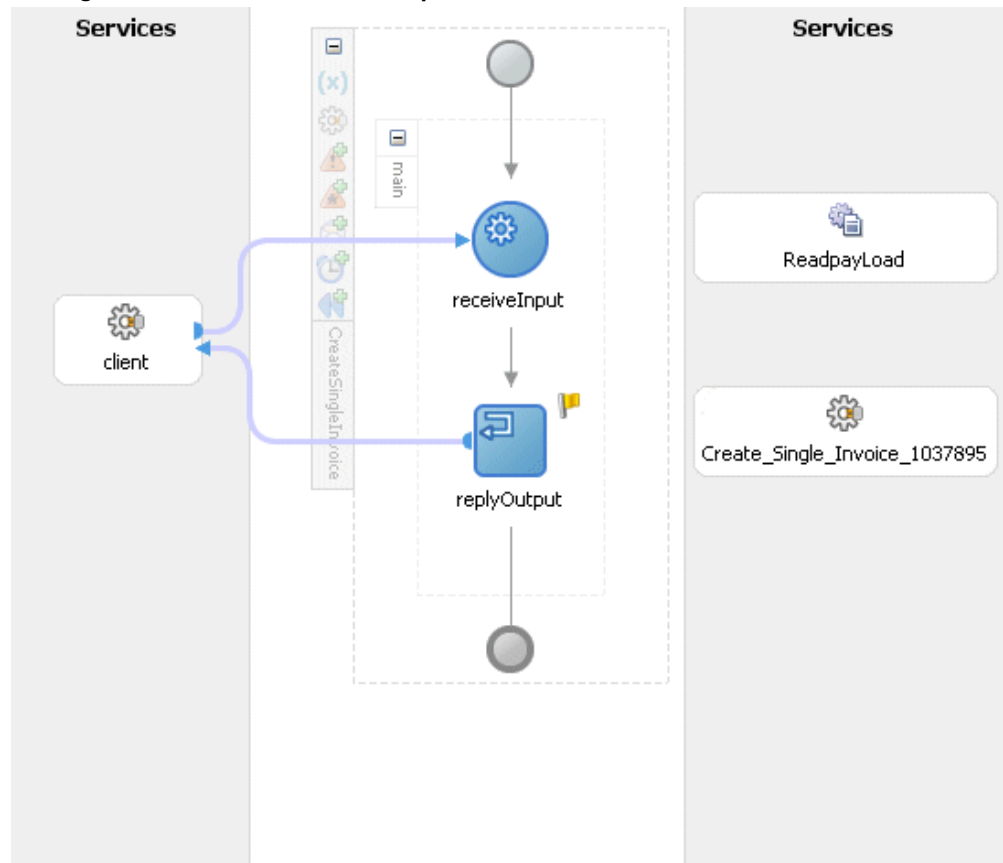
At the bottom of the dialog are four buttons: 'Help', 'Apply', 'OK', and 'Cancel'.

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 ReadPayload Partner Link appears in the following BPEL process diagram:



### Adding the Partner Link for File Adapter



## Adding Invoke Activities

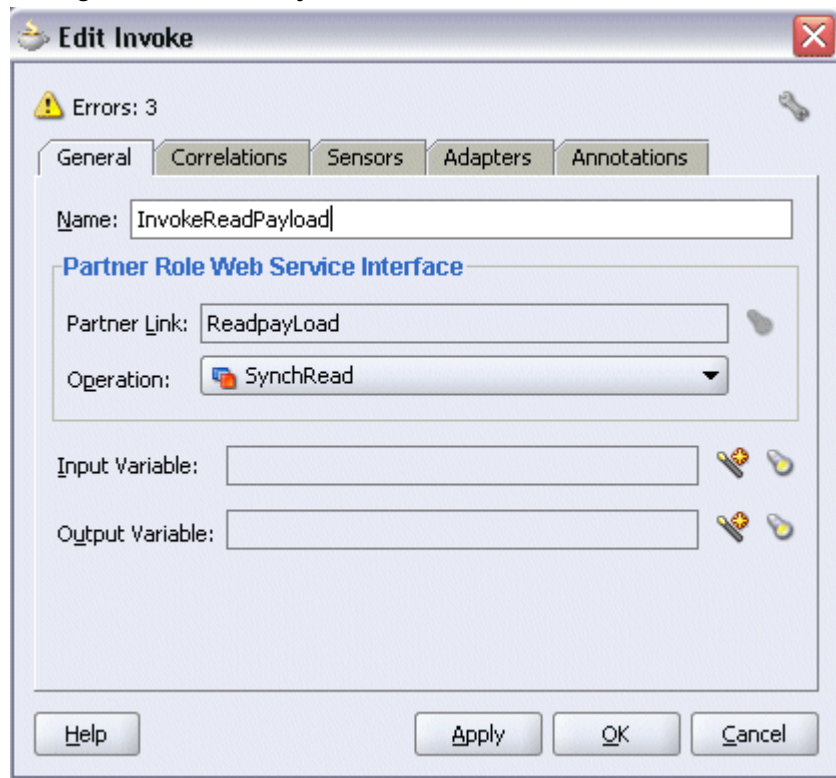
This step is to configure two Invoke activities:

- Read invoice creation details that is passed from the first Assign activity using ReadPayload partner link for File Adapter.
- Send the payload and transaction details received from the Assign activities to create a single invoice by using the CreateSingleInvoice partner link.

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

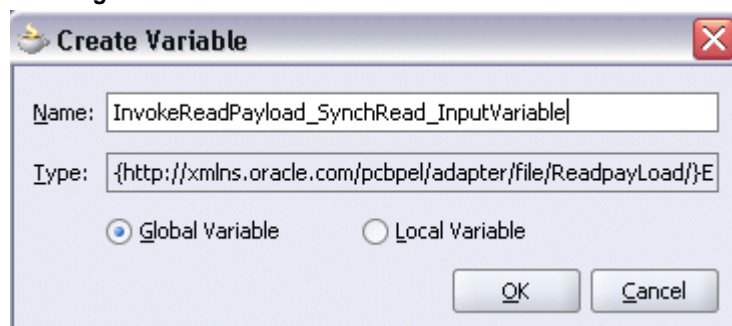
1. In JDeveloper BPEL Designer, drag and drop the **Invoke** activity from the Component Palette into the Activity box of the process diagram, between the **Receive** and **Reply** activities.
2. Link the Invoke activity to the ReadPayload service. The Invoke activity will send invoice data to the partner link. The Edit Invoke dialog box appears.

### Editing the Invoke Activity



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

### Creating a Variable

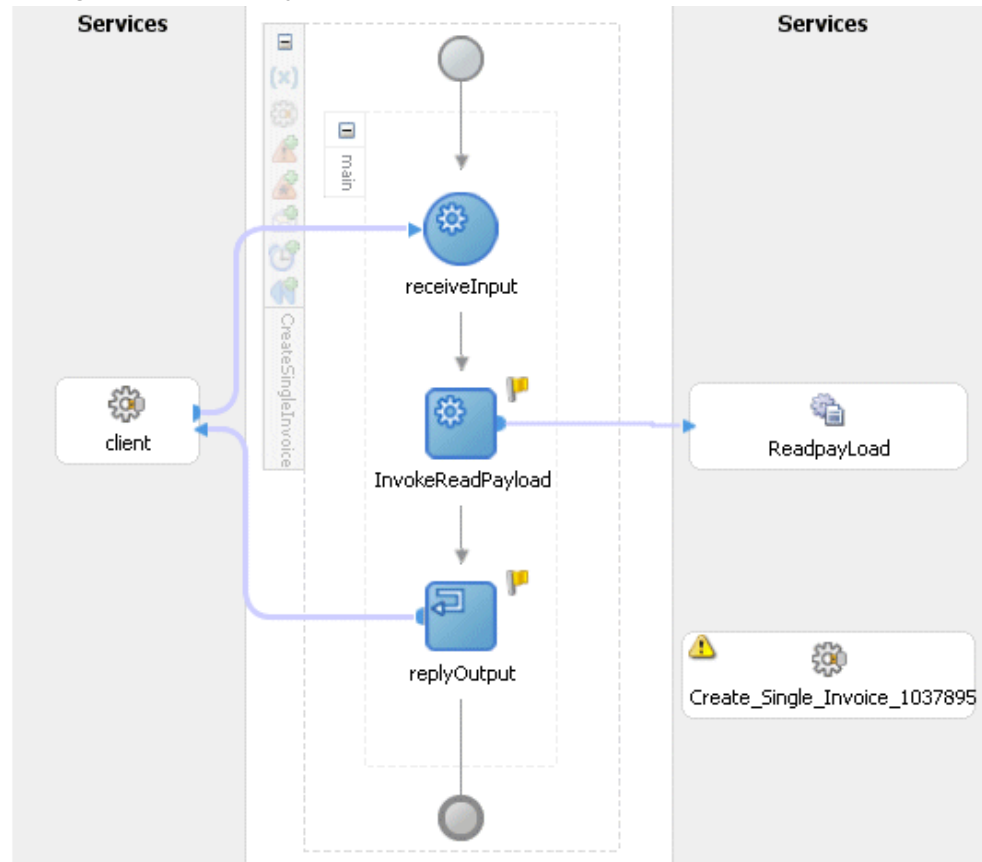


4. Select **Global Variable**, then enter a name for the variable. You can also accept the default name. Click **OK**.

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**

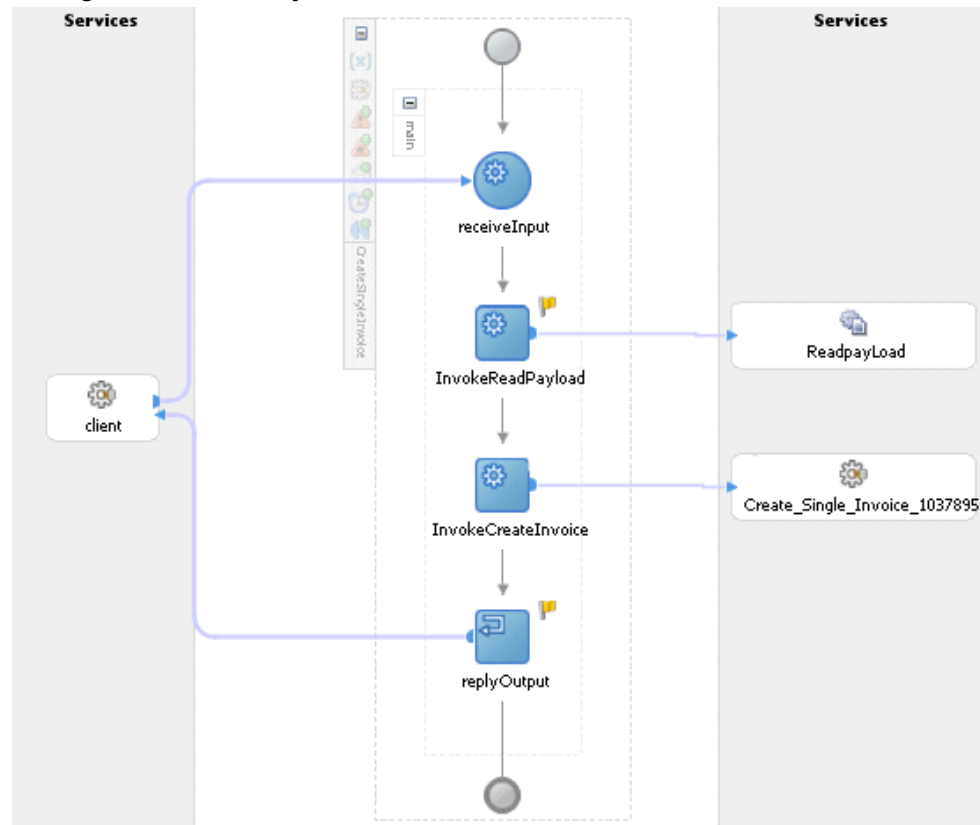


#### **To add an Invoke activity for CreateSingleInvoice Partner Link:**

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 `CreateSingleInvoice` 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 'InvokeCreateInvoice' and then 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 invoice header details with the information received from the dummy Receive activity to the ReadPayload Invoke activity.

**Note:** You also need to populate certain variables in the BPEL process for SOAHeader elements to pass values that may be required to embed application context into SOAP envelopes for Web service authorization. These SOAHeader elements are *Responsibility Name*, *Responsibility Application Name*, *Security Group Name*, and *NLS Language*.

Detailed information on how to set SOAHeader for the SOAP request, see *Assign a SOAHeader Activity*, page 6-21.

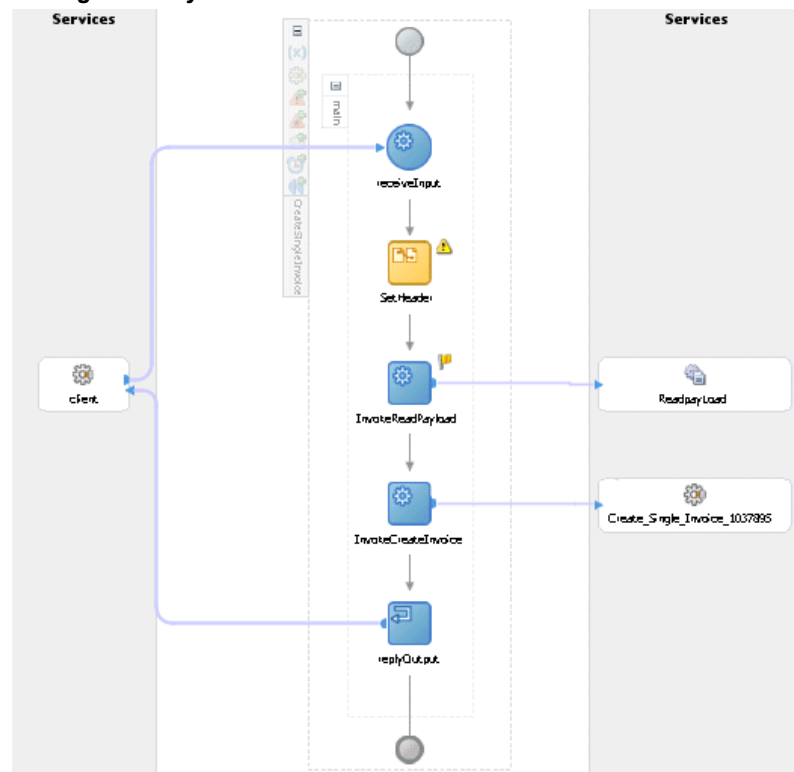
2. To pass the payload information to the InvokeCreateInvoice Invoke activity.

3. To pass the transaction number information to the `InvokeCreateInvoice` `Invoke` activity.
4. To pass the customer transaction number information back to the dummy `Reply` activity.

To add the first **Assign** activity to pass invoice header details to the `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 **Invoke** activity.

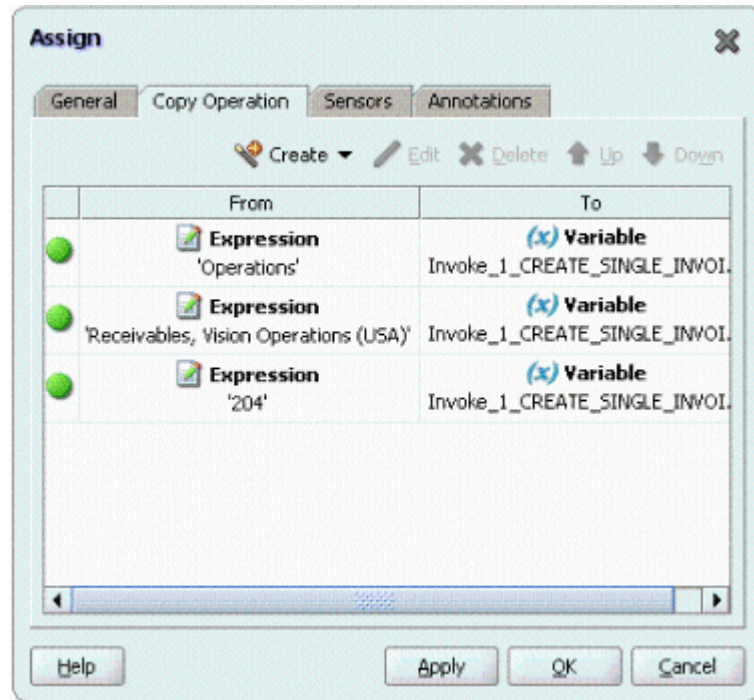
#### Adding an Assign Activity



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**, 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, then enter 'Operation' in the Expression box.
  - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_1\_Create\_Single\_Invoice\_1037895\_InputVariable > Body > ns1:SOARrequest** and select **ns3:InputParameters**. 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, then enter 'Receivables, Vision Operations (USA)' in the Expression box.
  - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_1\_Create\_Single\_Invoice\_1037895\_InputVariable > Body > ns1:SOARrequest** and select **ns3:InputParameters**. 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, then enter '204' in the Expression box.
  - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_1\_Create\_Single\_Invoice\_1037895\_InputVariable > Body > ns1:SOARrequest** and select **ns3:InputParameters**. The XPath field should contain your selected entry.
  - Click **OK**.
8. The Edit Assign dialog box appears.

### Assign Parameters



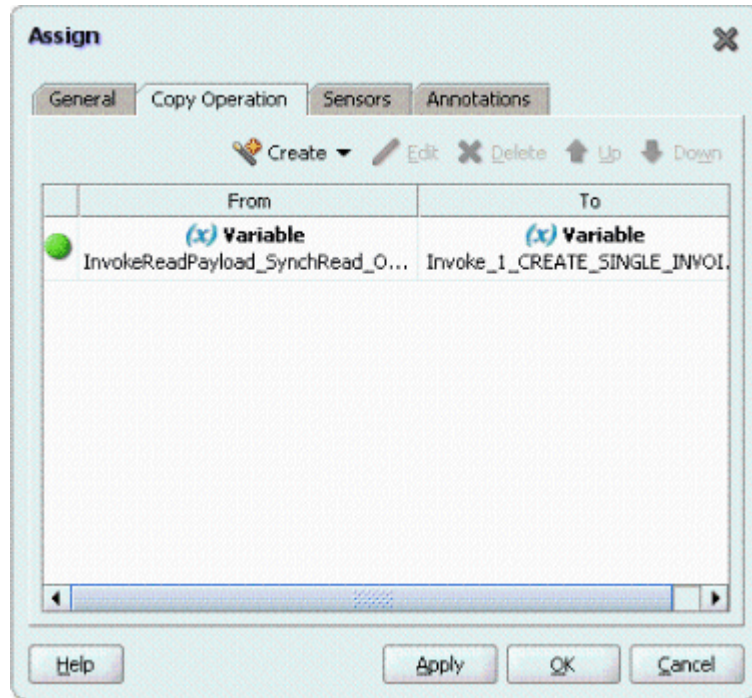
9. Click **Apply** and then **OK** to complete the configuration of the Assign activity.

**To enter the second Assign activity to pass payload information to the InvokeCreateInvoice 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 > InvokeReadPayload\_SynchRead\_OutVariable** select an appropriate value. The XPath field should contain your selected entry.
  - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_1\_Create\_Single\_Invoice\_1037895\_InputVariable > Body > ns1:SOARequest** and select **ns3:InputParameters**. 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.

**To enter the third Assign activity to pass the transaction number to the InvokeCreateInvoice Invoke activity:**

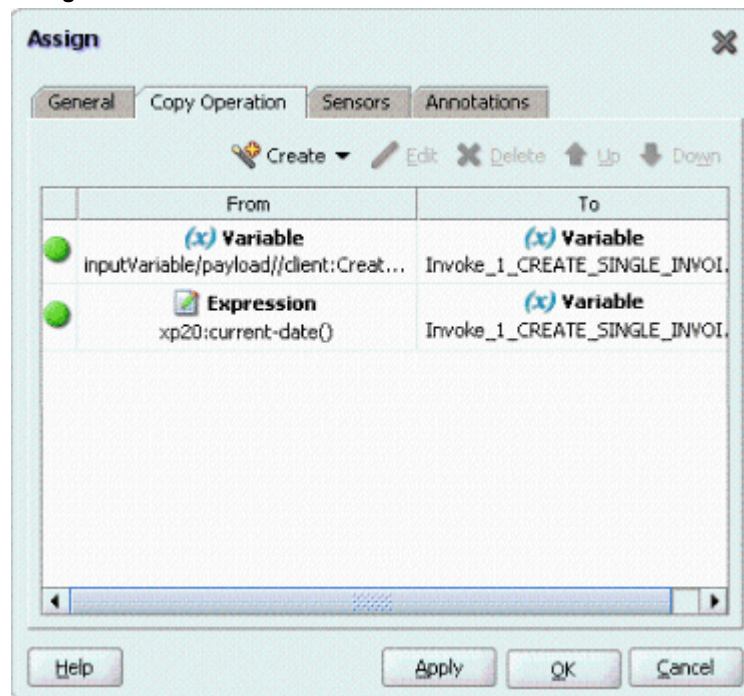
1. Add the third Assign activity by dragging and dropping the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the second **Assign** activity and the InvokeCreateInvoice **Invoke** activity.
2. Repeat Step 2 to Step 4 described in creating the first Assign activity to add the third Assign activity called 'SetTransactionNumber'.
3. Enter the following information:
  - In the From navigation tree, navigate to **Variable > Process > Variables > inputVariable > Payload > client:CreateSingleInvoiceProcessRequest** and select **client:input**. The XPath field should contain your selected entry.
  - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables >**



**Invoke\_1\_Create\_Single\_Invoice\_1037895\_InputVariable > Body > ns1:SOARrequest > ns3:InputParameters > ns3:P\_TRX\_Header\_TBL > ns3:P\_TRX\_Header\_TBL\_ITEM** and select **ns3: TRX\_NUMBER**. 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, then enter 'xp20:current-date()' in the Expression box.
  - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_1\_Create\_Single\_Invoice\_1037895\_InputVariable > Body > ns1:SOARrequest > ns3:InputParameters > ns3:P\_TRX\_Header\_TBL > ns3:P\_TRX\_Header\_TBL\_ITEM** and select **ns3: TRX\_NUMBER**. The XPath field should contain your selected entry.
  - Click **OK**.
5. The Edit Assign dialog box appears.

#### Assign Parameters

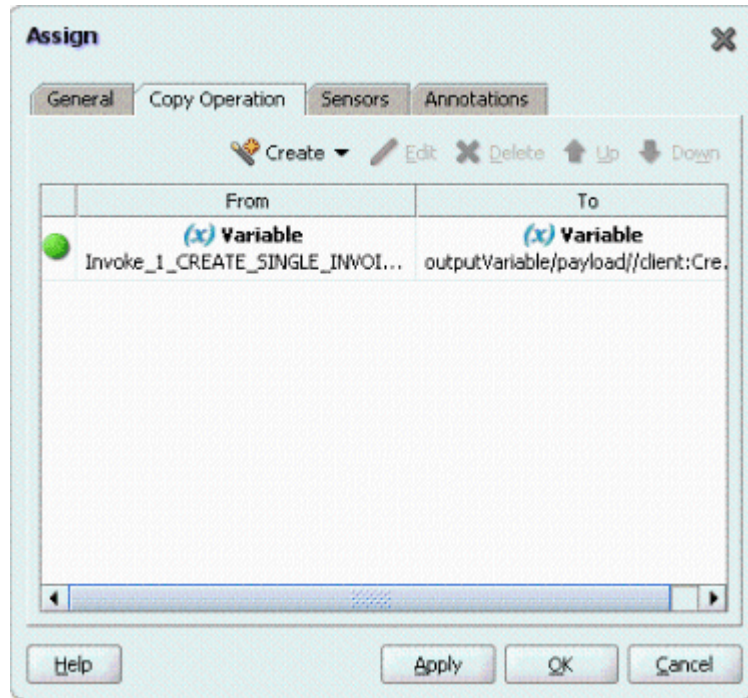


6. Click **Apply** and then **OK** to complete the configuration of the Assign activity.

**To add the fourth Assign activity to reply back customer transaction 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 `InvokeCreateInvoice` **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 'SetCustomTransNumber'.
3. Enter the following information:
  - In the From navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_1\_Create\_Single\_Invoice\_1037895\_OutputVariable > Body > ns1:SOAResponse > ns3:OutputParameters** and select **ns3:X\_CUSTOMER\_TRX\_ID**. The XPath field should contain your selected entry.
  - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > outputVariable > payload > client:CreateSingleInvoiceProcessResponse** and select **client:result**. 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.

## Deploying and Testing a 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.

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

1. Deploy the BPEL process, page 3-26

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. Test the BPEL process, page 3-27

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

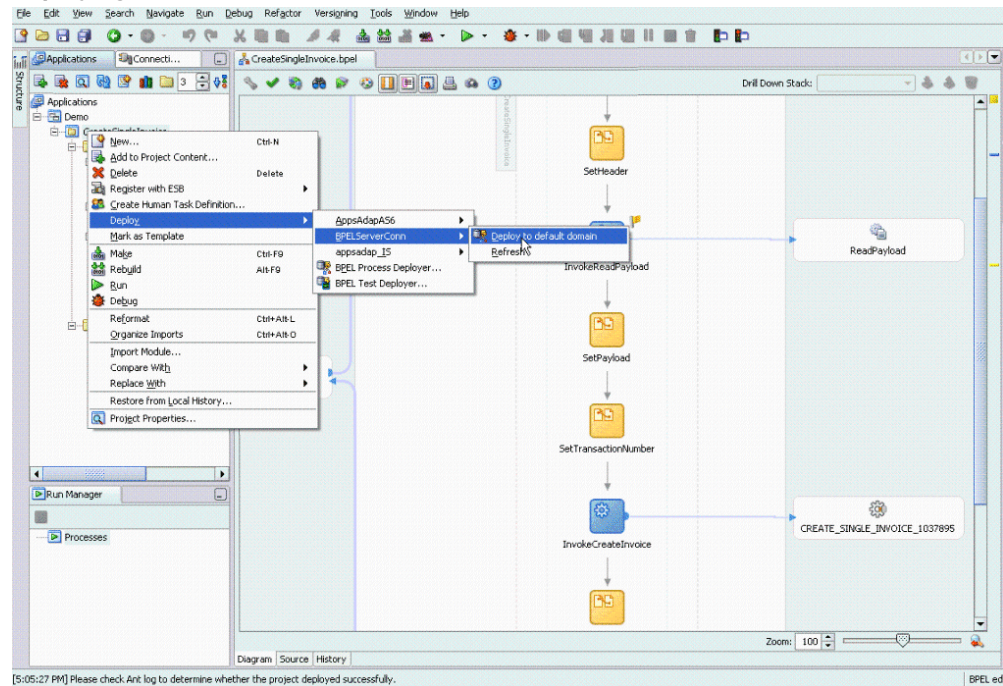
You must deploy the Create Single Invoice BPEL process (`CreateSingleInvoice.bpel`) that you created earlier before you can run it.

### To deploy the BPEL process:

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

## 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 Applications 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 BPEL Process Manager, then select **BPEL Console**. The BPEL console login screen appears.
2. Select **Default** in the **Domain** box. Enter the password for the default domain and click **Login** to access the console.



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

➔ [Goto BPEL Admin](#)

3. In the BPEL console, confirm that CreateSingleInvoice 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 ↑
CreateSingleInvoice TaskActionHandler TaskManager			
Recently Completed BPEL Process Instances (More...)			
<div>Deploy New Process</div> <div> <span>✓</span> 1 : Instance #1 of CreateSingleInvoice           <span>CreateSingleInvoice (v. 1.0)</span> <span>11/1/07 4:54:35 PM</span> </div>			
Oracle BPEL Console v10.1.3.1.0			

- Click the CreateSingleInvoice link to open the Initiate tab
- Enter Payload input field, such as INV99 and click **Post XML Message** to initiate the process.

ORACLE Enterprise Manager 10g BPEL Control			
Manage BPEL Domain   Logout   Support Logged to domain: default			
Dashboard BPEL Processes Instances Activities			
BPEL Process: CreateSingleInvoice    Version: 1.0    Lifecycle: Active Statistics: <a href="#">0 Open Instances</a>   <a href="#">1 Closed Instances</a>			
Manage	Initiate	Descriptor	WSDL
Testing this BPEL Process			
<b>Initiating a test instance</b> To create a new 'test' instance of this BPEL Process, fill this form and click on the 'Post XML Message' button.			
Operation <input type="text" value="process"/> <input checked="" type="radio"/> HTML Form <input type="radio"/> XML Source			
<div> <input checked="" type="checkbox"/> WS-Security <input type="checkbox"/> Include In Header  <input checked="" type="checkbox"/> WS-Addressing <input type="checkbox"/> Include In Header  <input checked="" type="checkbox"/> payload           </div>			
input <input type="text" value="INV99"/> xsd:string			
<small>Note: XML source view contents will not be reflected in the HTML form view</small>			
<input type="checkbox"/> Save Test <input type="checkbox"/> Perform stress test			
<input type="button" value="Post XML Message"/>			
<small>Help: XML Schema Type Formats</small>			

- Log on to Oracle Applications with the Receivables responsibility. Select Transactions link from the navigation menu to open the Transactions window.  
This is to verify an invoice is created successfully.
- Enter transaction number such as INV99 to search and verify if an invoice is

created.

Transactions (Vision Operations : USD)

**Transaction**

Source: BR Manual Date: 01-NOV-2007  
Number: INV99 GL Date: 01-NOV-2007  
Class: Invoice Currency: USD  
Type: Invoice Document Num: 100428  
Reference: Transaction: [ ]  
Legal Entity: Vision Operations [ ] Complete

**Balance Due**

Line	120.00
Tax	8.40
Freight	0.00
Charges	0.00
Total	128.40

Details Refresh

Main More Notes Commitment Reference Information

**Ship To**

Name: [ ]  
Number: [ ]  
Location: [ ]  
Address: [ ]  
Contact: [ ]

**Bill To**

Name: A. C. Networks  
Number: 1143  
Location: Provo (OPS)  
Address: 3405 East Bay Blvd.  
Contact: Provo, UT 84606 United States

**Sold To**

Name: A. C. Networks  
Number: 1143

**Paying Customer**

Name: A. C. Networks  
Number: 1143  
Location: Provo (OPS)

**Payment Details**

Receipt Method: [ ]  
Payment Method: [ ]  
Instrument Number: [ ]  
Select Instrument

Commitment: [ ]  
Payment Term: 30 Net  
Invoicing Rule: [ ]  
Due Date: 01-DEC-2007

Like Items Tag Freight Distributions Sales Credits Incomplete Open





---

# Using XML Gateway Inbound and Outbound Interfaces

## Overview

Oracle E-Business Suite Integrated SOA Gateway provides a communication infrastructure between Oracle Applications and Web consumers. Inbound and outbound XML data is exchanged between the consumers and Oracle Applications 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 allows easy integration with Oracle Applications 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 Applications 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 Applications.
- 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 Applications 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 4-30
- Using XML Gateway Outbound Through Subscription Model, page 4-35
  - Using XML Gateway Outbound Messages in Creating a BPEL Process at Design Time, page 4-36
  - Deploying and Testing a BPEL Process at Run Time, page 4-32

## 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 4-30

## 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.

After deploying the BPEL process, 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 XML Gateway Inbound Services**

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 a XML Gateway inbound interface `INBOUND:Process Purchase Order XML Transaction (ONT:TOI)`) 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, click **Deploy** in the Web Service - SOA Provider

region of the interface details page to deploy the service.

Once the service is successfully deployed, a confirmation message appears on top of the interface details page.

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**

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 shows the Oracle Integration Repository web application. On the left is a tree view of the repository structure, including categories like Business Event, Business Service Object, Concurrent Program, EDI, Interface View, Java, Open Interface, PL/SQL, Service Data Object, Web Service, and XML Gateway Map. The 'XML Gateway Map' is selected, showing sub-items like Advanced Planning, Applications Technology, XML Gateway, E-Business Suite, Financial Payables Suite, Financial Receivables Suite, Financials, Order Management Suite, Order Management, Release Management, Transportation Execution, Process Manufacturing, Procurement, Public Sector, Service Suite, Supply Chain Management, and University.

The main content area displays details for the selected interface: 'XML Gateway: INBOUND: Process Purchase Order XML Transaction'. It includes a 'Diagnostics' tab and a 'Search' button. The interface details are as follows:

Internal Name	Type	Scope
ONT-POI	XML Gateway Map	Public
Product	Interface Source	
Order Management	Oracle	
Status		
Active		
Standard		
OAG 7.2 Process_PO_007		

**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	WSDL
Deployed	<a href="http://ap0012rems.us.oracle.com:8020/webservices/SOAProvider/xmlgateway/poi/?wsdl">http://ap0012rems.us.oracle.com:8020/webservices/SOAProvider/xmlgateway/poi/?wsdl</a>

**Source Information**

Source File	Source Version	Source Product
patch/115/xml/US/ONT_3A4R_OAG72_IN.xgm	120.3	ONT

**Methods**

Select Details	Name	Internal Name	Status	Description
No Methods.				

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 context information for Oracle Applications. 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 4-20 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 Applications:

1. Log in to Oracle Applications with the responsibility of Workflow Administrator.
2. The Navigator page . Click the **Workflow Administrator Web Applications** link.
3. Click the **Workflow Manager** link under Oracle Applications Manager.
4. Click the status icon next to **Agent Listeners**.
5. Configure and schedule the **ECX Inbound Agent Listener** and the **ECX Transaction Agent Listener**. Select the listener, and select Start from the **Actions** box. Click **Go** 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 4-6

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 4-8

Use this step to create a partner link to allow the inbound message to be inserted to the Oracle Applications.

3. Add Partner Links for File Adapters, page 4-10

Use this step to add two partner links for File Adapter in order to:

- Pick up an XML file received from the trading partner to get the XML message.
- Get the transaction information for the ECX header.

4. Add Invoke activities, page 4-18

Use this step to add three Invoke activities in order to:

1. To get the XML message details that is received from the Receive activity.

2. To get the ECX Header properties details.
  3. To enqueue the purchase order information to the ECX\_INBOUND queue.
  5. Add Assign activities, page 4-20
- Use this step to create two Assign activities in order to:
1. To pass XML message obtained from the first Invoke activity to the last Invoke activity.
  2. To pass ECX header variables to the last Invoke activity as input variables in order to provide context information for Oracle Applications.

**Note:** You also need to populate certain variables in the BPEL process for SOAHeader elements to pass values that may be required to embed application context into SOAP envelopes for Web service authorization. These SOAHeader elements are *Responsibility Name*, *Responsibility Application Name*, *Security Group Name*, and *NLS Language*.

Detailed information on how to set SOAHeader for the SOAP request, see *Assign a SOAHeader Activity*, page 6-21.

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

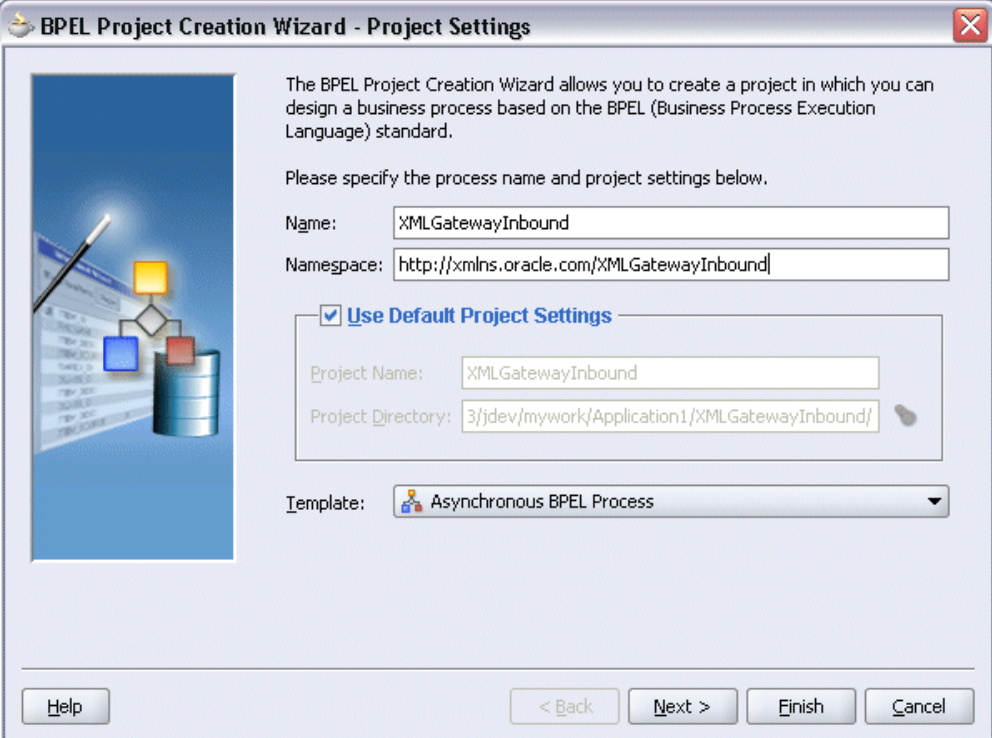
## Creating a New BPEL Project

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, 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:

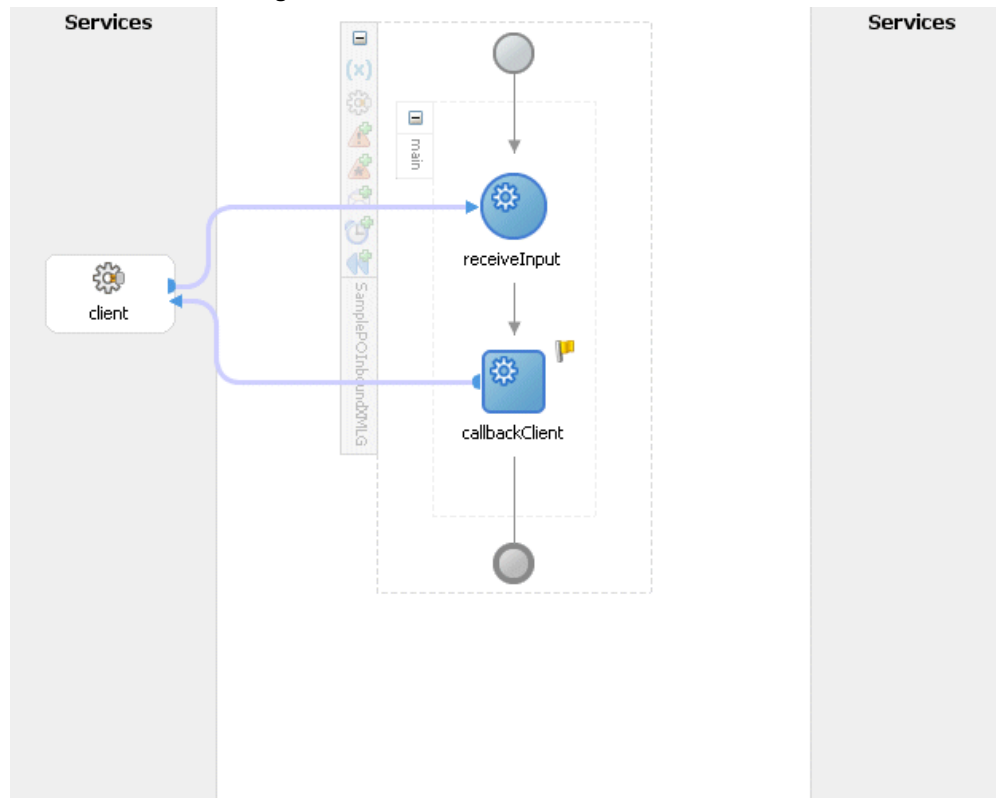
Template:

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

7. In the **Name** field, enter a descriptive name; for example, XMLGatewayInbound.
8. From the Template list, select **Asynchronous BPEL Process**, 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 `EnqueueMsg` to insert sales order data to Oracle Applications.

### To create a partner link to insert sales data to Oracle Applications:

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:TOI)` 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, Partner Link Type, and Partner Role values populated automatically. 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

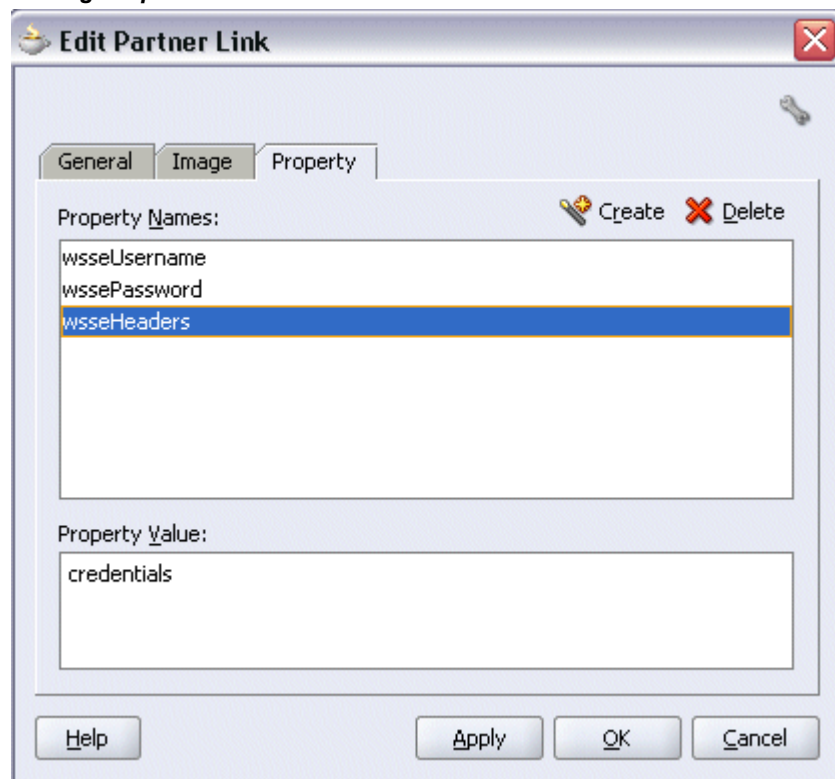
Specify the corresponding password for the username to be passed in the Property Value box.

- wsseHeaders

Enter `credentials` as the property value.

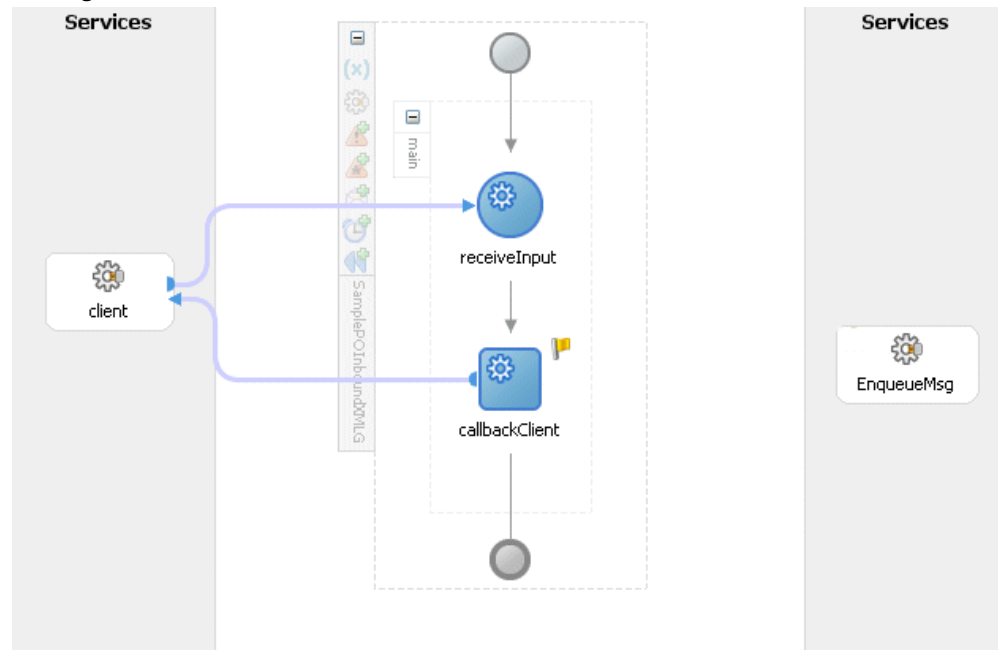
Click **Apply** to save the selected property values.

#### **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 the following two partner links for File Adapter:

1. To pick up an XML file received from the third party application to get the XML message.
2. To get the transaction information for the ECX header.

#### To add the first 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; for example, `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

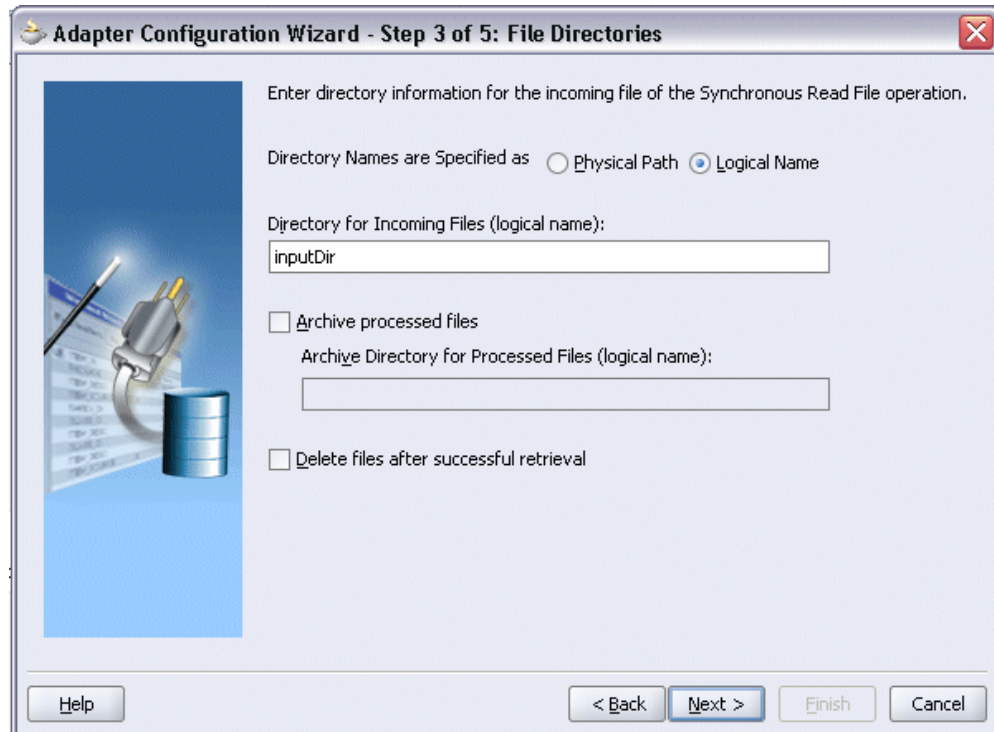
Operation Name:

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



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 (logical name):

☐ Archive processed files  
Archive Directory for Processed Files (logical name):

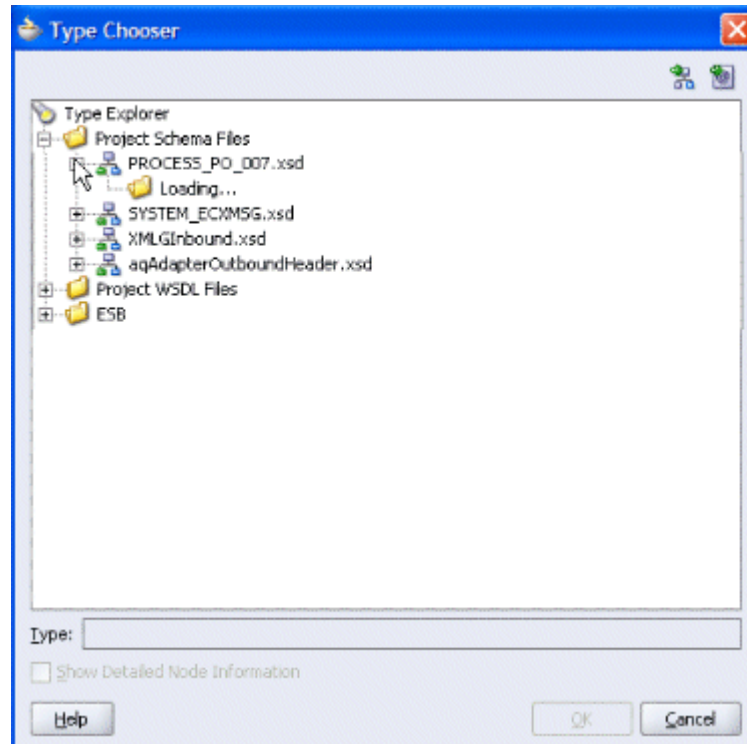
☐ Delete files after successful retrieval

Help < Back Next > Finish Cancel

6. Select **Logical Name** check box and specify the logical directory for the incoming file. 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\_xm1g.xml'. Click **Next**. The Messages dialog box appears.
8. Select **Browse** to open the Type Chooser to specify the Schema location and element.

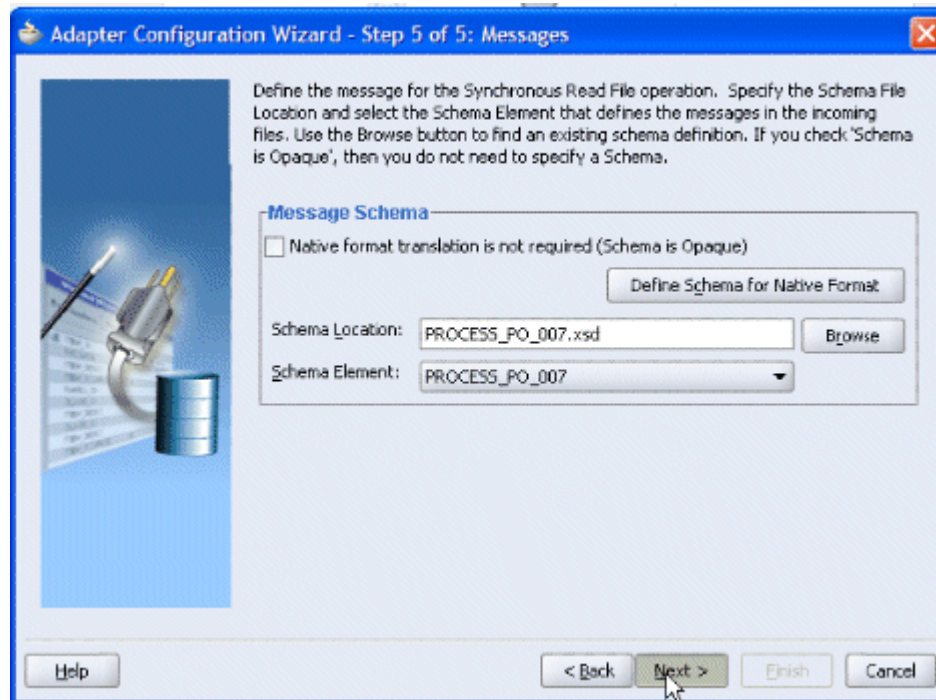
From the Type Chooser window, expand the node by clicking Project Schema Files > PROCESS\_PO\_007.xsd > PROCESS\_PO\_007.

### Selecting Schema from the Type Chooser



9. Click **OK** to populated the selected values in the Messages dialog box.

### Populating Selected Message Schema and Element

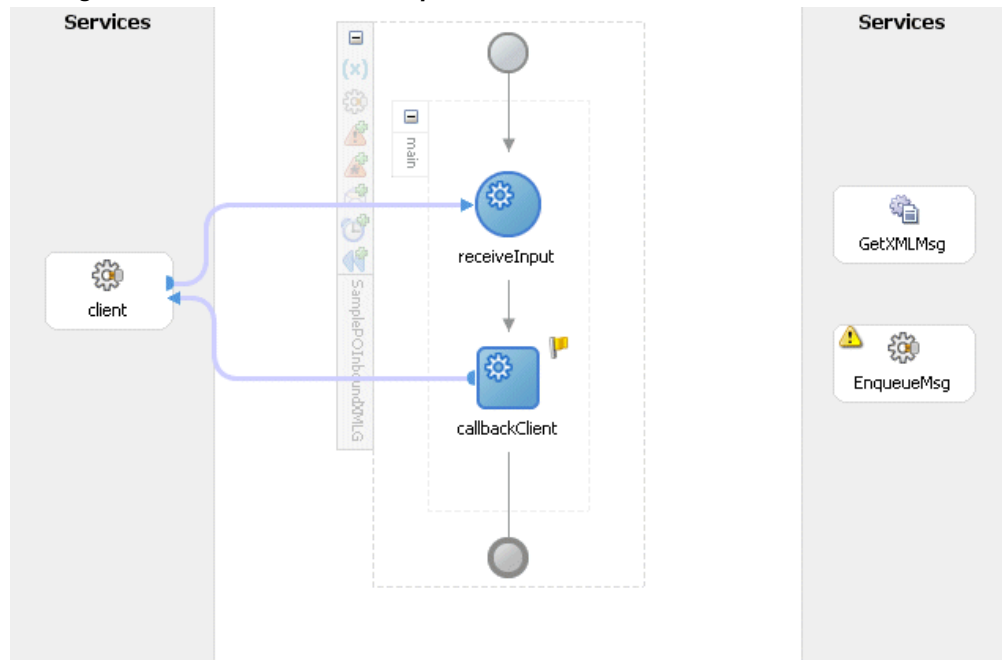


10. Click **Next**, then **Finish**. The wizard generates the WSDL file corresponding to the partner link. The main Create Partner Link dialog box appears, specifying the new WSDL file `GetXMLMsg.wsdl`.

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 following BPEL process diagram:

### Adding the Partner Link for File Adapter



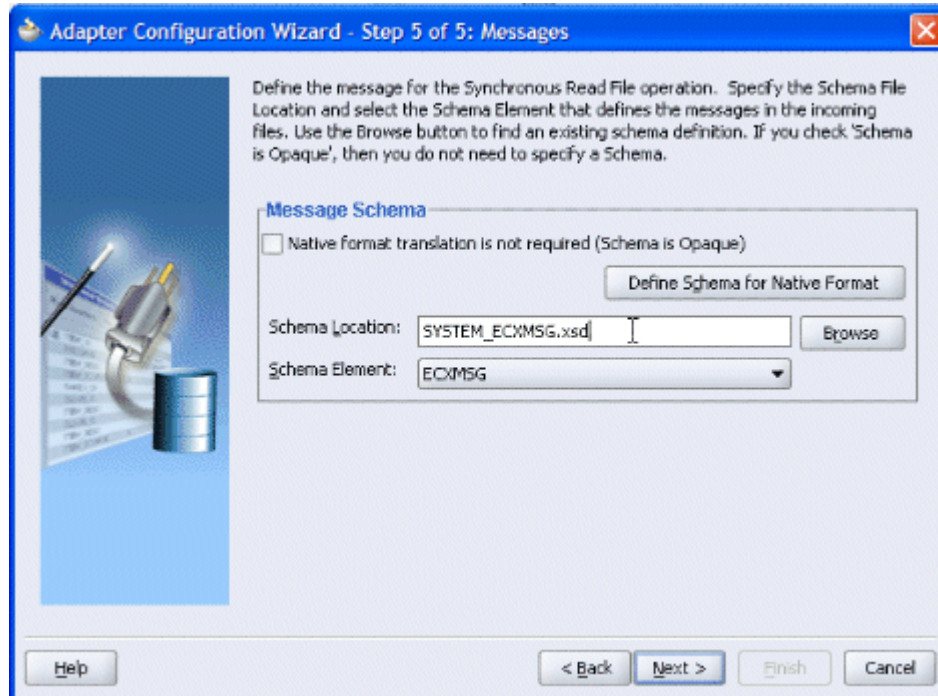
To add the second Partner Link for File Adapter to get the transaction information for the XML header:

1. Repeat Step 1 to Step 6 to add another partner link called `GetECXHeader`.
2. Enter the name of the file for the synchronous read file operation. For example, enter 'ecx\_header\_data.xml'. Click **Next**. The Messages dialog box appears.
3. Select **Browse** to open the Type Chooser to specify the Schema location and element.

From the Type Chooser window, expand the node by clicking Project Schema Files > SYSTEM\_ECXMSG.xsd > SYSTEM\_ECXMSG. Click **OK** to populated the selected schema location and element in the Messages dialog box.



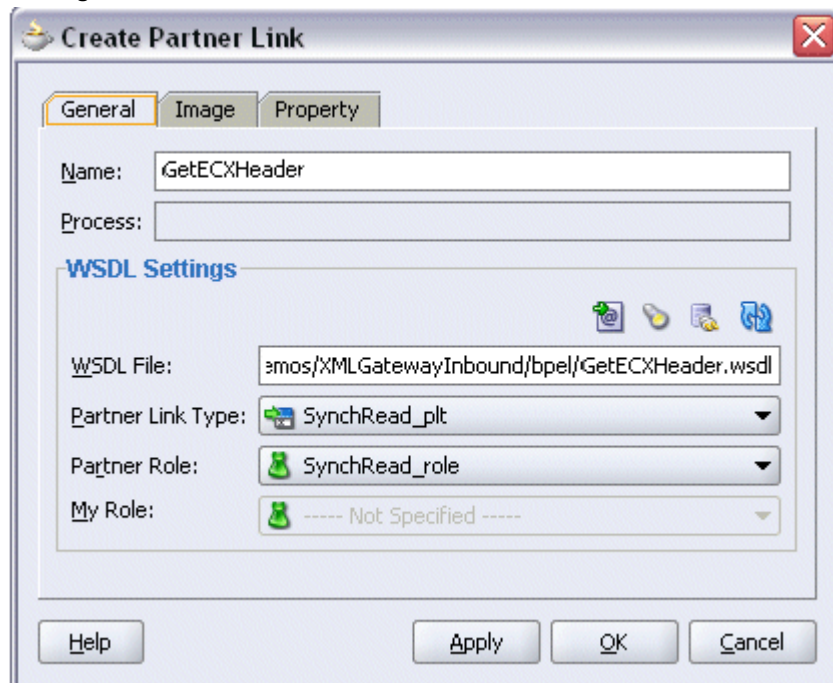
**Populated Selected Schema and Element in the Messages Dialog Box**



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



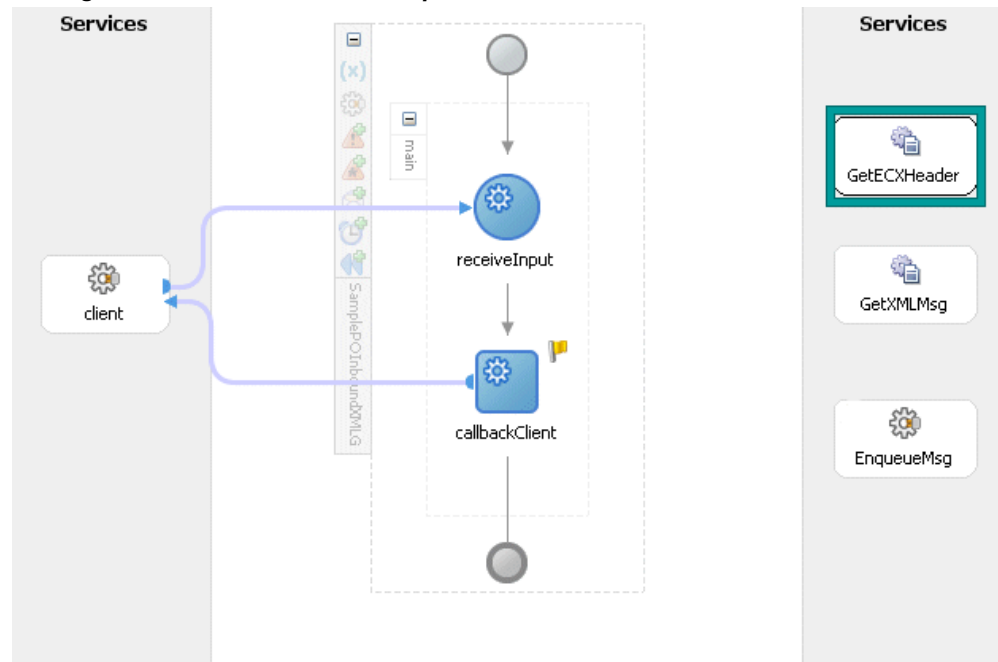
### Adding a Partner Link



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 `GetECXHeader` Partner Link appears in the following 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 get the ECX Header properties details by invoking *GetECXHeader* partner link in an XML file.
3. To enqueue the purchase order information to the ECX\_INBOUND queue by invoking *EnqueueMsg* partner link in a 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, 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**, then enter a name for the variable. You can also accept the default name. Click **OK**.
5. Enter a name for the Invoke activity, 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**, 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.

**To add the second Invoke activity for a partner link to get ECX Header properties:**

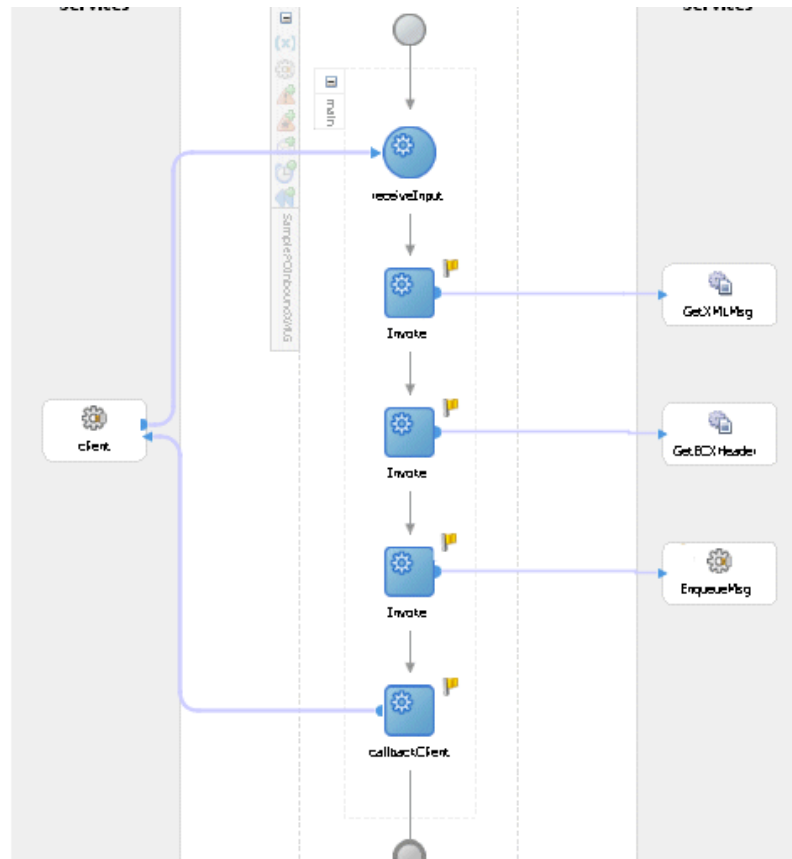
1. In JDeveloper BPEL Designer, drag and drop the second **Invoke** activity from the Component Palette into the Activity box of the process diagram, between the **Invoke** and **Callback** activities.
2. Link the Invoke activity to the `GetECXHeader` service. The Edit Invoke dialog box appears.
3. Repeat Step 3 to Step 6 described in the first Invoke activity procedure to complete the second Invoke activity.

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

1. In JDeveloper BPEL Designer, drag and drop the third **Invoke** activity from the Component Palette into the Activity box of the process diagram, between the **Invoke** and **Callback** activities.
2. Link the Invoke activity to the `EnqueueMsg` service. The Edit Invoke dialog box appears.
3. Repeat Step 3 to Step 6 described in the first Invoke activity procedure to complete the third Invoke activity.

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 last Invoke activity for enqueueing message.
2. To pass ECX header variables as input variables to the last Invoke activity in order to provide context information for Oracle Applications.

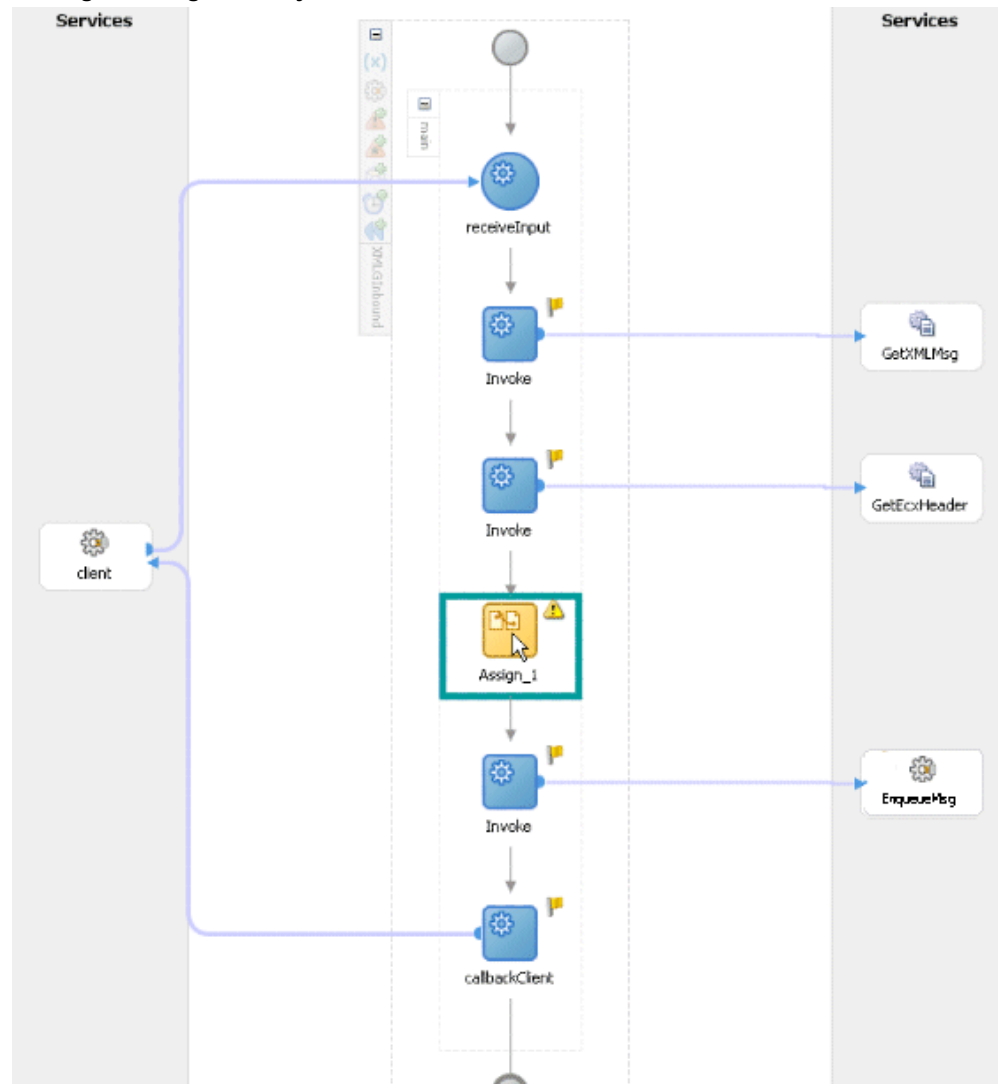
**Note:** You also need to populate certain variables in the BPEL process for SOAHeader elements to pass values that may be required to embed application context into SOAP envelopes for Web service authorization. These SOAHeader elements are *Responsibility Name*, *Responsibility Application Name*, *Security Group Name*, and *NLS Language*.

Detailed information on how to set SOAHeader for the SOAP request, see *Assign a SOAHeader Activity*, page 6-21.

**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 second and the third **Invoke** activities.

### Adding an Assign Activity

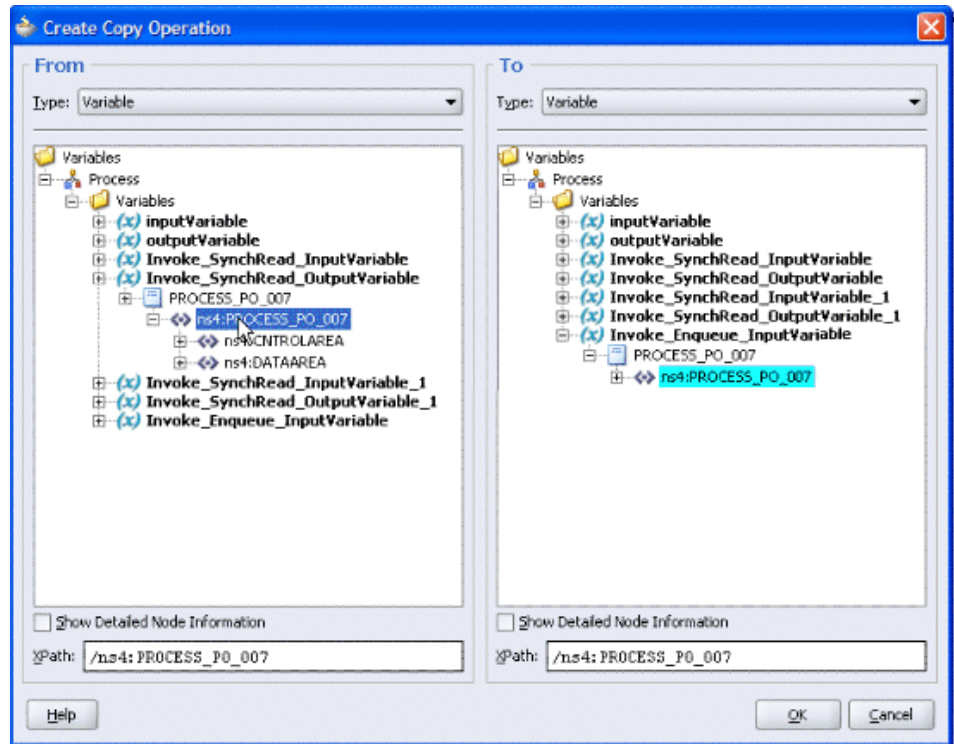


2. Double-click the **Assign** activity to access the Edit Assign dialog box.
3. Click the General tab to enter the name for the Assign activity, such as 'SetXMLMsg'.
4. On the Copy Operation tab, click **Create**, then select **Copy Operation** from the menu. The Create Copy Operation window appears.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

and select **ns4:Process\_PO\_007**.

The XPath field should contain your selected entry.

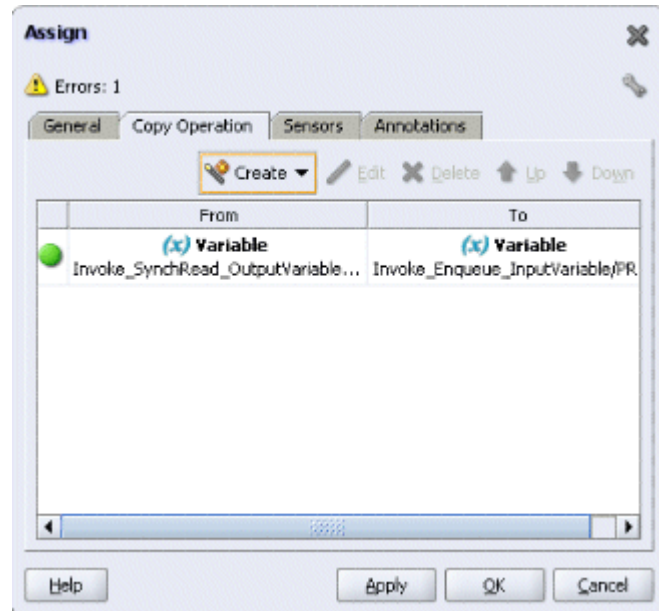
- In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_Enqueue\_InputVariable > Process\_PO\_007** and select **ns4:Process\_PO\_007**. The XPath field should contain your selected entry.



- Click **OK**.

6. The Edit Assign dialog box appears.

### Assign Parameters



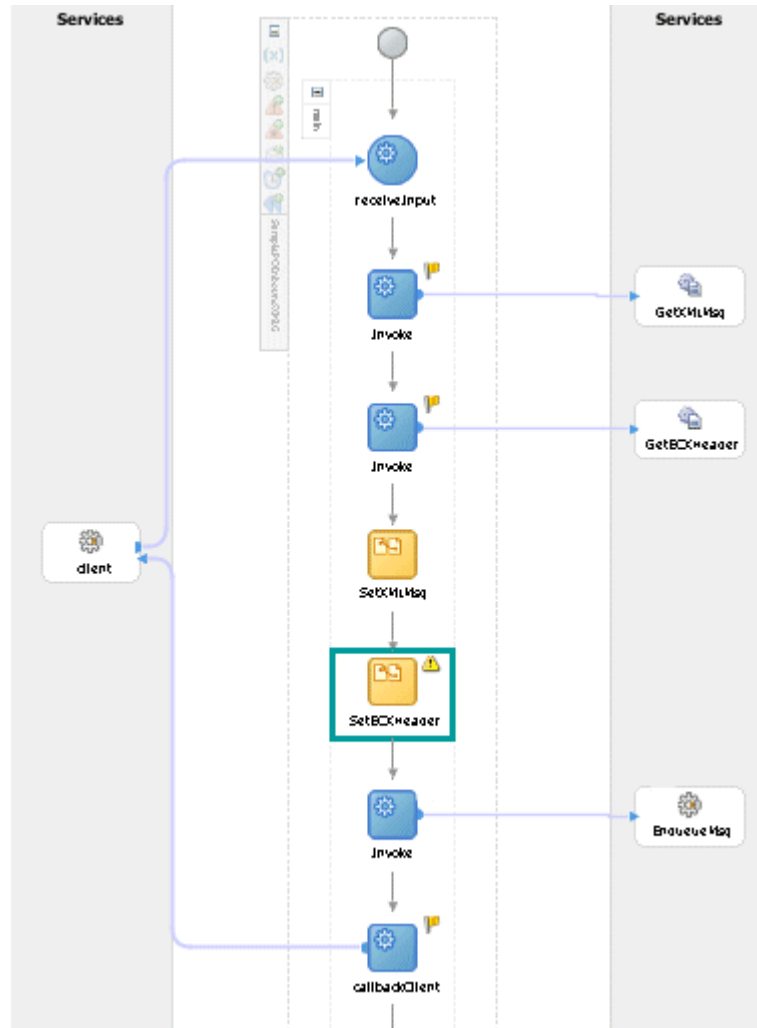
7. Click **Apply** and then **OK** to complete the configuration of the Assign activity.

**To add the second Assign activity to pass ECX header variables to the last Invoke activity for application context information:**

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 SetXMLMsg **Assign** activity and the last **Invoke** activity.

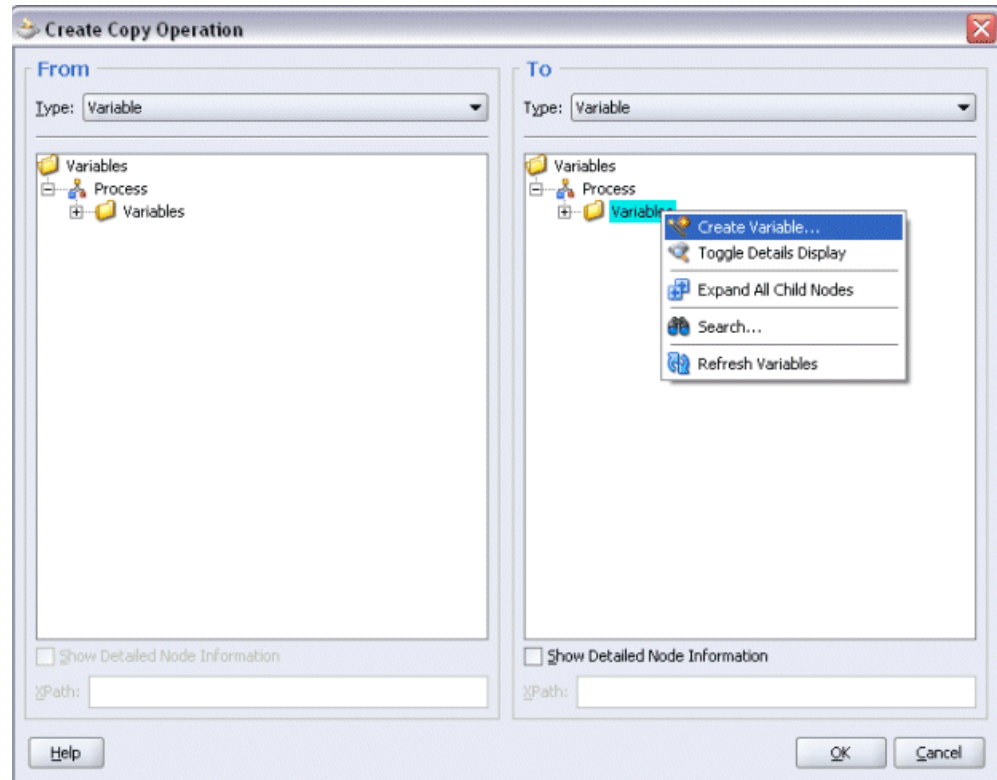


### Adding an Assign Activity



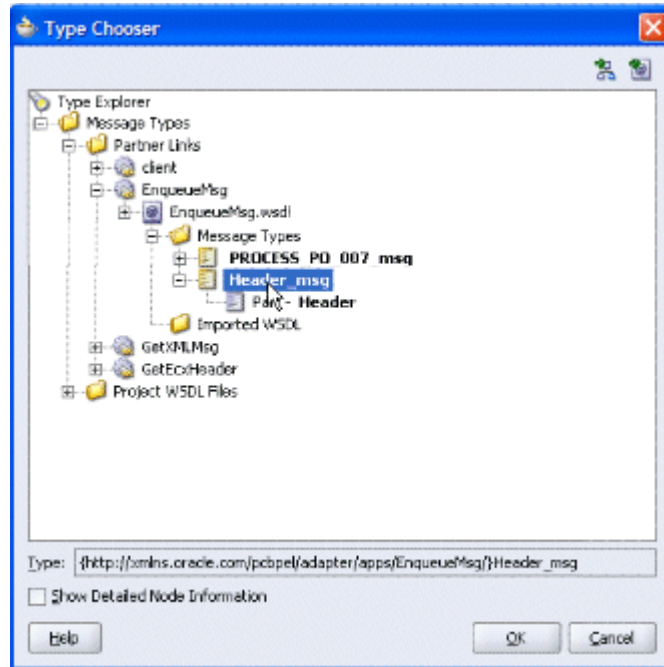
2. Repeat Step 2 to Step 4 described in creating the first Assign activity to add the second Assign activity called 'SetECXHeader'.
3. On the Copy Operation tab, click **Create**, then select **Copy Operation** from the menu. The Create Copy Operation window appears.
4. In the To navigation tree, select the **Variables** node and right-click on mouse to select Create Variable from the drop-down list.

## Creating Variables



The Create Variable dialog box opens. Select the **Message Type** check box and click the Browse icon to open the Type Chooser window.

### Declaring Header Variables

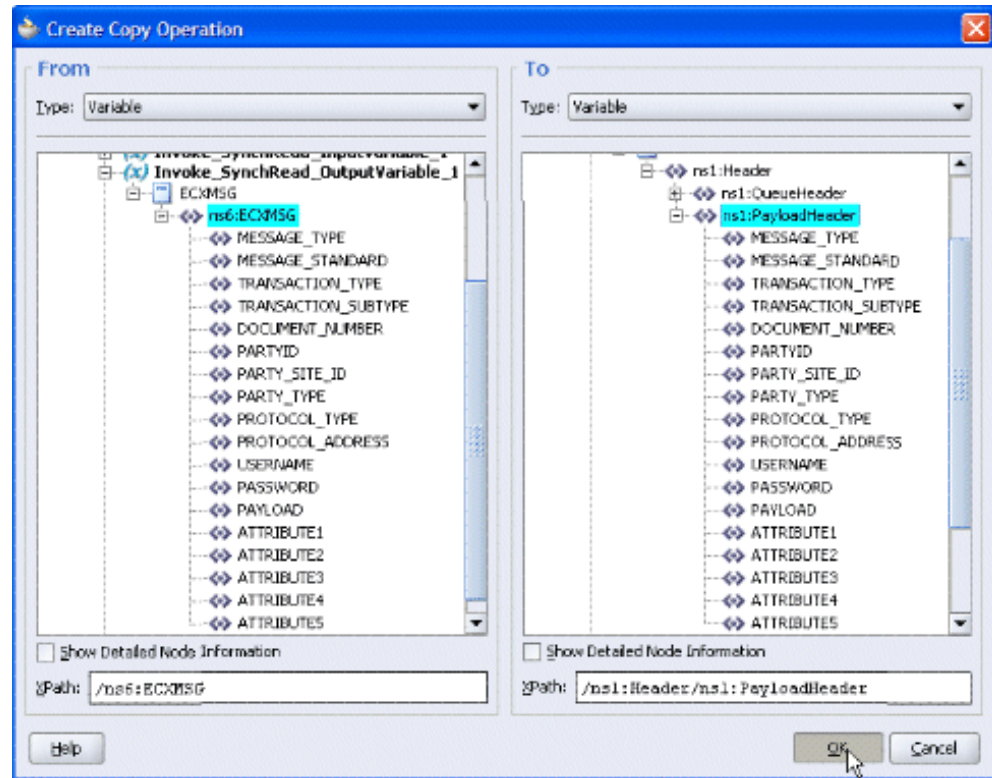


Expand the **Message Type** node, and select **Partner Links > EnqueueMsg > EnqueueMsg.wsdl > Message Types > Header\_msg**. Click **OK** to return to the Create Copy Operation dialog box.

5. The **ECXHeader** node appears in the Variables node of the To navigation tree. Expand the **ECXHeader** node and select **ns1: Header > ns1: PayloadHeader**.
6. In the From navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_SynchRead\_OutputVariable\_1 > ECXMSG** and select **ns6:ECXMSG**.

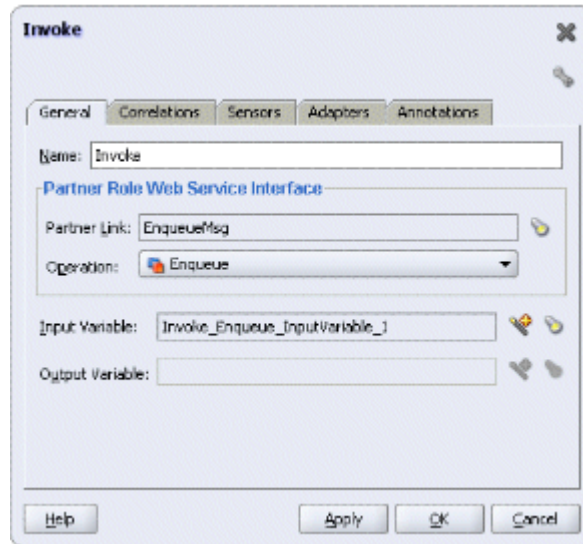
The XPath field should contain your selected entry.

### Assigning Header Variables



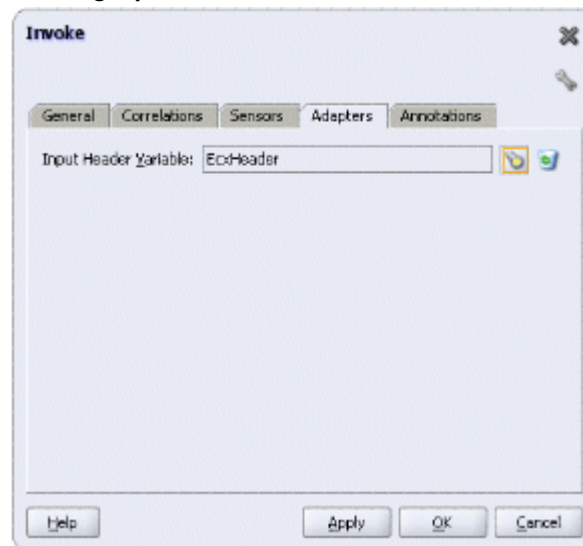
7. Click **OK** to complete the configuration of the Assign activity.
8. Modify the Invoke activity to pass the **ECXHeader** as input header variables you just declared and assigned through the Assign activity.
  1. Double-click the last Invoke activity to open the Invoke edit window.

### Editing Invoke Activity



2. Select Adapters tab and click the **Browse Variables...** icon next to the Input Header Variables field.
3. Select ECXHeader from the Variable Chooser. Click **OK**.

### Entering Input Header Variables



4. The selected ECXHeader is now populated in the Input Header Variables field. Click **Apply** and then **OK**.

## 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.

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

1. Deploy the BPEL process, page 4-30

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 4-32

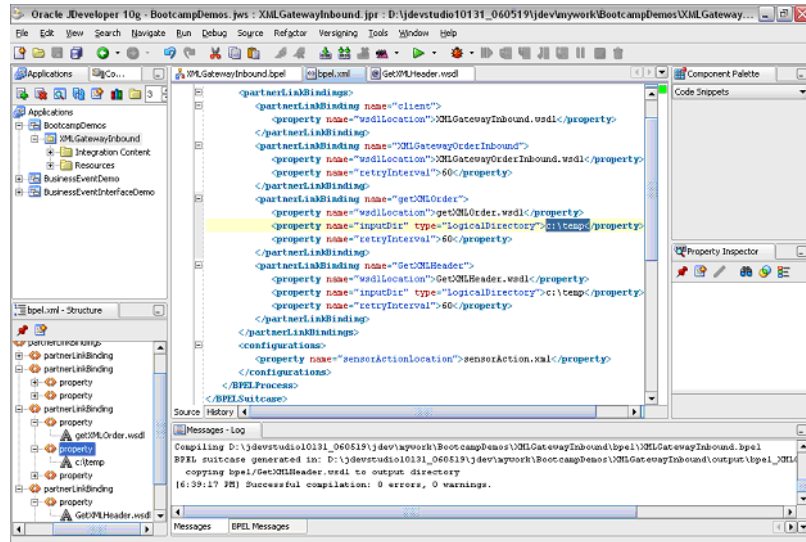
After deploying a BPEL process, you can manage the process from the BPEL console to validate the interface integration contained in your BPEL process.

## 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 Interface, you should:

- Load the `ecx_header` and `order_data_xmllg` into `C:\Temp` folder.
- Go to the `BPEL.xml` and point the logical directory to `C:\Temp`

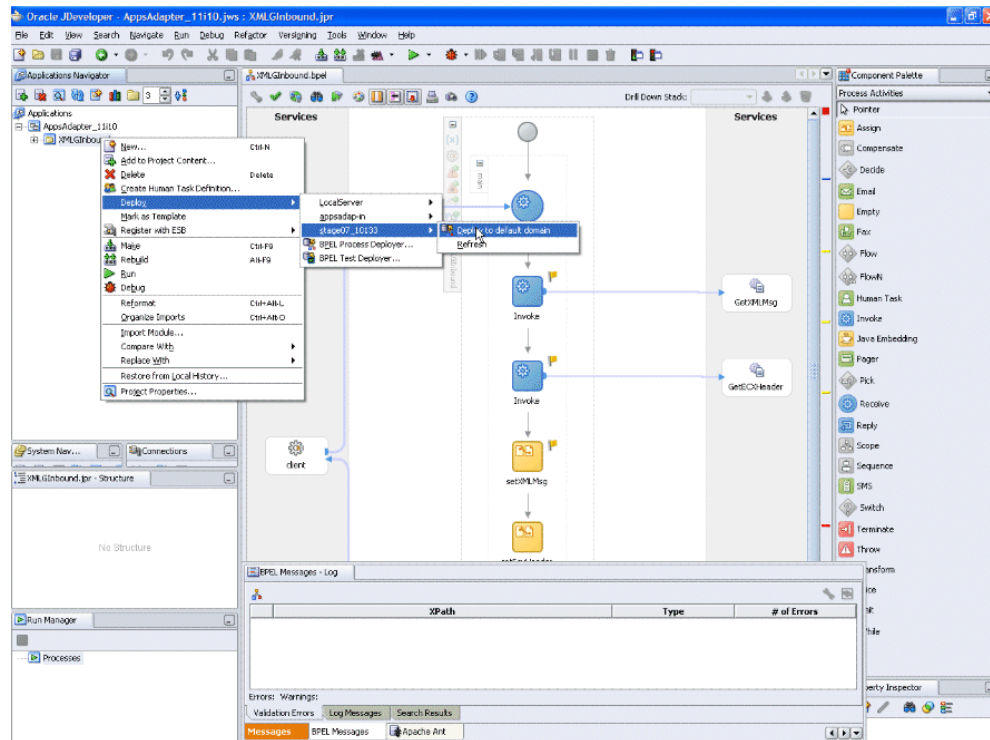


### To deploy the BPEL process:

1. In the Applications Navigator of JDeveloper BPEL Designer, select the XMLGInbound project.
2. Right-click the project and select **Deploy** from the menu. Click **Invoke Deployment Tool** and enter your BPEL Process Manager Information.

For example, if you are deploy to a local BPEL server, select **Local BPEL Server**, then **Deploy to Default Domain**.

## Deploying the BPEL Process

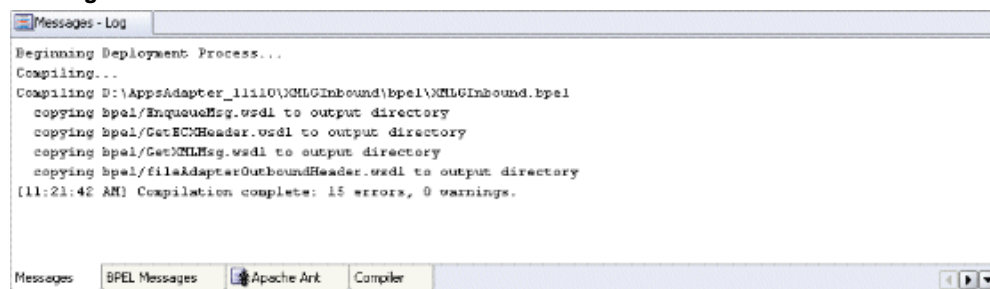


3. The Password Prompt dialog box appears.

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

4. The BPEL project is compiled and successfully deployed. You can check the progress in the Messages window.

## Messages Window



## Testing the BPEL Process

Once the BPEL process is deployed, it can be seen in the BPEL console. You can manage



and monitor the process from the BPEL console. You can also test the process and the integration interface by manually initiating the process.

**To test the BPEL process:**

1. Log into Oracle BPEL Process Manager, then select **BPEL Console**. The BPEL console login screen appears.
2. Select **Default** in the **Domain** box. Enter the password for the default domain and click **Login** to access the console.
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 Applications**

Once the BPEL process is successfully initiated and completed, you can validate it through the relevant module in Oracle Applications. For example, you can check for the creation of a purchase order in Oracle Order Management.

**To validate it in Oracle Order Management:**

1. Log on to the Forms-based Oracle Applications 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. This would bring up the details of newly created order.

## Sales Orders

**Sales Orders (Vision Operations) - 63735, Business World**

Order Information | Line Items

Default

Main | Others

Customer	Business World	Order Number	63735
Customer Number	1608	Order Type	Mixed
Customer PO	POID_01	Date Ordered	01-NOV-2007 06:18:00
Customer Contact		Price List	Corporate
Blanket Number		Salesperson	Sprague, Mr. Howard
Ship To Location	San Jose (OPS)	Status	Entered
	2391 L Street	Currency	USD
	San Jose, CA, 95106, US	Subtotal	2,399.00
Bill To Location	San Jose (OPS)	Tax	215.91
	2391 L Street	Charges	71.97
		Total	2,686.88
	San Jose, CA, 95106, US		

Actions | Related Items | Configurator | Availability | Book Order

You can also select the Items tab for item details.

Additionally, you can also 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 Applications 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.

To validate it using Oracle Transaction Monitor, you need to log on to Oracle Applications 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

**ORACLE<sup>TM</sup> Transaction Monitor** Diagnostics Home Logout Preferences Help

**Warning**  
Low-level logging is currently enabled. Your application will not perform as well while low-level logging is on.

Transaction Monitor: Search

**Search Criteria**

Select search criteria and press Go to view the report.

☒ Inbound Messages  
Processing Status:

☐ Outbound Messages  
Generation Status:   
Delivery Status:   
Retry Status:

Transaction Type:  Transaction Subtype:   
Source TP Location Code:  Trading Partner Name:   
Document ID:  Site Name:   
Party Type:   
From Date:   
To Date:

Diagnostics Home Logout Preferences Help

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

Click **Go** to retrieve all XML inbound messages listed in the Inbound Search Results region.

## Listing All XML Gateway Inbound Documents

**ORACLE<sup>TM</sup> Transaction Monitor** Diagnostics Home Logout Preferences Help

Transaction Monitor: Search >

**Warning**  
Low-level logging is currently enabled. Your application will not perform as well while low-level logging is on.

**Transaction Monitor**

**Inbound Search Results**

Document ID	Trading Partner Name	External Transaction Type	External Transaction Subtype	Processing Timestamp	Processing Status
<a href="#">demo_nacenda</a>	N/A	PO	PROCESS	01-Nov-2007 03:48:45	Success
<a href="#">demo_1</a>	N/A	PO	PROCESS	01-Nov-2007 05:01:24	Success
<a href="#">temp_order</a>	N/A	PO	PROCESS	01-Nov-2007 05:09:30	Success
<a href="#">trial1</a>	N/A	PO	PROCESS	01-Nov-2007 05:47:25	Success
<a href="#">POID_01</a>	N/A	PO	PROCESS	01-Nov-2007 06:15:42	Success

Diagnostics Home Logout Preferences Help

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

See *Oracle XML Gateway User's Guide* for details on using the Transaction Monitor.

## 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 4-36
- Deploying and Testing a BPEL Process at Run Time, page 4-32

## Using a XML Gateway Outbound in Creating a BPEL Process 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 Applications.

### 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.

After deploying the BPEL process, 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 Applications. The correlation identifier enables you to label messages meant for a specific agent, in case there are multiple agents listening on the outbound queue. The agent listening for a particular correlation picks up the messages that match the correlation identifier for the agent.

To set up the correlation identifier:

1. Log in to Oracle Applications with the XML Gateway responsibility. The Navigator page appears.
2. Click the **XML Gateway** link.
3. Click the **Define Lookup Values** link under XML Gateway.
4. 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 4-37  
Use this step to create a new BPEL project called `XMLGOutbound.bpel`.
2. Create a Partner Link for AQ Adapter, page 4-38  
Use this step to dequeue the event details from the `ECX_OUTBOUND` queue.
3. Add a Receive activity, page 4-44  
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 4-46  
This is to write PO acknowledgement details in an XML file as an output file.
5. Add an Invoke activity, page 4-52  
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 4-54  
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 A-1 and *Oracle BPEL Process Manager Developer's Guide* for details.

### **Creating a New BPEL Project**

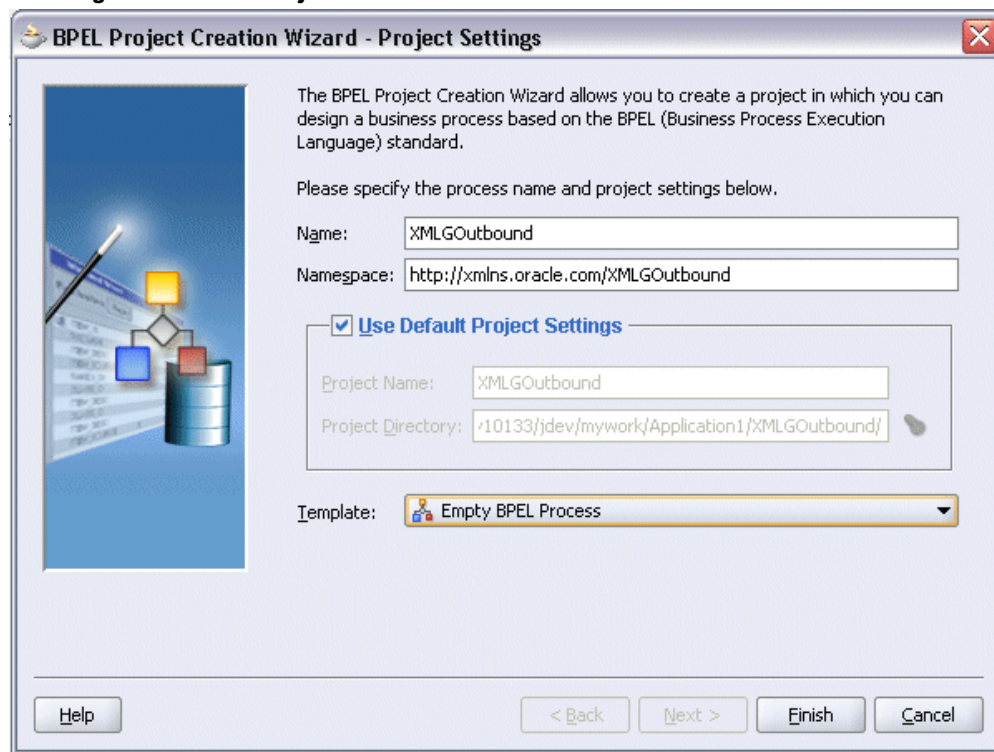
Use this step to create a new BPEL project that will contain various BPEL process activities.

#### **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, 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; for example, XMLGOutbound.
8. From the Template list, select **Empty BPEL Process**, then select **Use Default Project Settings**.
9. Click **Finish**.

#### Creating a New BPEL Project



**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: XMLGOutbound

Namespace: http://xmlns.oracle.com/XMLGOutbound

☒ **Use Default Project Settings**

Project Name: XMLGOutbound

Project Directory: \\10133\\jdev\\mywork\\Application1\\XMLGOutbound\\

Template: Empty BPEL Process

Help < Back Next > Finish Cancel

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

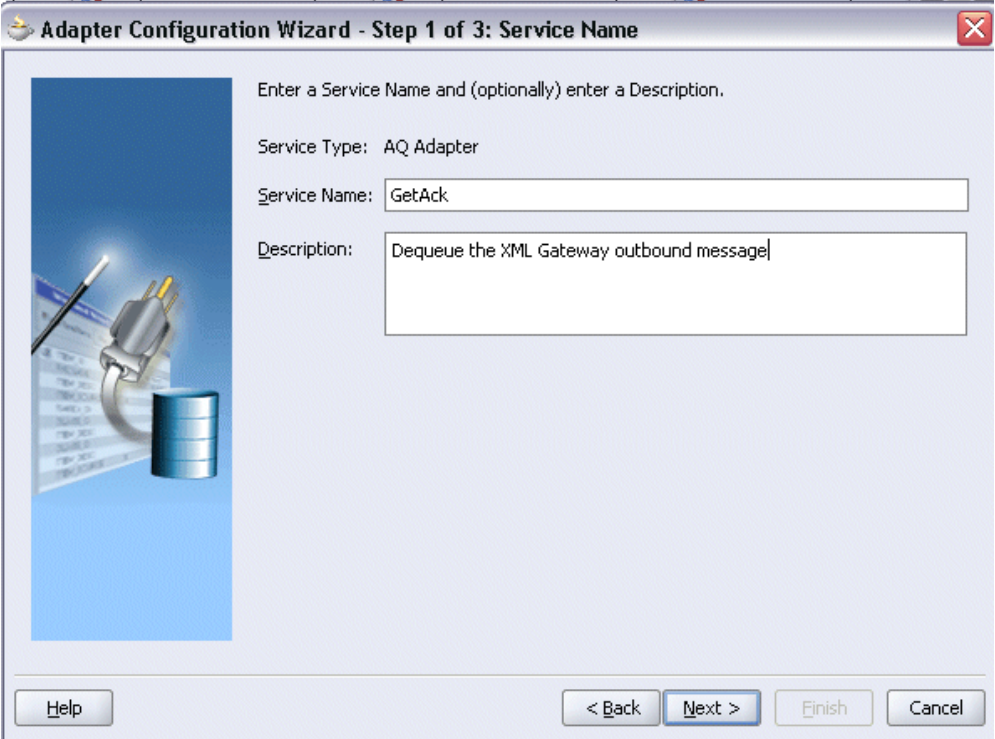
#### Creating a Partner Link for AQ Adapter

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 the 'Adapter Configuration Wizard - Step 1 of 3: Service Name' dialog box. 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' is set to 'AQ Adapter'. The 'Service Name' field contains 'GetAck' and the 'Description' field contains '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 Applications is running.

#### To create a new database connection:

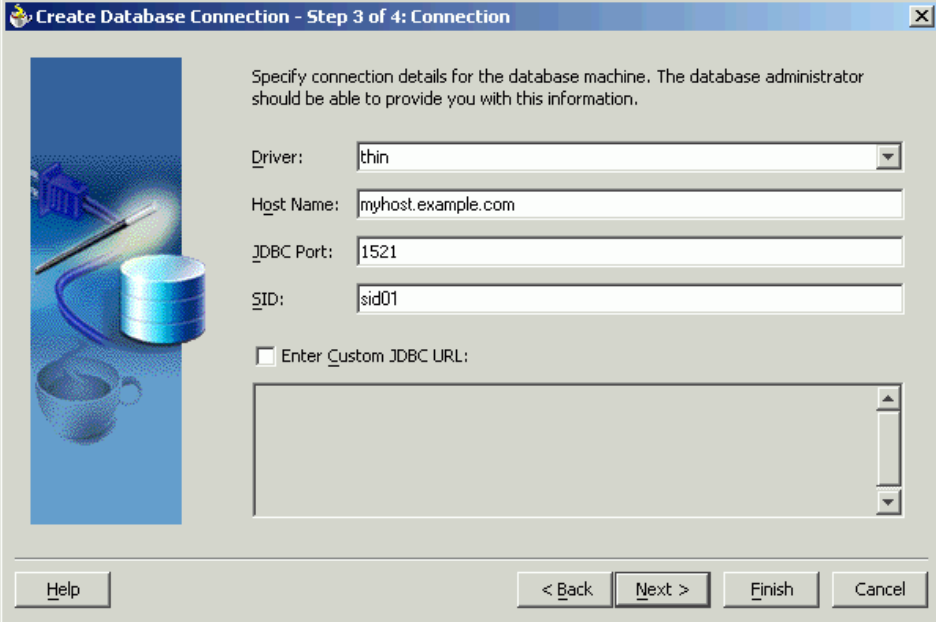
1. Enter an unique connection name and then select a connection type for the database connection. Click **Next**.
2. Enter an appropriate username and password to authenticate the database



connection in the Authentication dialog box. Click **Next**

3. Specify the following information in the Connection dialog box:

#### ***Specifying the New Database Connection Information***



**Create Database Connection - Step 3 of 4: Connection**

Specify connection details for the database machine. The database administrator should be able to provide you with this information.

**Driver:** thin

**Host Name:** myhost.example.com

**JDBC Port:** 1521

**SID:** sid01

☐ Enter Custom JDBC URL:

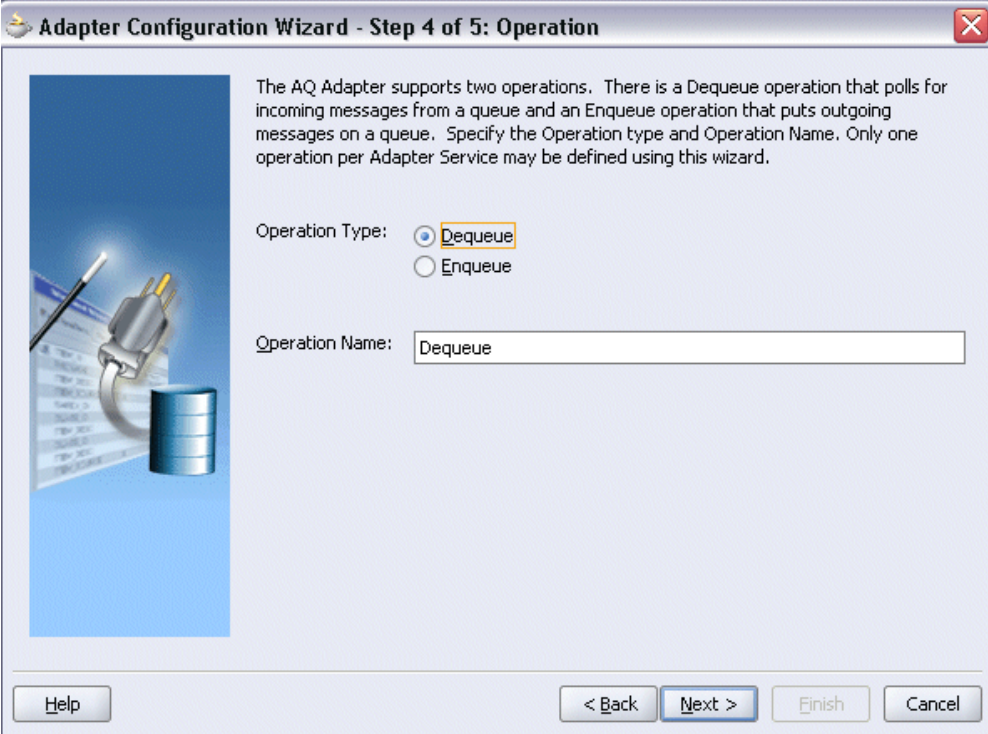
Help < Back Next > Finish Cancel

- 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.



**Adapter Configuration Wizard - Step 4 of 5: Operation**

The AQ Adapter supports two operations. There is a Dequeue operation that polls for incoming messages from a queue and an Enqueue operation that puts outgoing messages on a queue. Specify the Operation type and Operation Name. Only one operation per Adapter Service may be defined using this wizard.

Operation Type: ☒ Dequeue ☐ Enqueue

Operation Name:

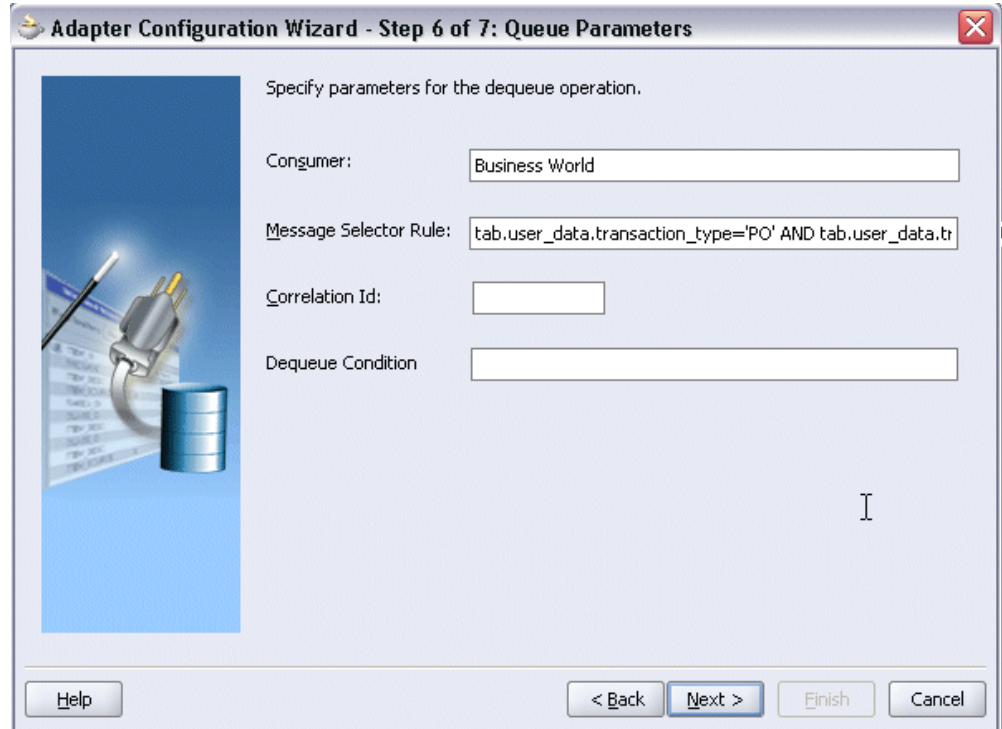
Help < Back Next > Finish Cancel

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:

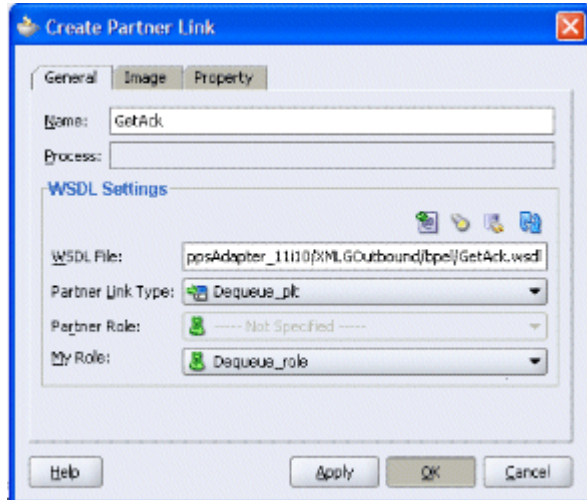
- Enter an unique consumer name.
- Enter message selector rule information (such as  
tab.user\_data.transaction\_type='PO' AND  
tab.user\_data.transaction\_subtype='POO').

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, then click the **Create** icon next to the **Variable** field to create a new variable. The Create Variable dialog box appears.
4. Select **Global Variable**, 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 **OK** to finish configuring the Receive activity.

### Editing the Receive Activity

**Edit Receive**

Errors: 1

General Correlations Sensors Adapters Annotations

Name: Receive

**My Role WebService Interface**

Partner Link: GetAck

Operation: Dequeue

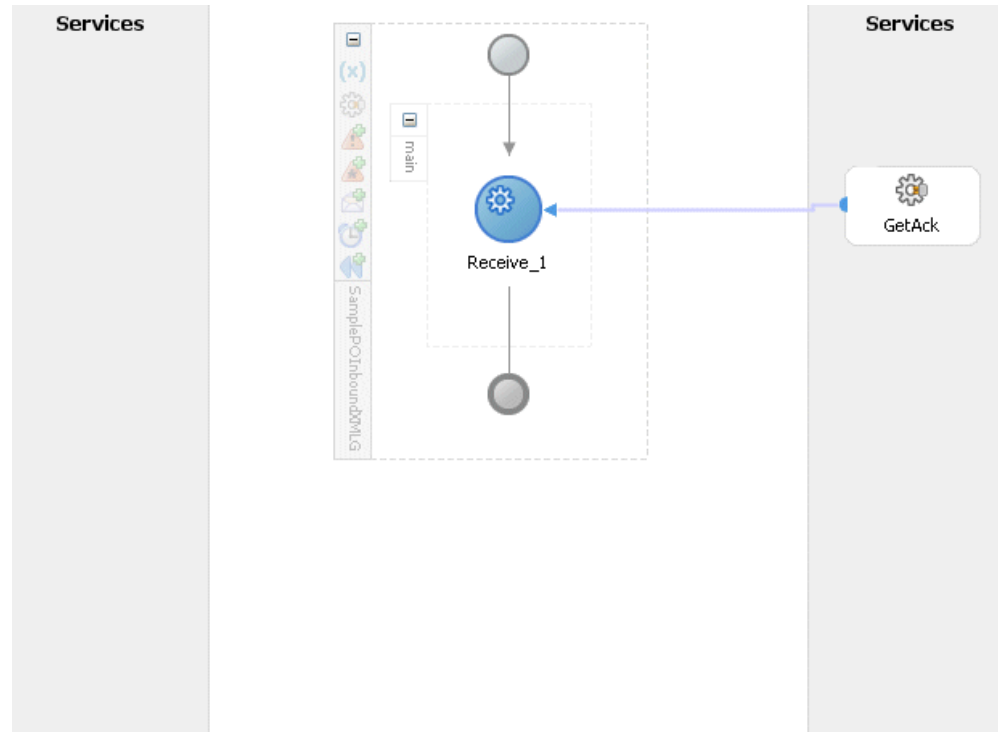
Variable: Receive\_Dequeue\_InputVariable

☒ Create Instance

Help Apply OK Cancel

The Receive activity appears in the following BPEL process diagram:

### Adding a Receive Activity



### Adding a Partner Link for File Adapter

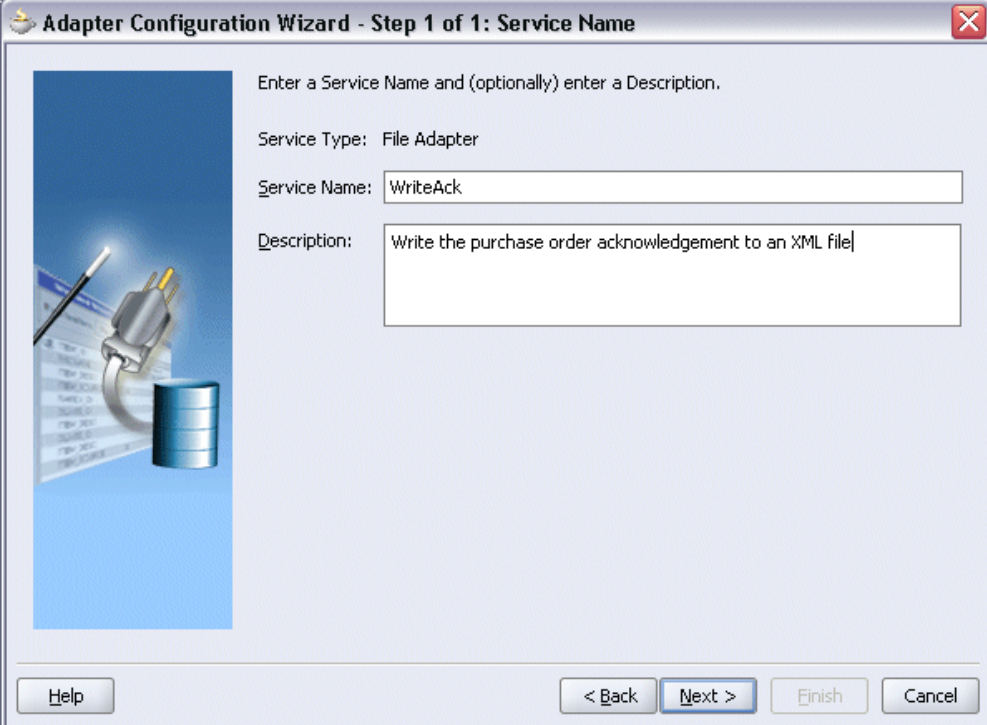
Use this step to configure a partner link by writing the purchase order acknowledgement to an XML file.

#### 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 >", "Finish", and "Cancel".

Adapter Configuration Wizard - Step 1 of 1: Service Name

Enter a Service Name and (optionally) enter a Description.

Service Type: File Adapter


Service Name: WriteAck

Description: Write the purchase order acknowledgement to an XML file

Help < Back Next > Finish Cancel

3. Enter a name for the file adapter service; for example, `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

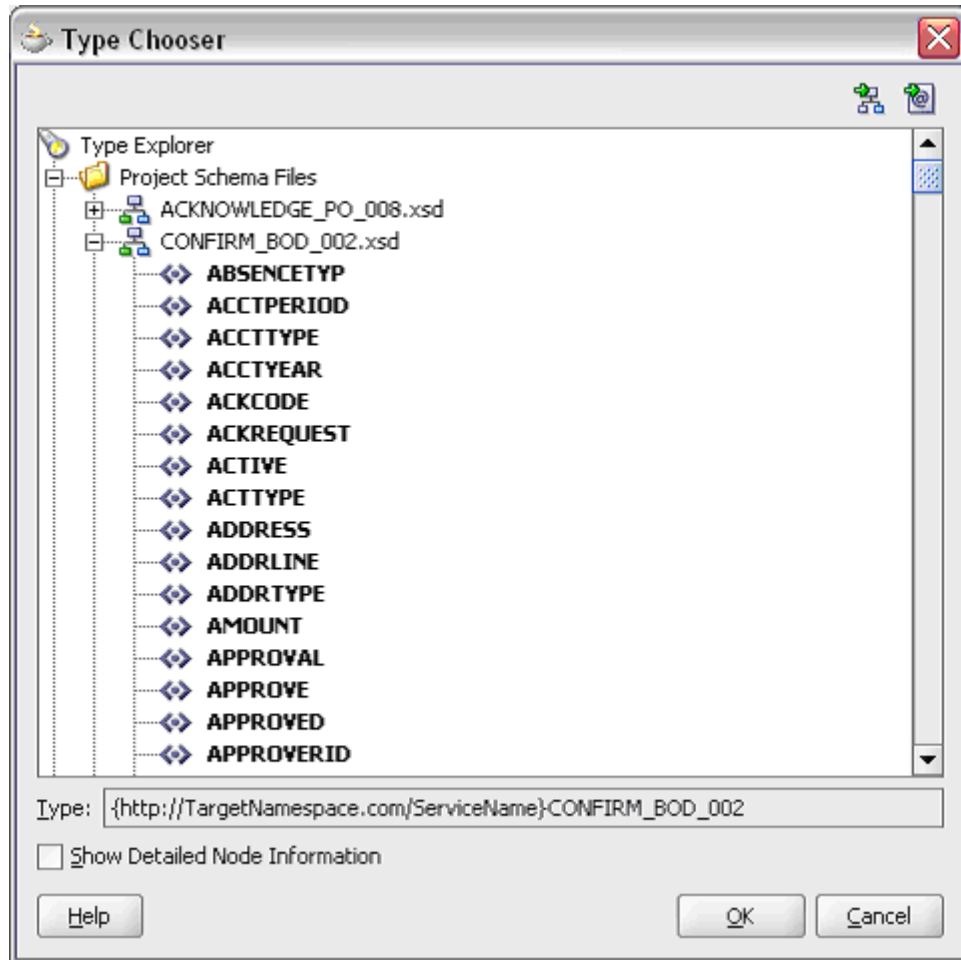
The screenshot shows the 'Adapter Configuration Wizard - Step 3 of 4: File Configuration' dialog box. On the left is a graphic of a USB cable and a database cylinder. The main area contains the following fields and options:

- Specify the parameters for the Write File operation.
- Directory specified as: ☐ Physical Path ☒ Logical Name
- Directory for Outgoing Files (logical name):
- File Naming Convention (po\_%SEQ%.txt):
- Write to new file when existing file meets any of these conditions:
  - ☒ Number of Messages Equals:  (dropdown arrow)
  - ☐ Elapsed Time Exceeds:  (dropdown arrow) minutes (dropdown arrow)
  - ☐ File Size Exceeds:  (dropdown arrow) kilobytes (dropdown arrow)

At the bottom are buttons: Help, < Back, Next > (highlighted), 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; for example, 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**.

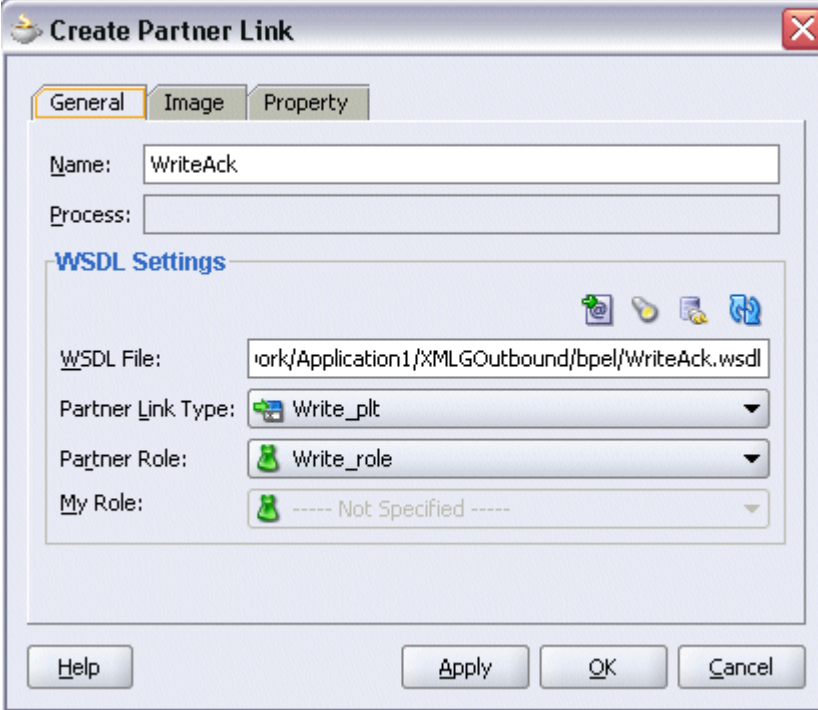
### Type Chooser



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

9. Click **Next**, 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:

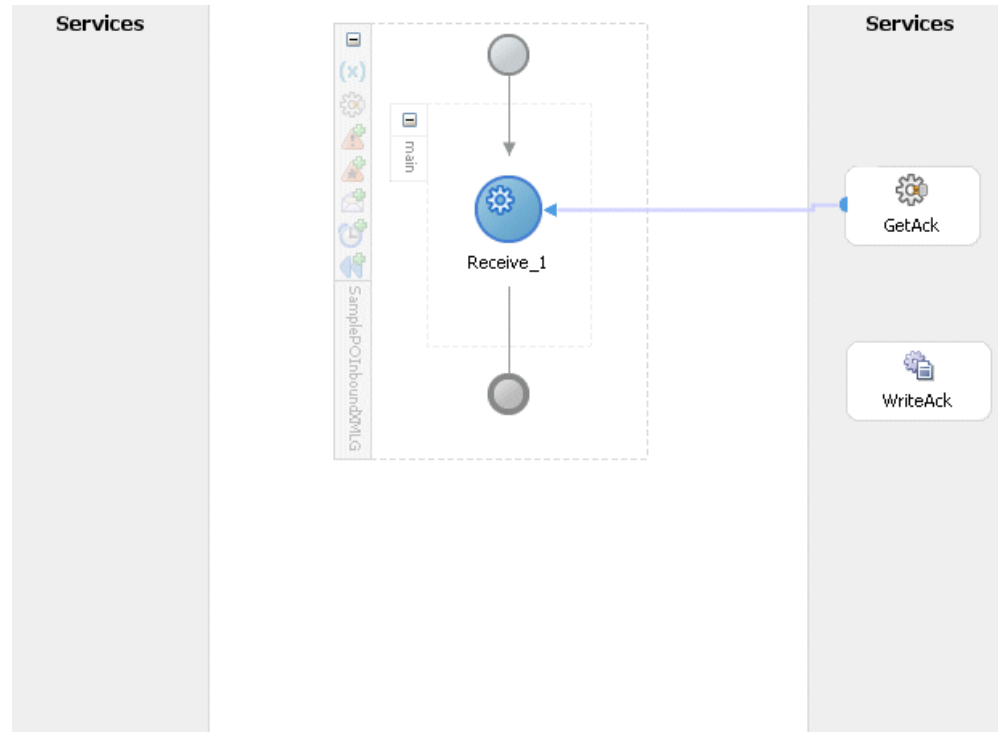
- Name:** WriteAck
- Process:** (empty text box)
- WSDL Settings:**
  - WSDL File:** iork/Application1/XMLGOutbound/bpel/WriteAck.wsdl
  - Partner Link Type:** Write\_plt (dropdown menu)
  - Partner Role:** Write\_role (dropdown menu)
  - My Role:** ---- Not Specified ---- (dropdown menu)

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 WriteAck Partner Link appears in the following 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 a 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:

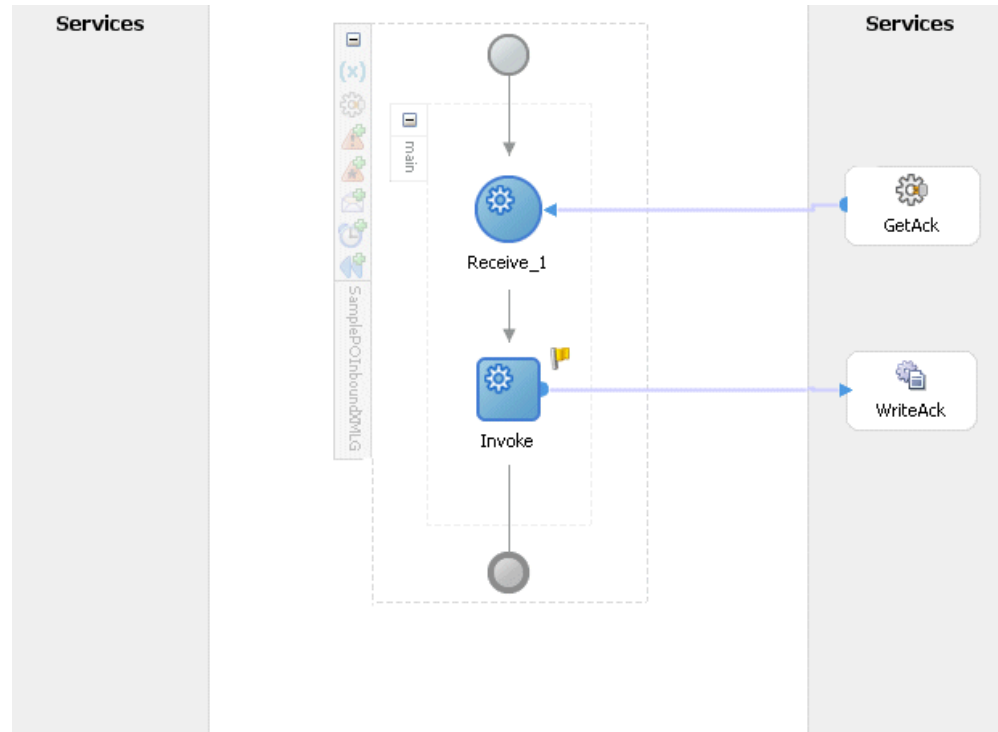
Help Apply OK Cancel

3. Enter a name for the Invoke activity, 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**, 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



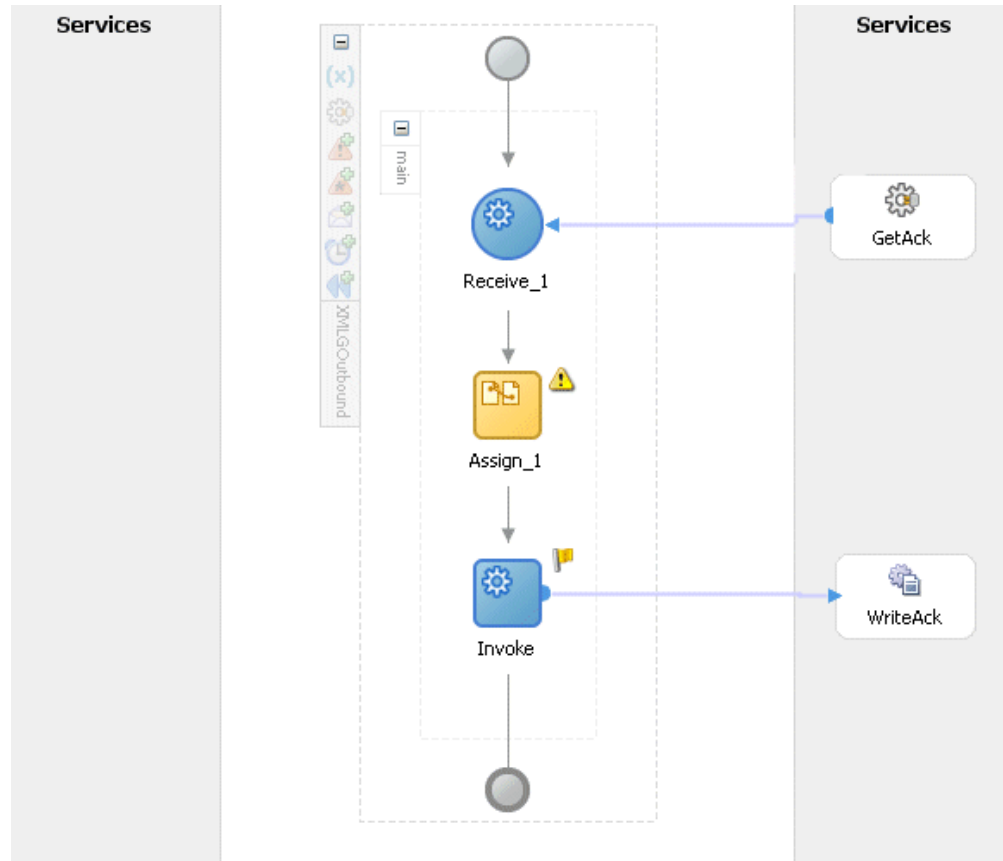
### Adding an Assign Activity

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

**To add an Assign activity:**

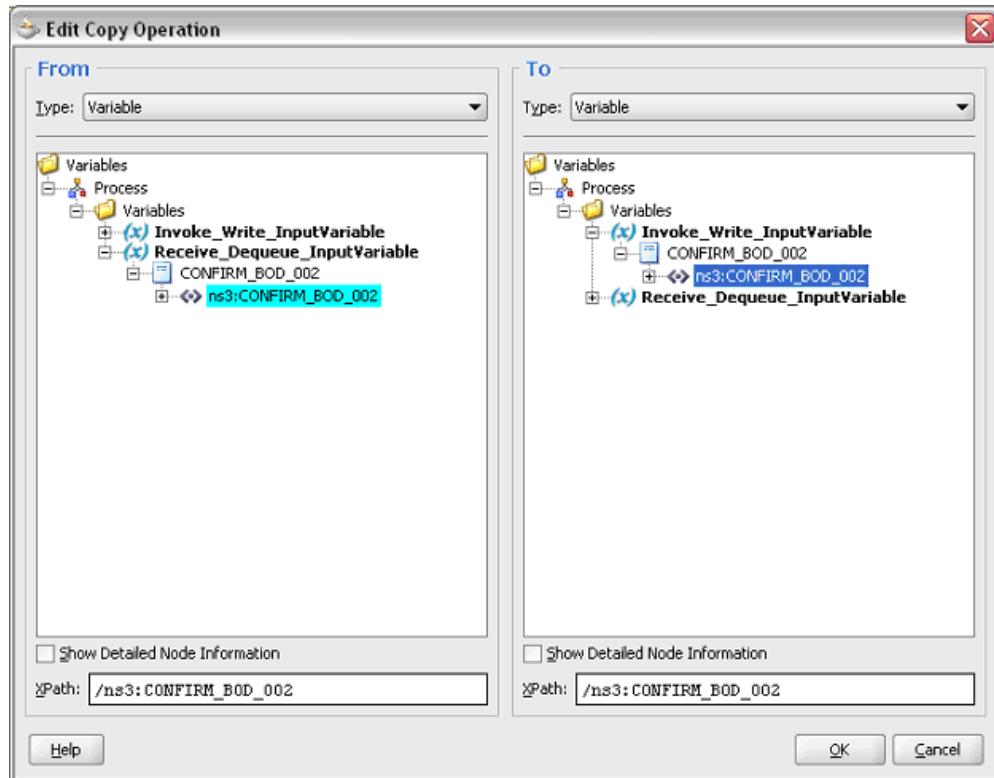
1. In JDeveloper BPEL Designer, drag and drop the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the **Receive** activity and the **Invoke** activity.

### 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**, then select **Copy Operation** from the menu. The Create Copy Operation window appears.
4. In the From navigation tree, select type Variable, then 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, then 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.

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

1. Deploy the BPEL process, page 4-57

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 4-58

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

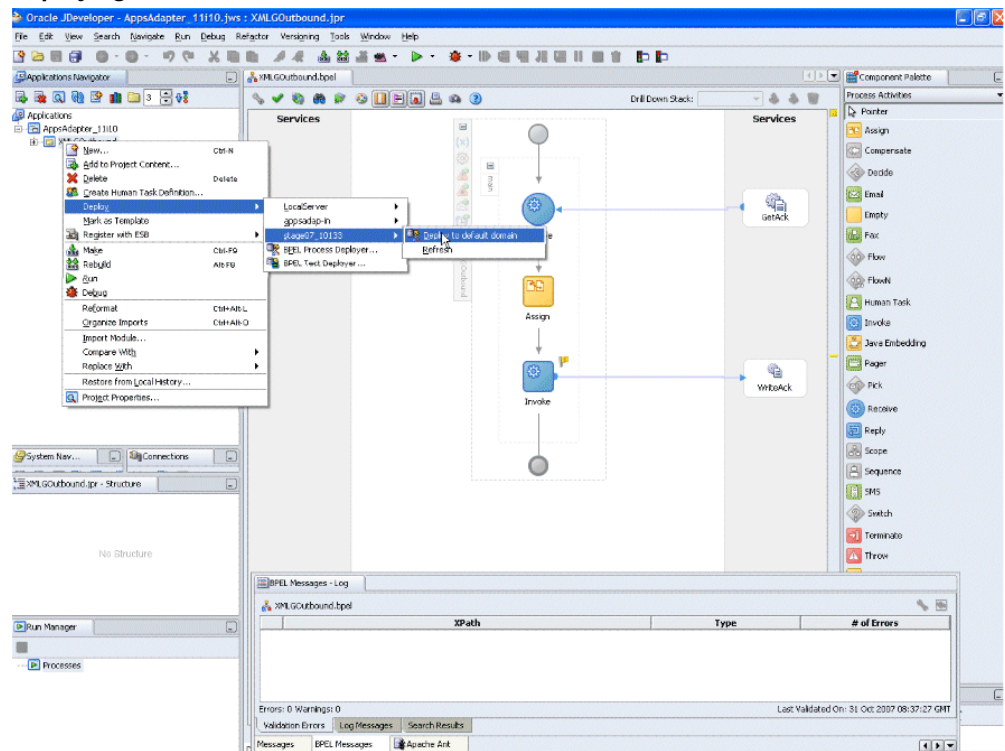
Before manually test the BPEL process, you first need to deploy it to the BPEL server.

### To deploy the BPEL process:

1. In the Applications Navigator of JDeveloper BPEL Designer, select the **XMLGOutbound** project.
2. Right-click the project and select **Deploy** action. Click **Invoke Deployment Tool** and enter your BPEL Process Manager Information.

For example, select **Deploy > RIntegrationServerConnection > Deploy to Default Domain** from the menu if it is the appropriate deploy path.

### 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.

## 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 Applications 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 XML Gateway ECX\_OUTBOUND queue.

### To manually test the BPEL process:

1. Log on to Oracle Applications 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_CBODL		
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	POSO	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

**Sales Orders (56707) - Business World**

**Order Information** | Line Items

**Main** | Others

Customer	Business World	Order Number	56707
Customer Number	1600	Order Type	Mixed
Customer PO	order_xmlg_000	Date Ordered	19 JUL 2006 05:08:48
Customer Contact		Price List	Corporate
Ship To Location	San Jose (OPS) 2391 L Street San Jose, CA, 95106, US	Salesperson	Sprague, Mr. Howard
Bill To Location	San Jose (OPS) 2391 L Street San Jose, CA, 95106, US		

**Note**: Order has been booked.

	2,399.00
	215.91
	71.97
	2,686.88

**Actions** | Related Items | Configurator | Availability | **Book Order**

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 from the Oracle Transaction Monitor. 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 '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

Applications 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 web application. At the top is a blue header with the Oracle logo and the text "Transaction Monitor". On the right side of the header are links: "Diagnostics", "Home", "Logout", "Preferences", and "Help". Below the header is a warning message: "Warning: Low-level logging is currently enabled. Your application will not perform as well while low-level logging is on." Below the warning is a section titled "Transaction Monitor: Search". Inside this section is a "Search Criteria" form. The form has a heading "Select search criteria and press Go to view the report." and two radio buttons: "Inbound Messages" (unselected) and "Outbound Messages" (selected). 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 with magnifying glass icons: "Transaction Type", "Transaction Subtype", "Source TP Location Code", "Trading Partner Name", "Document ID", and "Site Name". There is also a "Party Type" dropdown menu. At the bottom of the form are two date-time pickers: "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 very bottom of the page is a blue footer with links "About this Page" and "Privacy Statement" on the left, and "Diagnostics", "Home", "Logout", "Preferences", "Help" in the center, and "Copyright (c) 2006, Oracle. All rights reserved." on the right.

### Validating Using Oracle BPEL Console

Log into Oracle BPEL Console to confirm that the XMLGOutbound process has been deployed. This process is continuously polling 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 XML file received. Note the Reference ID such as Customer PO.

## Viewing XML File for the Receive Activity

The screenshot shows the Oracle Enterprise Manager 10g BPEL Control console. The main window displays the audit trail for a BPEL process instance titled "XMLGatewayOutbound". The audit trail shows the following steps:

- [2006/07/19 17:53:18] New instance of BPEL process "XMLGatewayOutbound" initiated (# "44").
- [2006/07/19 17:53:18] Received "Receive\_Dequeue\_InputVa" (View xml document)
- [2006/07/19 17:53:18] Updated variable "Invoke\_Write\_Input"
- [2006/07/19 17:53:18] Invoked 1-way operation "Write" on p
- [2006/07/19 17:53:18] BPEL process instance "44" completed

A separate window displays the XML content of the received document. The XML is as follows:

```
<LOGICALID />
<COMPONENT>BPEL</COMPONENT>
<TASK>POISSUE</TASK>

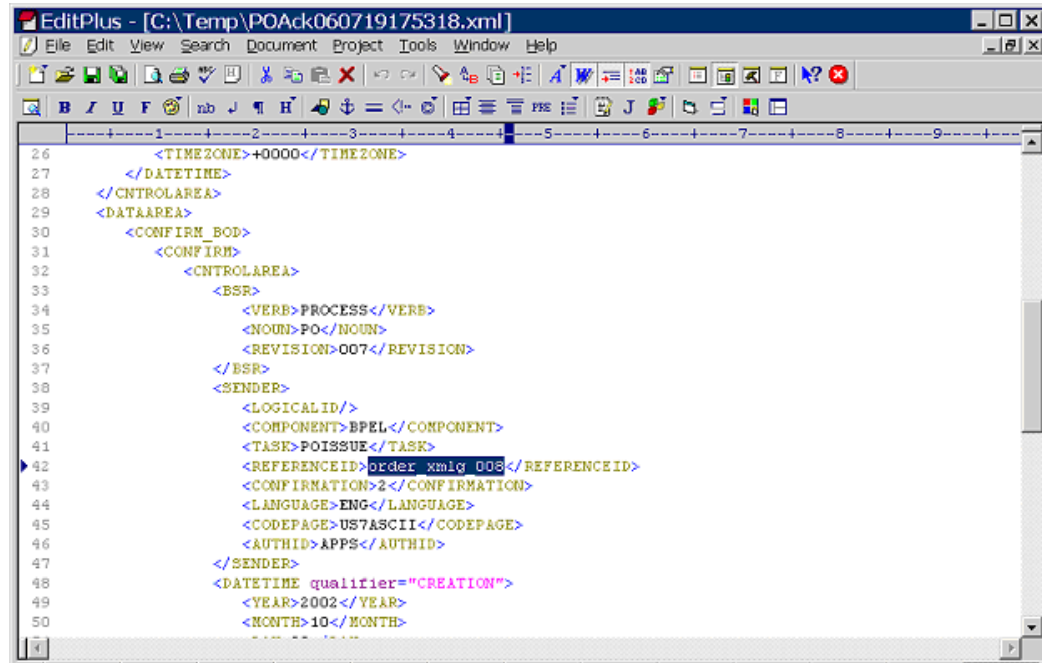
<REFERENCEID>order_xmlg_008</REFERENCEID>

<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 described in this chapter.

## 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 and enqueued to `WF_BPEL_Q` as Out Agent, we will create a BPEL process to first dequeue the subscription from the `WF_BPEL_Q` queue in order 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.

After deploying the BPEL process, 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 Business Events

Integration repository administrators must first subscribe to a business event from the Oracle Integration Repository user interface. This creates an event subscription with Out Agent as `WF_BPEL_Q`.

For example, a business event `oracle.apps.po.event.xmlpo` needs to be successfully subscribed. A confirmation message appears if the event subscription is successfully created and enqueued to the `WF_BPEL_Q`.

#### Subscribing to a Business Event

The screenshot displays the Oracle Integration Repository user interface. The top navigation bar includes links for Home, Logout, Preferences, Help, and Diagnostics. The left sidebar shows a tree view of the repository structure, with 'Purchasing' selected under the 'Business Event' category. 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' section provides information about the event, including its internal name, type, product, status, and business entity. The 'Full Description' section shows the event name 'Send PO via XML'. The 'Source Information' section lists the source file, version, and product. The interface also includes search and printable page buttons.

Business Event Details : Send PO via XML	
Internal Name	oracle.apps.po.event.xmlpo
Type	Business Event
Product	Purchasing
Status	Active
Business Entity	<a href="#">Standard Purchase Order</a>
Scope	Public
Interface Source	Custom

Full Description	
Send PO via XML	

Source Information	
Source File	patch/115/po/US/rpodemoUpdatede.wfx
Source Version	12.0
Source Product	PO

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 5-4

Use this step to create a new BPEL project called `GetPurchaseOrder.bpel`.

2. Create a Partner Link for AQ Adapter, page 5-5

Use this step to dequeue the event details from the `WF_BPEL_Q` queue.

3. Add a Receive activity, page 5-12

Use the Receive activity to take event details as an input to the Assign activity.

4. Create a Partner Link for File Adapter, page 5-14

This is to write event details in an XML file as an output file.

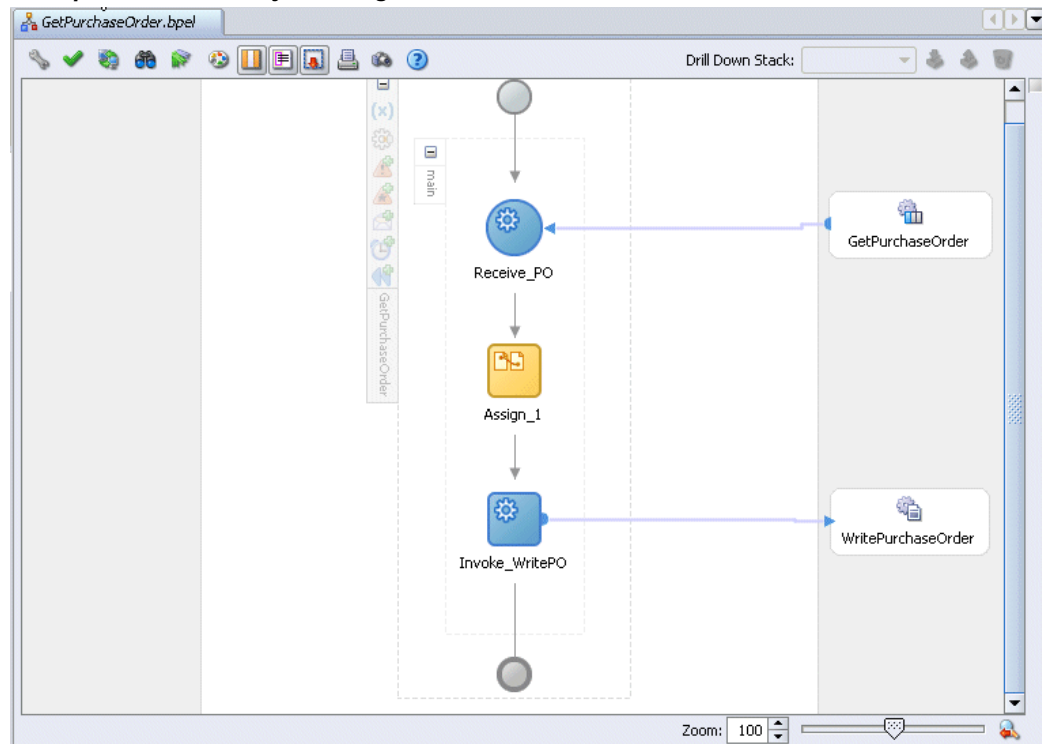
5. Add an Invoke activity, page 5-20

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 5-22

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 A-1 and *Oracle BPEL Process Manager Developer's Guide* for details.

## Creating a New BPEL Project

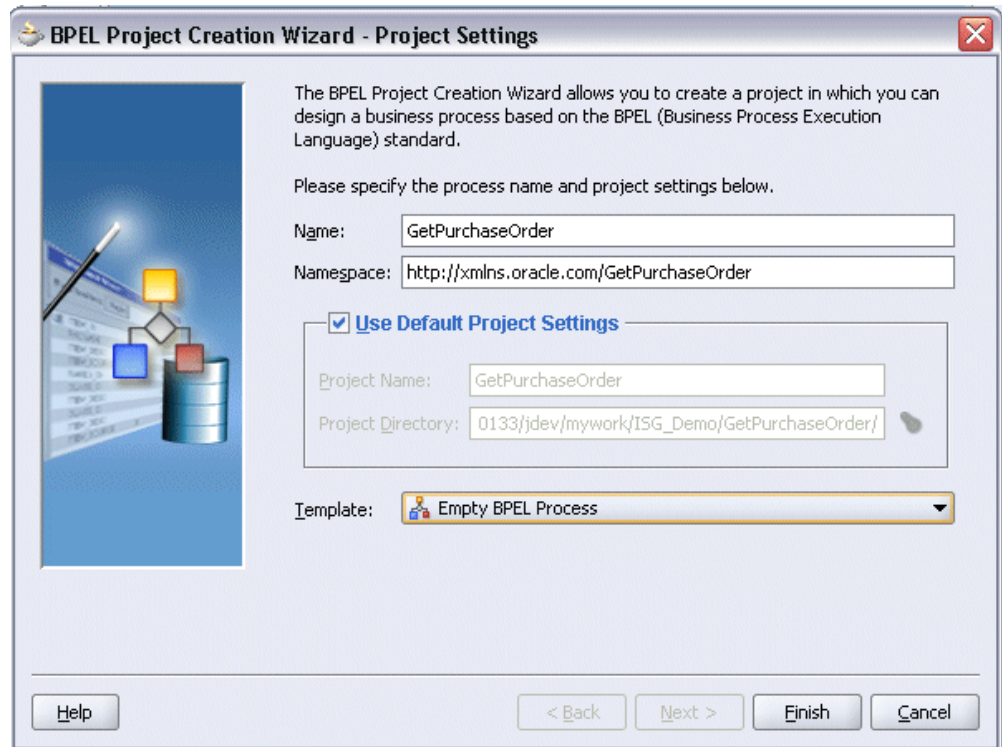
Use this step to create a new BPEL project that will contain various BPEL process activities.

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, 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; for example, `GetPurchaseOrder`.
8. From the Template list, select **Empty BPEL Process**, then select **Use Default Project Settings**.
9. Click **Finish**.

#### Creating a New Project



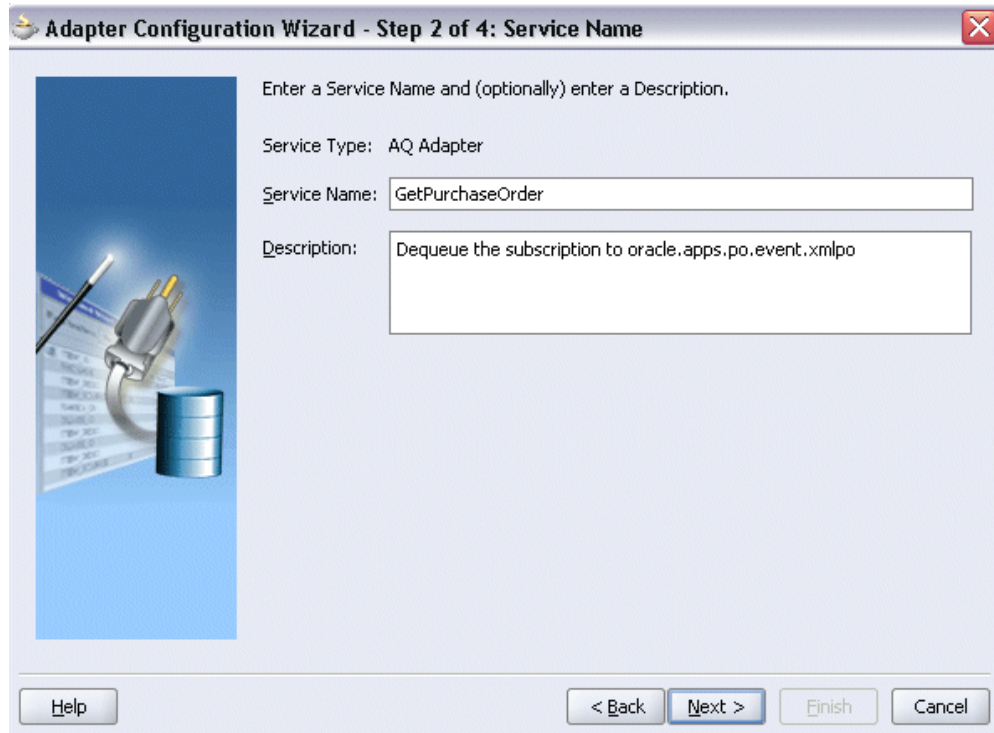
A new BPEL project is created with the required source files including `bpel.xml`, using the name you specified (for example, `GetPurchaseOrder.bpel`).

## 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.

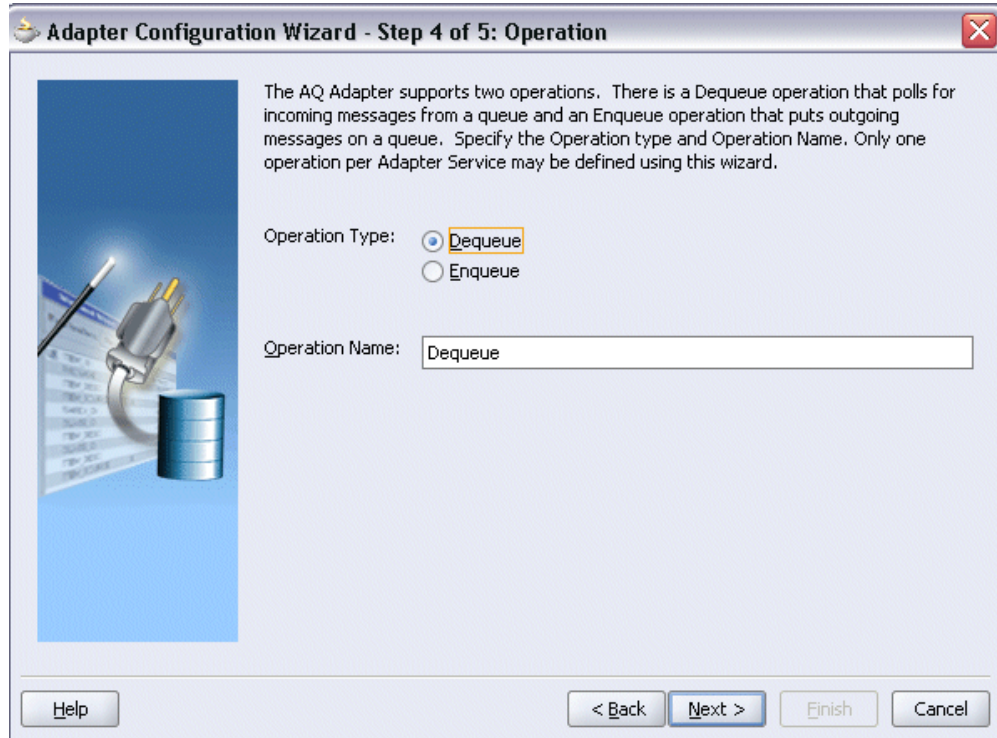
**Note:** You need to connect to the database where Oracle Applications is running.

**To create a new database connection:**

1. 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.

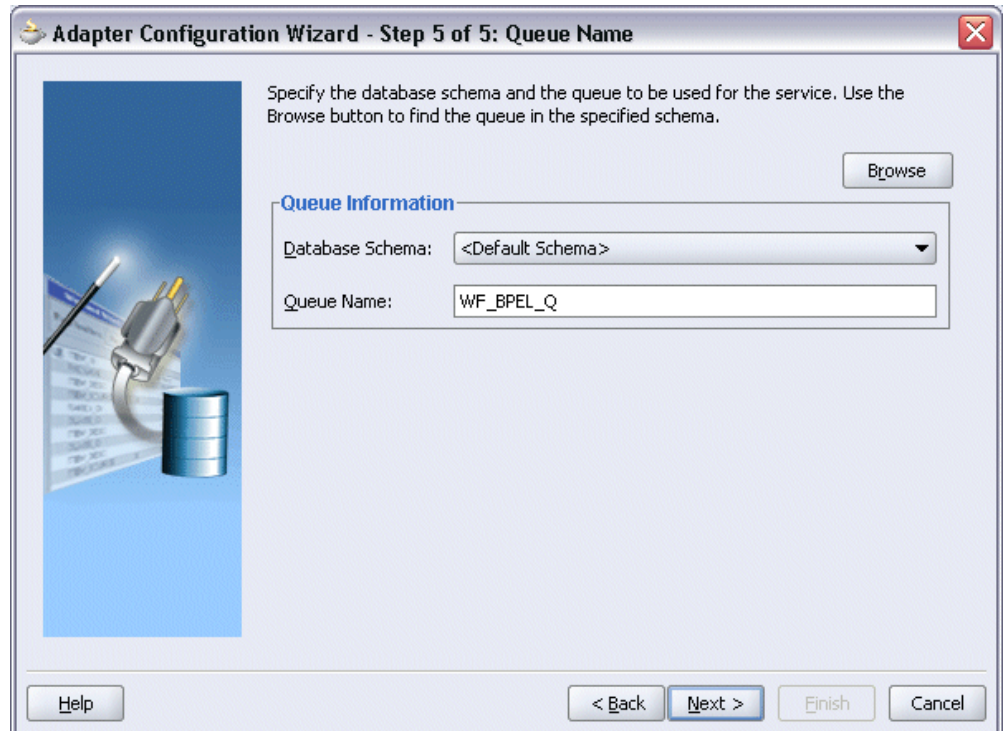




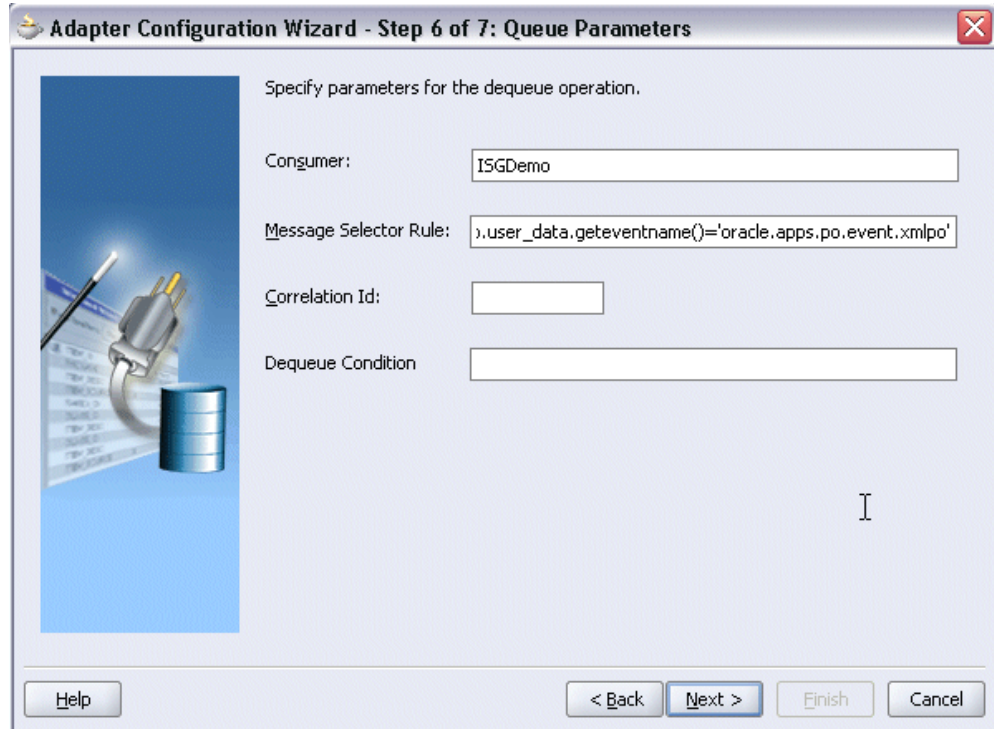
7. Click **Next** to open the Queue Name dialog box.

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.

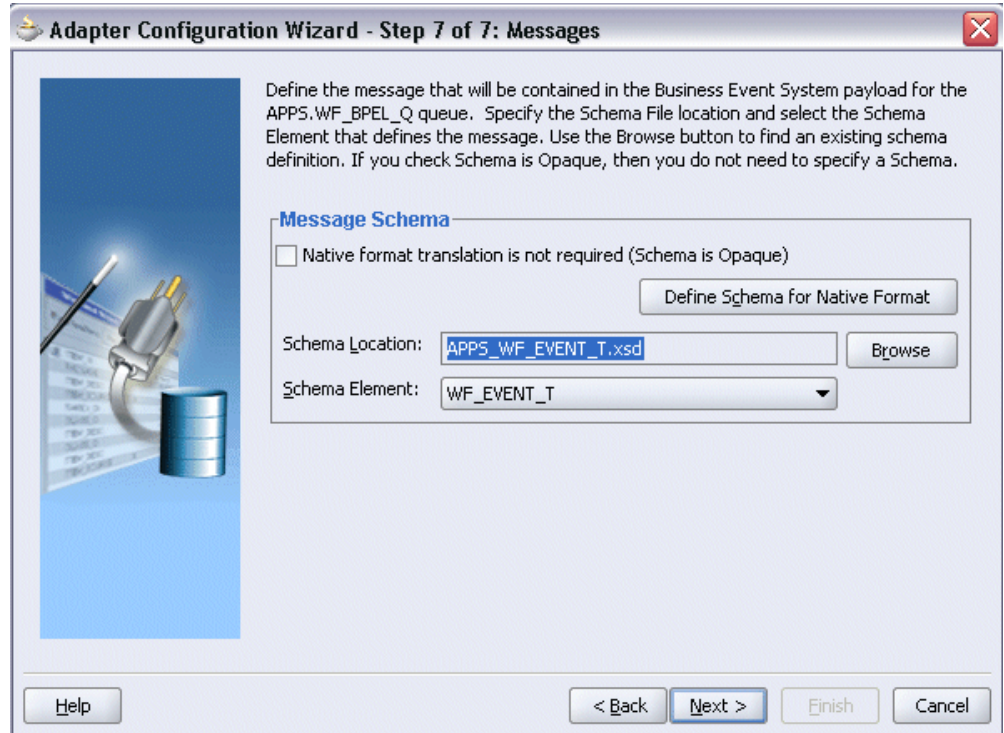


Enter the following information:

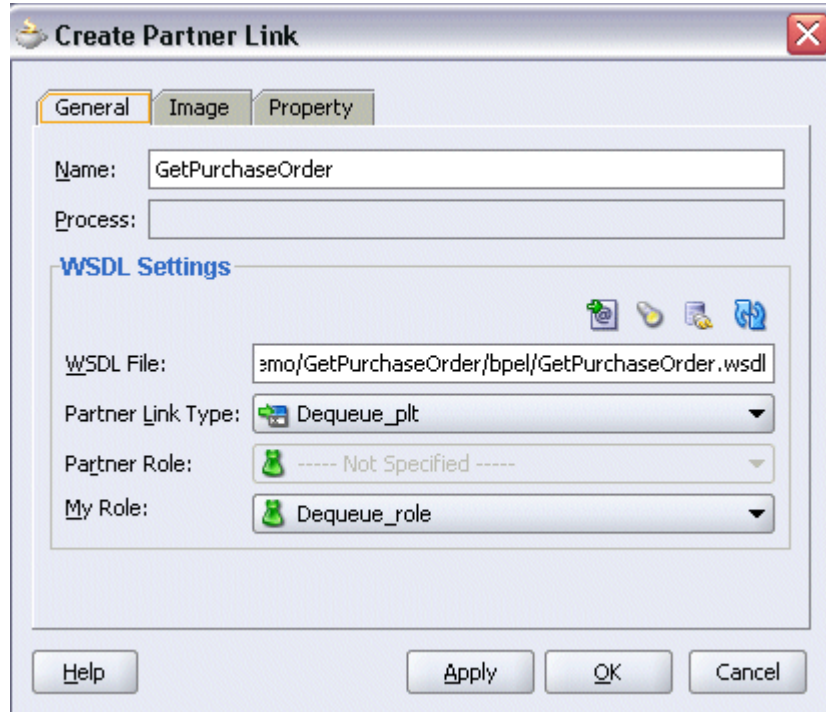
- Enter an unique consumer name.  
**Important:** In order for the subscribed business event to be successfully enqueued to WF\_BPEL\_Q queue, the consumer name must be unique.
- Enter message selector rule information (such as `tab.user_data.geteventname()='oracle.apps.po.evnt.xmlpo'`).

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

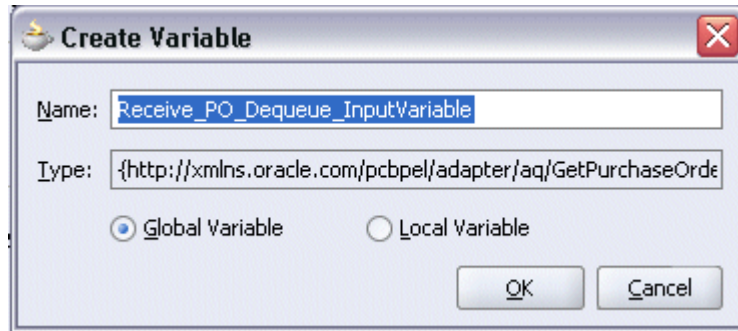
This step is to configure a Receive activity to receive XML data from the partner link `GetPurchaseOrder` that you configured for the AQ adapter service for the business event.

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', then click the **Create** icon next to the **Variable** field to create a new variable. The Create Variable dialog box appears.

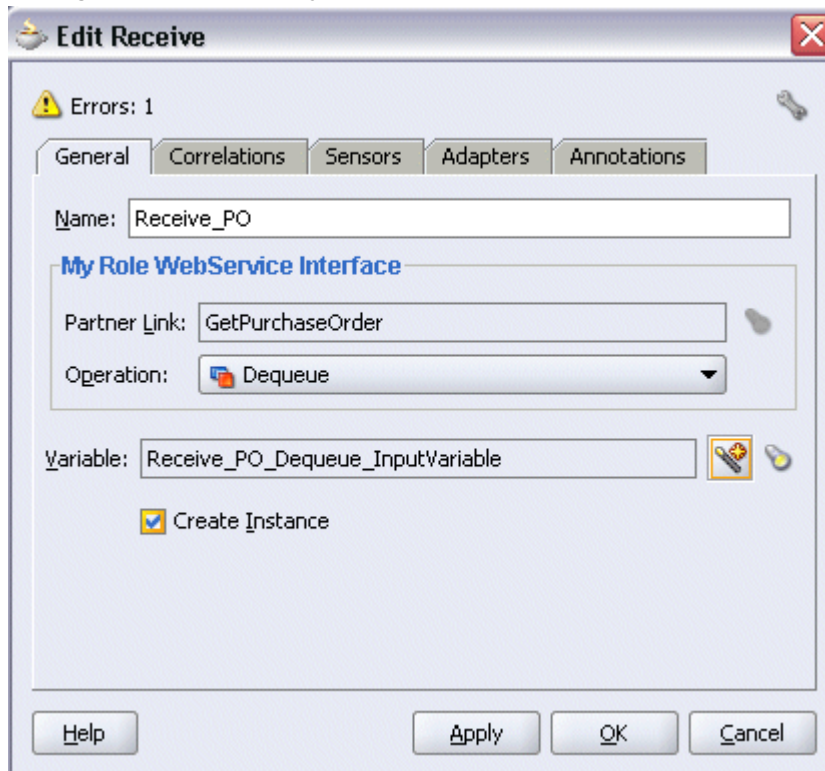
#### Creating a Variable



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". Below the type field, the **Global Variable** radio button is selected, and the **Local Variable** radio button is unselected. At the bottom right, there are **OK** and **Cancel** buttons.

4. Select **Global Variable**, 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 **OK** to finish configuring the Receive activity.

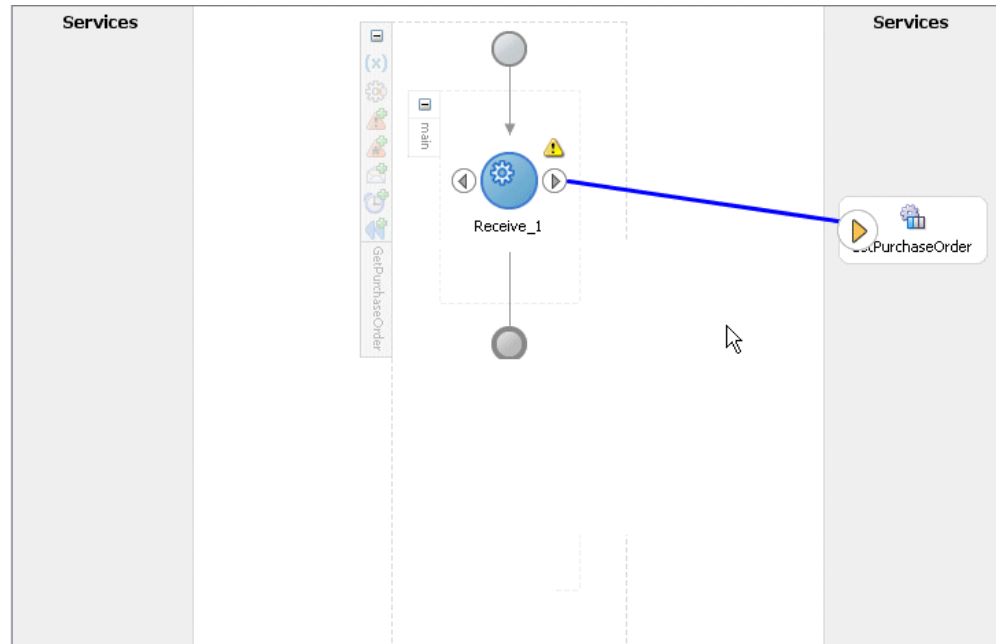
#### Editing the Receive Activity



The **Edit Receive** dialog box is shown. It has a title bar with a close button. Below the title bar, there is a warning icon and the text "Errors: 1". There are five tabs: **General**, **Correlations**, **Sensors**, **Adapters**, and **Annotations**. The **General** tab is selected. The **Name** field contains "Receive\_PO". Below the name field, there is a section titled "My Role WebService Interface". Inside this section, the **Partner Link** field contains "GetPurchaseOrder". The **Operation** field is a dropdown menu with "Dequeue" selected. Below the operation field, the **Variable** field contains "Receive\_PO\_Dequeue\_InputVariable". To the right of the variable field are two icons: a key icon and a lightbulb icon. Below the variable field, the **Create Instance** checkbox is checked. At the bottom, there are **Help**, **Apply**, **OK**, and **Cancel** buttons.

The Receive activity appears in the following BPEL process diagram:

### Process Diagram with Receive Activity



## Adding a Partner Link for File Adapter

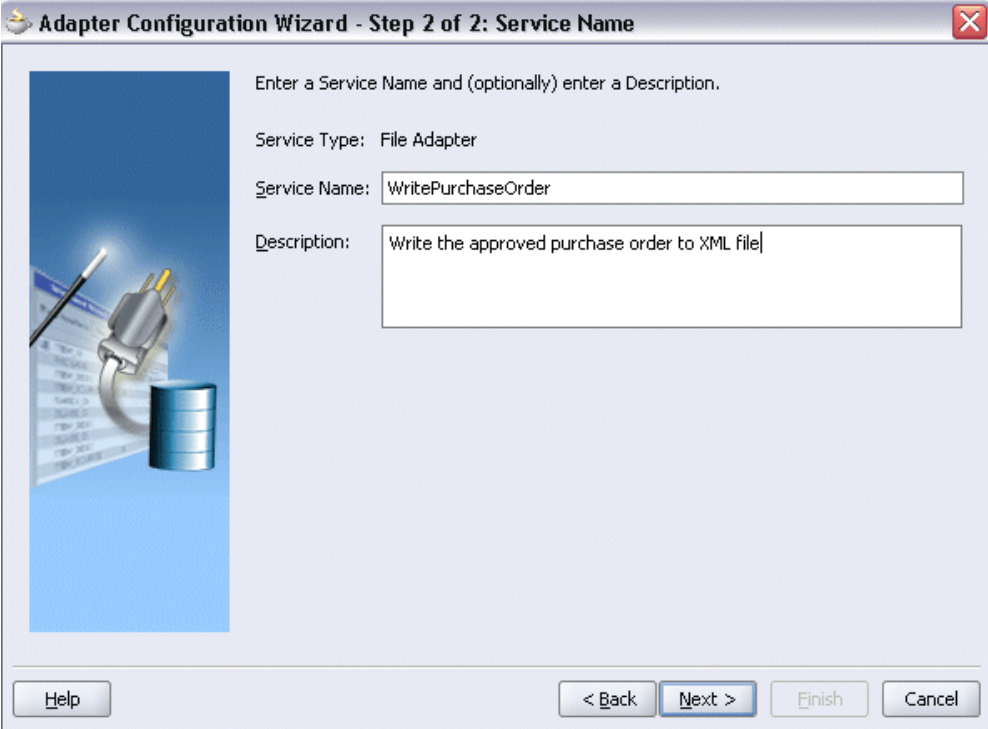
Use this step to configure a business event by writing the event data to an XML file.

### 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 with 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" is set to "File Adapter". The "Service Name" field contains "WritePurchaseOrder" and the "Description" field contains "Write the approved purchase order to XML file". At the bottom are buttons for "Help", "< Back", "Next >", "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; for example, `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:

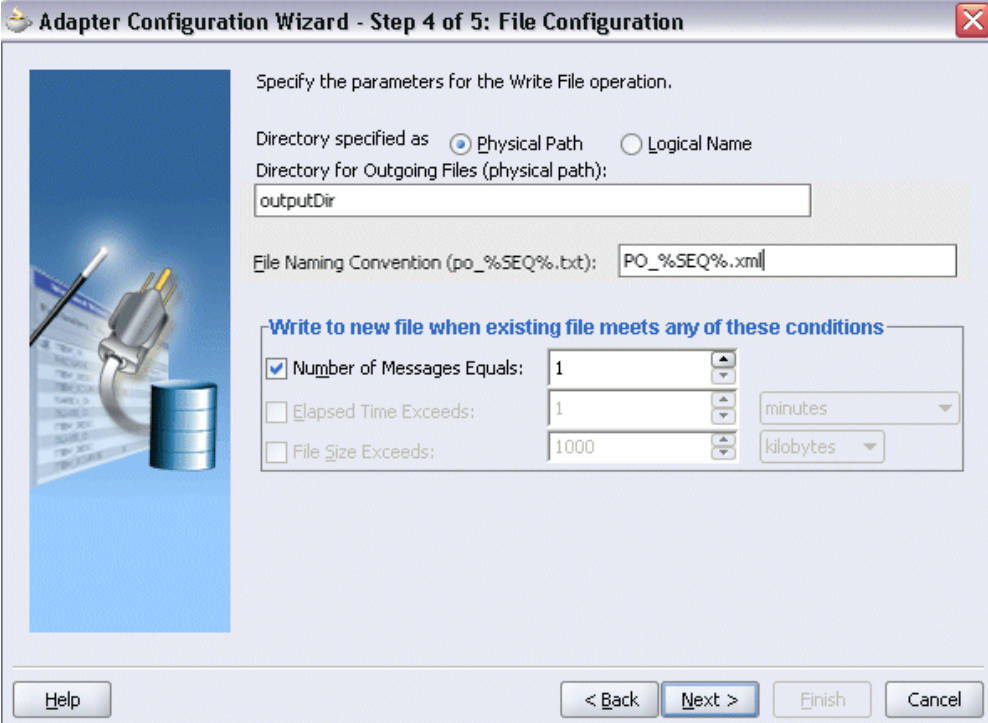
Help < Back Next > Finish Cancel

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 a Windows-style dialog box titled "Adapter Configuration Wizard - Step 4 of 5: File Configuration". On the left is a vertical blue bar with an illustration of a USB cable and a database cylinder. 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 four buttons: "Help", "< Back", "Next >", 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

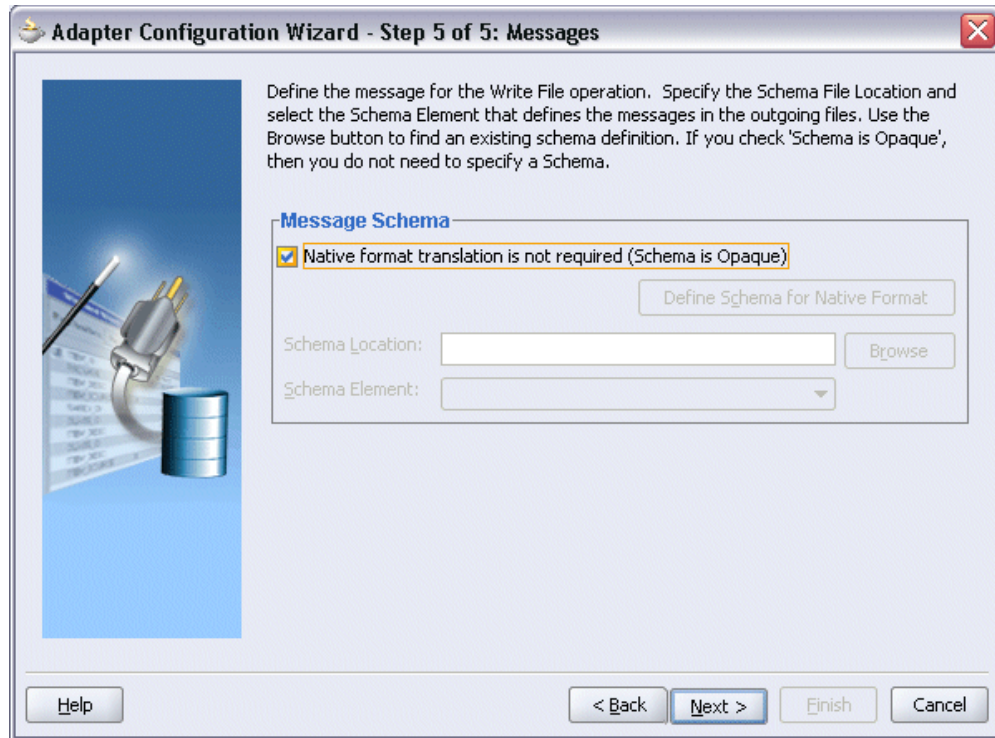
☐ Elapsed Time Exceeds: 1 minutes

☐ 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; for example, 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 panel with a blue background and a graphic of a USB cable and a database cylinder. The main area contains 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 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

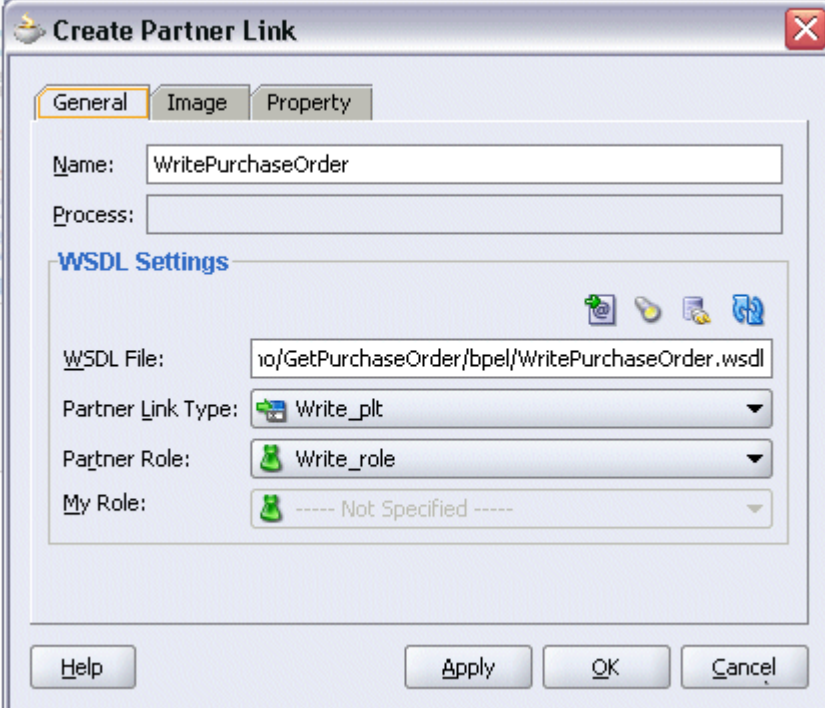
Schema Location:  Browse

Schema Element:

Help < Back Next > Finish Cancel

9. Click **Next**, 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 active. It contains the following fields and controls:

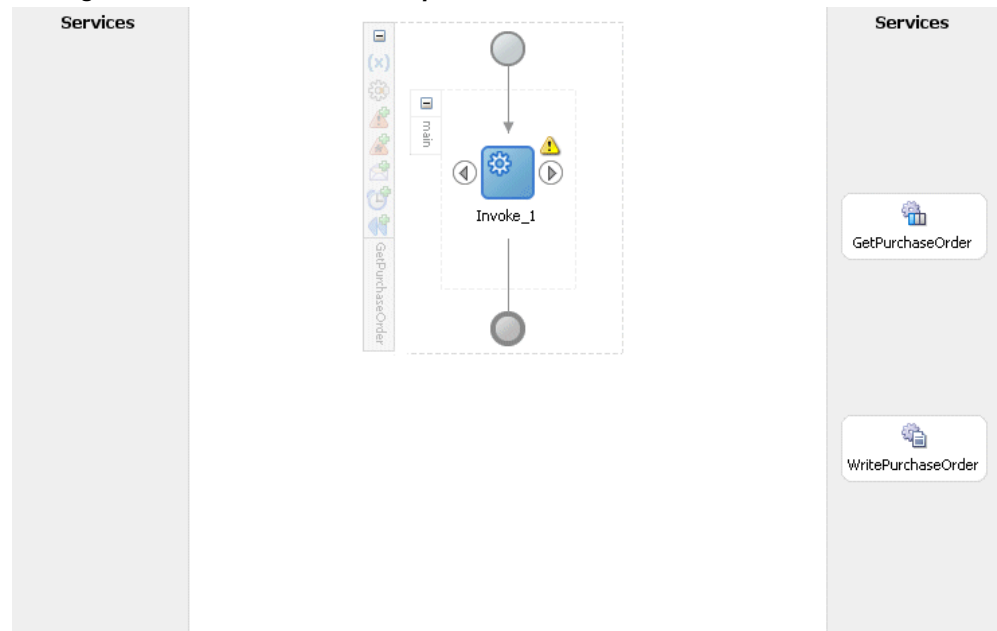
- Name:** WritePurchaseOrder
- Process:** (empty text box)
- WSDL Settings:** (section header with icons for file, key, and help)
- WSDL File:** io/GetPurchaseOrder/bpel/WritePurchaseOrder.wsdl
- Partner Link Type:** Write\_plt (dropdown menu)
- Partner Role:** Write\_role (dropdown menu)
- My Role:** ----- 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 following BPEL process diagram:

### Adding the Partner Link for File Adapter



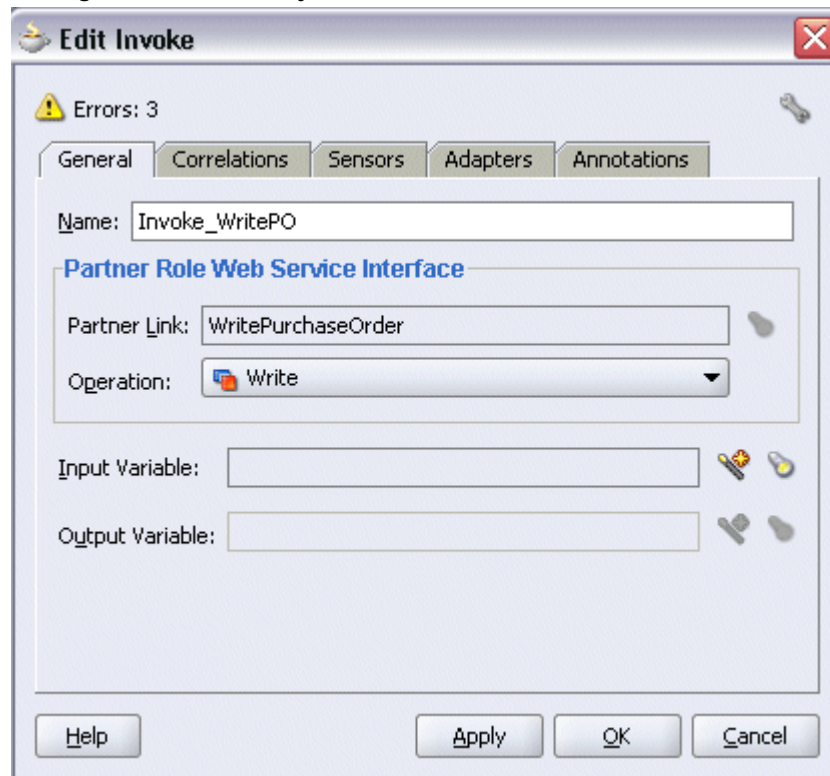
## Adding an Invoke Activity

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.

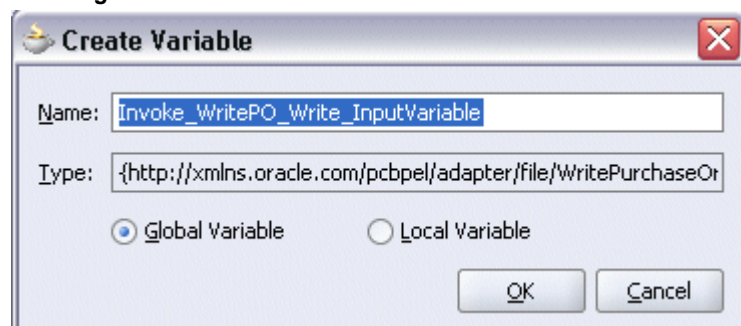
### Editing the Invoke Activity



The **Edit Invoke** dialog box is shown with the **General** tab selected. It features a tabbed interface with **General**, **Correlations**, **Sensors**, **Adapters**, and **Annotations**. At the top, it indicates **Errors: 3**. The **Name** field contains **Invoke\_WritePO**. Below this is the **Partner Role Web Service Interface** section, which includes a **Partner Link** field with **WritePurchaseOrder** and an **Operation** dropdown menu set to **Write**. There are empty fields for **Input Variable** and **Output Variable**, each with a **Create** icon (a lightbulb) to its right. At the bottom, there are **Help**, **Apply**, **OK**, and **Cancel** buttons.

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

### Creating a Variable



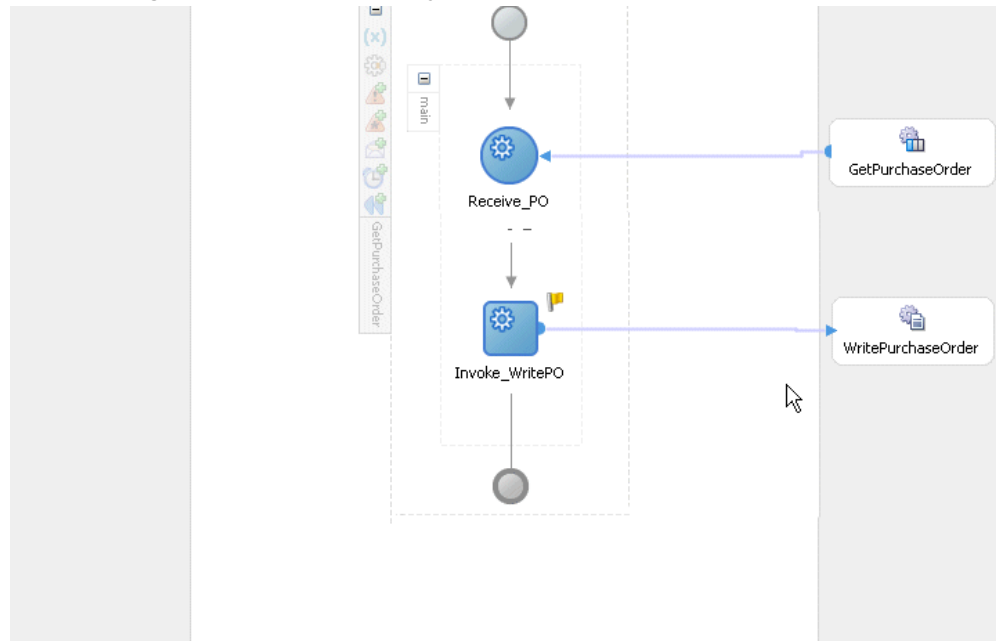
The **Create Variable** dialog box is shown. It has a **Name** field containing **Invoke\_WritePO\_Write\_InputVariable** and a **Type** field containing **{http://xmlns.oracle.com/pcbpel/adapter/file/WritePurchaseOr**. Below these fields are two radio buttons: **Global Variable** (which is selected) and **Local Variable**. At the bottom, there are **OK** and **Cancel** buttons.

4. Select **Global Variable**, 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**



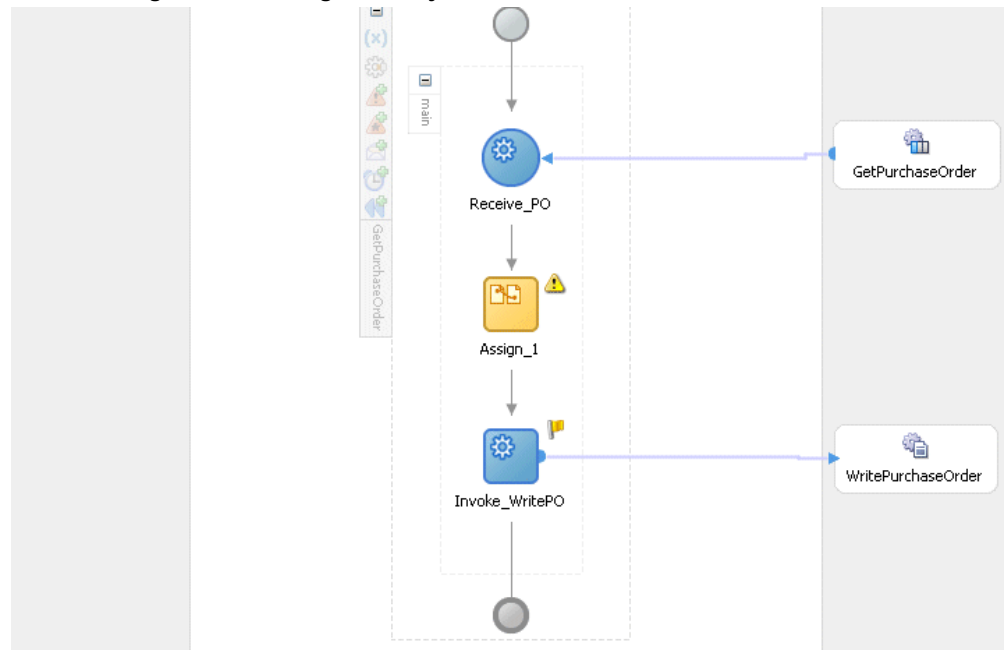
### **Adding an Assign Activity**

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

#### **To add an Assign activity:**

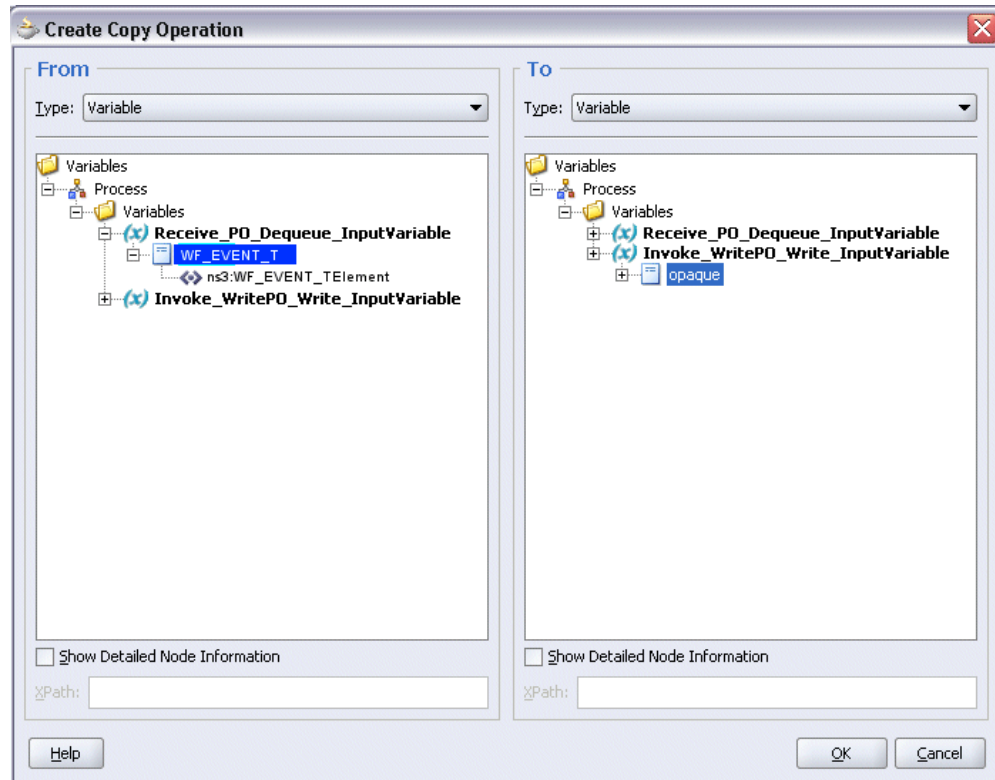
1. In JDeveloper BPEL Designer, drag and drop the **Assign** activity from the Component Palette into the Activity box of the process diagram, between the **Receive** activity and the **Invoke** activity.

### 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**, 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, then 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, then 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 a 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.



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

1. Deploy the BPEL process, page 5-25

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 5-26

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 a BPEL Process

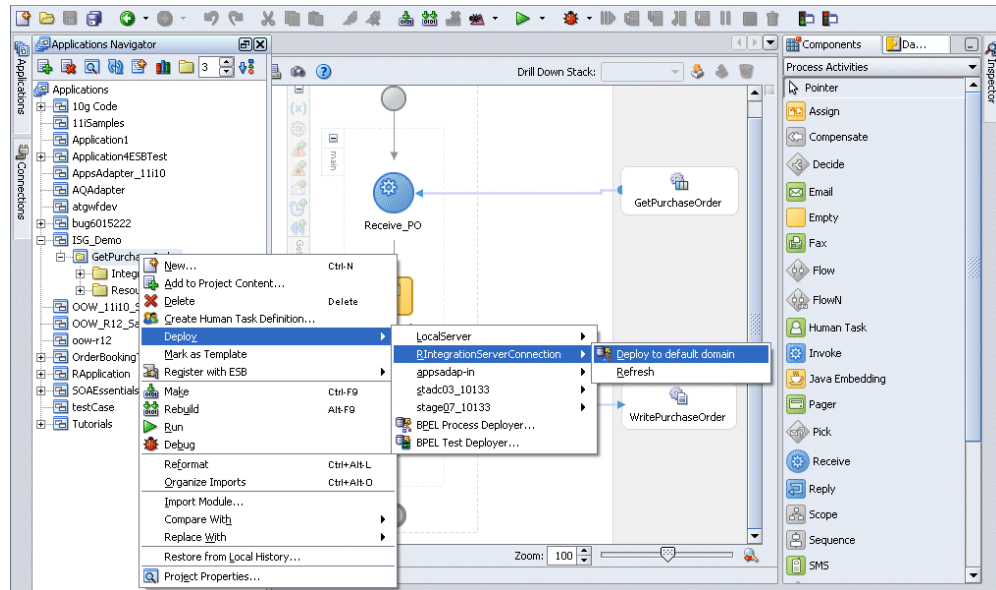
Before manually test the BPEL process, you first need to deploy it to the BPEL server.

**To deploy the BPEL process:**

1. In the Applications Navigator of JDeveloper BPEL Designer, select the **GetPurchaseOrder** project.
2. Right-click the project and select **Deploy** action. Click **Invoke Deployment Tool** and enter your BPEL Process Manager Information.

For example, select **Deploy > RIntegrationServerConnection > Deploy to Default Domain** from the menu if it is the appropriate deploy path.

### Deploying the BPEL Process

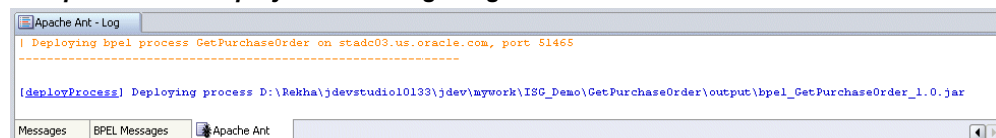


3. The Password Prompt dialog box appears.

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

The BPEL project is compiled and successfully deployed.

### 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 Applications 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 BPEL Process Manager, then select **BPEL Console**. The BPEL

console login screen appears.

2. Select **Default** in the **Domain** box. Enter the password for the default domain and click **Login** to access the console.
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: default
Dashboard		BPEL Processes	Instances
Activities			
Deployed BPEL Processes		In-Flight BPEL Process Instances	
Name		Instance	Last Modified ↑
API_1			
BPELProcess_BE			
BPELProcess_BESubscription			
BPEvent			
<a href="#">GetPurchaseOrder</a>			
Plsql			
SamplePOInboundXMLG			
TaskActionHandler			
TaskManager			
bug_66292			
psitest2			
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
Deploy New Process		Oracle BPEL Console v10.1.3.3.0	

4. Log on to Oracle Applications 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

The screenshot shows the 'Trading Partner Setup' window. The top section contains input fields for:
 

- Operating Unit: Vision Operations
- Trading Partner Type: Supplier
- Trading Partner Name: Advanced Network Devices
- Trading Partner Site: 2000 Century Way Santa Clara CA 95613-4565
- Company Admin Email: nobody@localhost

 Below these fields are two buttons: 'User Setup' and 'Code Conversion'.

The bottom section, titled 'Trading Partner Details', contains a table with the following columns: Transaction Type, Transaction SubType, Standard Code, External Transaction Type, External Transaction SubType, Direction, Map, Connection/Hub, and Protocol Type. The first row of the table is populated with the values: PO, PRO, OAG, PO, PROCESS, OUT, itg\_process\_p, DIRECT, and SOAP. The rest of the table is empty.

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.

Purchase Orders - 5789

Operating Unit: Vision Operations | Created: 07-JUN-2008 01:29:02

PO, Rev: 5789 | 0 | Type: Standard Purchase Order

Supplier: Advanced Network Devices | Site: SANTA CLARA-ERS

Ship-To: M1- Seattle Mfg | Bill-To: V1- New York City

Buyer: Stock, Ms. Pat | Status: Approved

P-Card: | Contact: | Currency: USD | Total: 150.39

Description:

Lines | Price Reference | Reference Documents | More | Agreement | Temporary Labor

Num	Type	Item	Rev	Job	Category	Description	UOM	Quantity	Price
1	Goods	CM13139			PRODUCTN.DRN	Hard Drive - 8GB	Each	1	150.393

Item: CM13139 | Hard Drive - 8GB

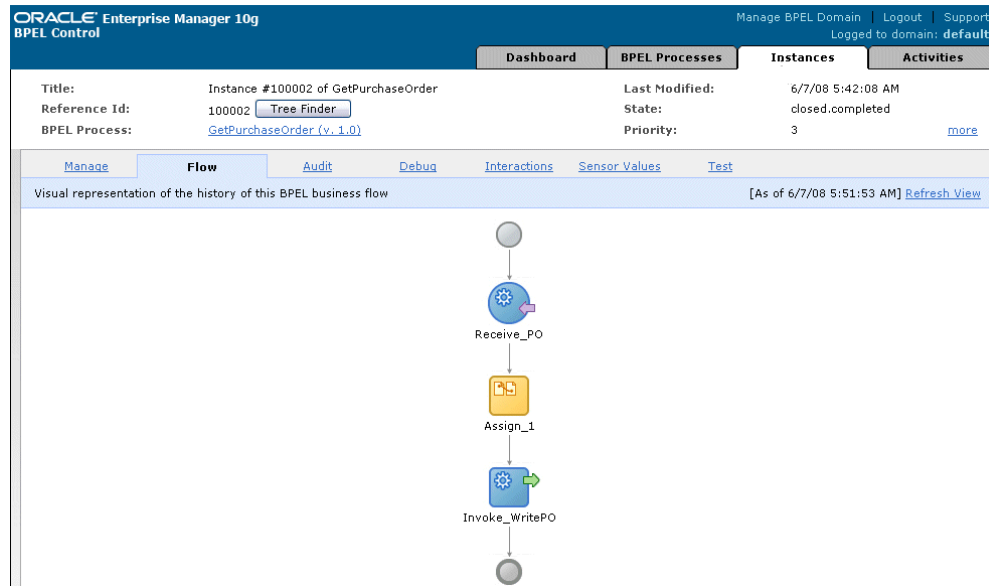
Catalog... | Currency... | Terms | Shipments | Approve...

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 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>2768</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 Applications is described in this chapter.

### Using Concurrent Program WSDL in Creating a BPEL Process 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.

After deploying the BPEL process, 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 Concurrent Program Web Services**

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, click **Deploy** in the Web Service - SOA Provider region of the interface details page to deploy the service.

Once the service is successfully deployed, a confirmation message appears on top of the interface details page.

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 web application. The left sidebar shows a tree view of various business suites, with 'Shipping Execution Common' selected. The main content area shows details for the 'Concurrent Program : Departure Shipment Notice Outbound'. It includes a search bar, a 'Printable Page' button, and a table of metadata. The 'Full Description' section explains that this program generates a Departure Ship Notice Outbound (DSNO). The 'Source Information' section lists the source file, version, and product. The 'Parameters' section contains a table with columns for Name, Type, Required, Displayed, and Description. The 'Methods' section has a 'Select All' and 'Select None' link, followed by a table with columns for Name, Internal Name, Status, and Description. A 'Show Process' button is also present.

Internal Name	Type	Product	Status	Business Entity	Scope	Interface Source
WSHDSNO	Concurrent Program	Shipping Execution Common	Active	<a href="#">Link</a>	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.

**Source Information**

Source File: patch/115/import/US/wshprg.ldt  
Source Version: 120.22  
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**

Select All | Select None

Select Details	Name	Internal Name	Status	Description
<input type="checkbox"/> Show Process	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 embed application context into SOAP envelopes for Web service authorization. These SOAHeader elements are *Responsibility Name*, *Responsibility Application Name*, *Security Group Name*, and *NLS Language*.

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 6-4

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 6-6

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 6-9

This is to synchronous read sales order details received from the trading partner.

4. Add Invoke activities, page 6-14

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 6-18

Use this step to create three Assign activities in order to:

1. To pass the order details from the output of the Synchronous Read - File Adapter service to the input of the DSNO creation.
2. To pass the payload of the concurrent program service to the input of the DSNO creation.
3. To pass the SOAHeader variables for the invocation of the DSNO concurrent program service.

For general information and basic concept of a BPEL process, see Understanding BPEL Business Processes, page A-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, 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:

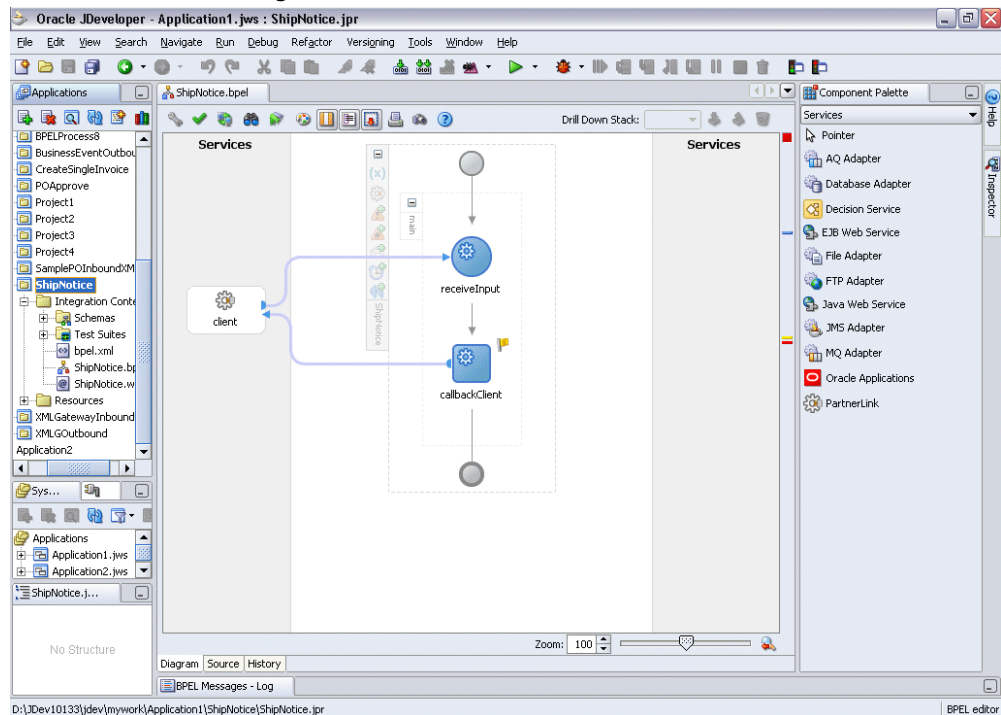
Project Directory:

Template:

7. In the **Name** field, enter a descriptive name; for example, ShipNotice.
8. From the Template list, select **Synchronous BPEL Process**, 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.

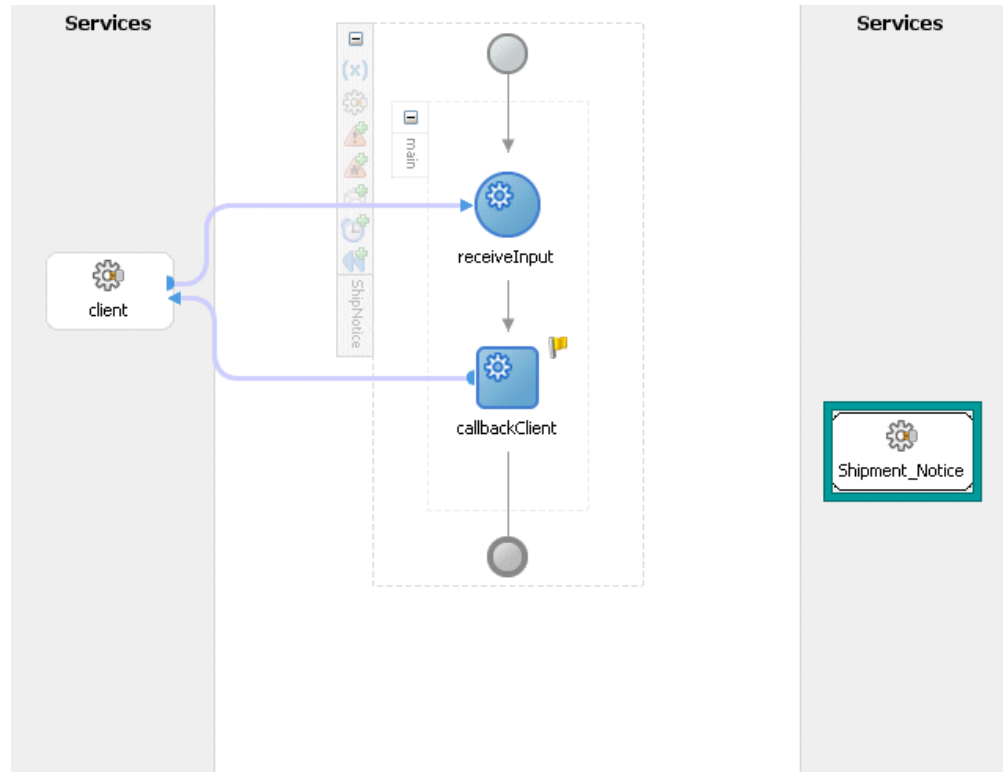
**To create a partner link for Shipment\_Notice Web service:**

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. 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.

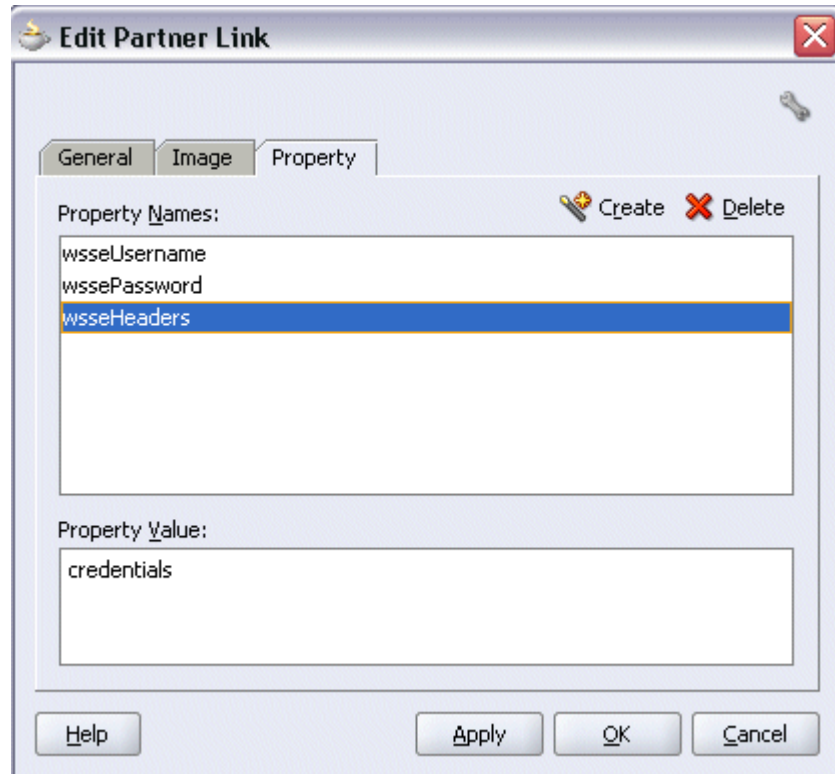
- wsseHeaders

Enter credentials as the property value.

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



### Adding Properties



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

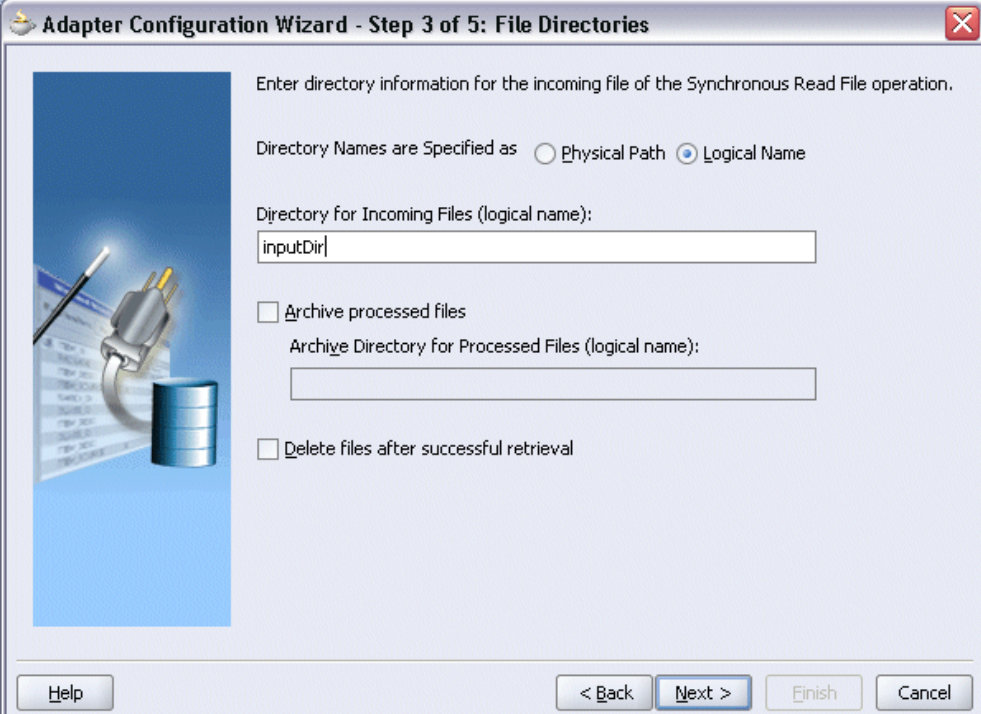
Operation Name:

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.

### Configuring the Input File



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 (logical name):

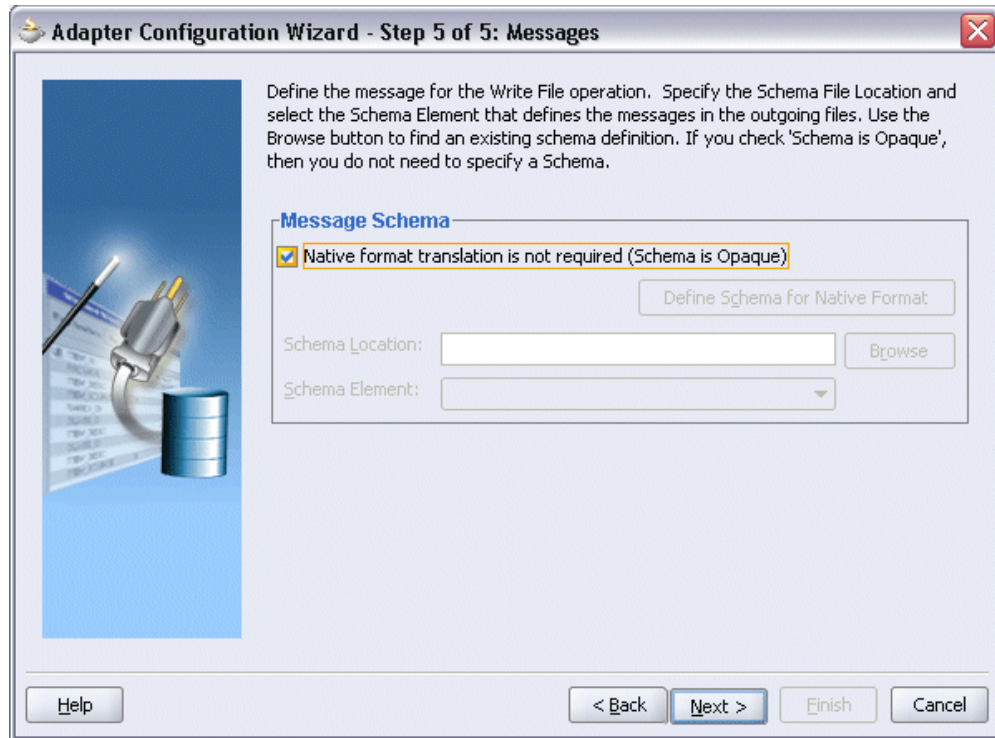
☐ Archive processed files  
Archive Directory for Processed Files (logical name):

☐ Delete files after successful retrieval

Help < Back Next > Finish Cancel

6. Select **Logical Name** radio button and specify directory for incoming files, such as 'inputDir'.  
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.
8. Select **Native format translation is not required (Schema is Opaque)** check box.

### Specifying Message Schema



The screenshot shows a dialog box titled "Adapter Configuration Wizard - Step 5 of 5: Messages". On the left is a vertical panel with a blue background, featuring an illustration of a USB cable and a database cylinder. The main area contains 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 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 "Finish", with the "Next >" button highlighted.

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

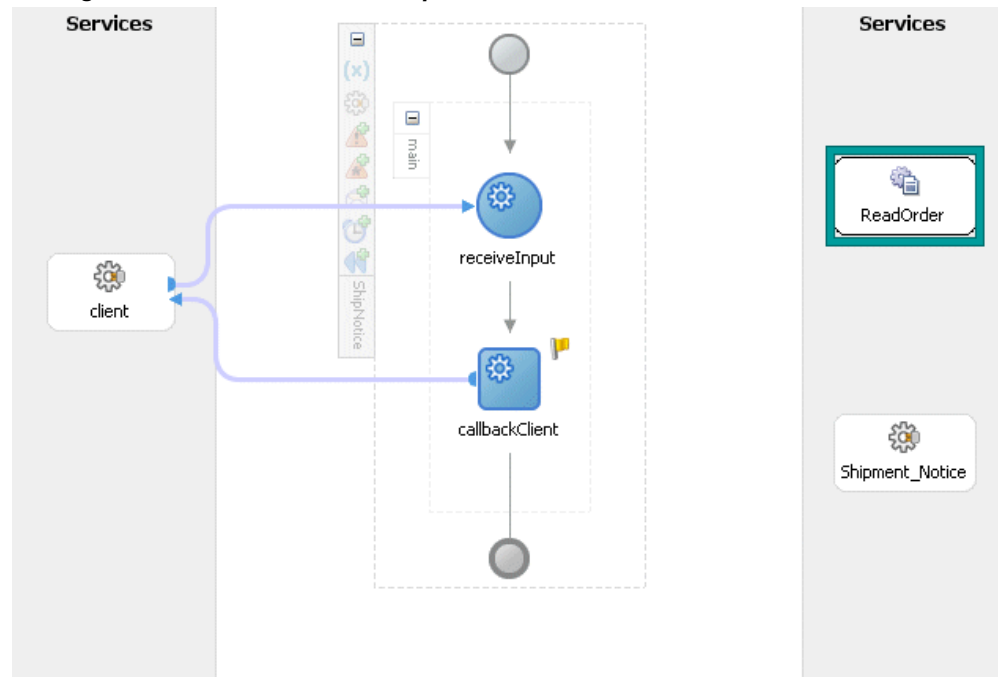
My Role: ----- Not Specified -----

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 following BPEL process diagram:

### Adding the Partner Link for File Adapter



## Adding Invoke Activities

This step is to configure two Invoke activities:

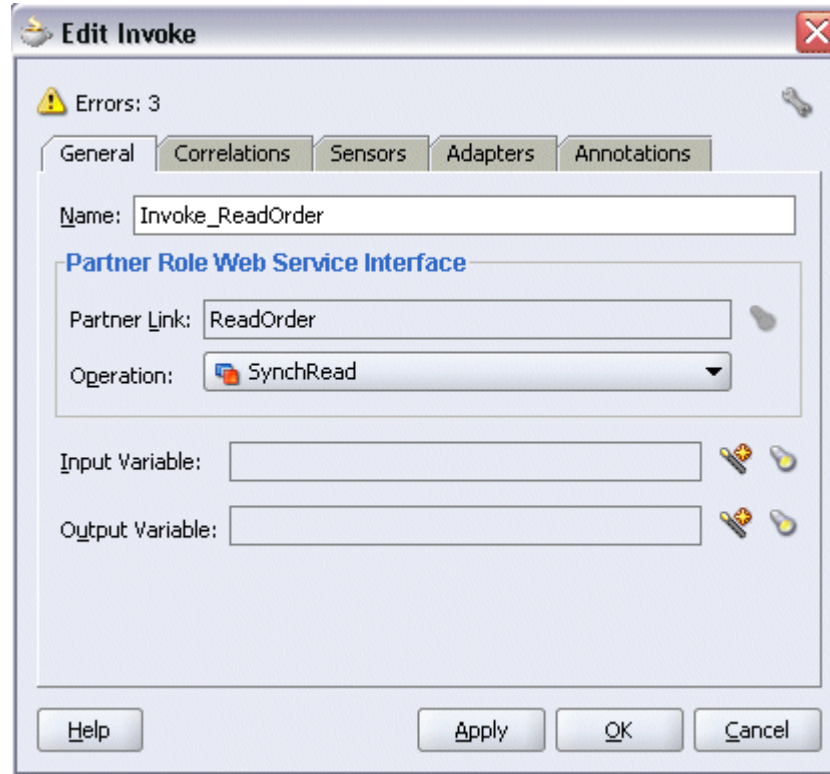
- Read order details that is passed from the first Assign activity through the `ReadOrder` partner link for File Adapter.
- Send the order header and line details received from the Assign activities to generate an outbound shipment notice (DSNO) by using the `Shipment_Notice` partner link.

### 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.

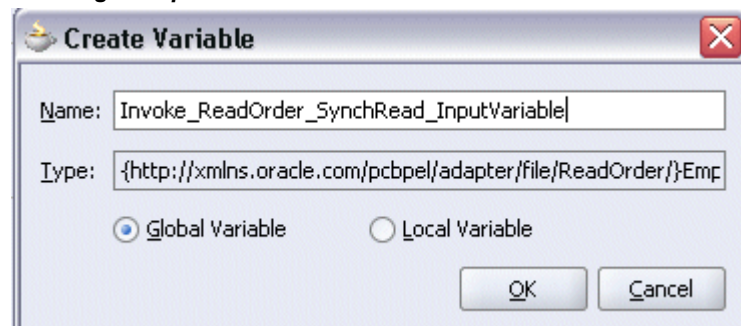


### Editing the Invoke Activity



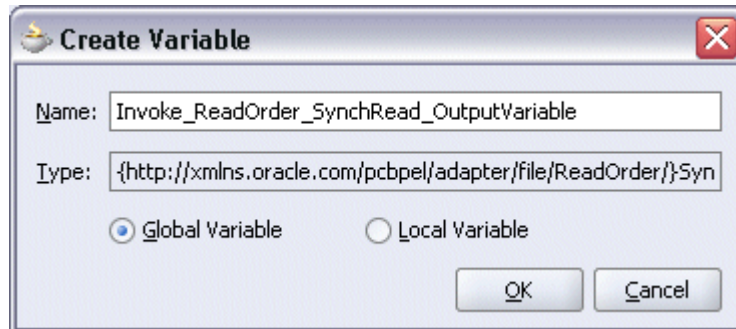
3. Enter a name for the Invoke activity, then click the **Create** icon next to the **Input Variable** field to create a new variable. The Create Variable dialog box appears.

### Creating an Input Variable



4. Select **Global Variable**, 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**, 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.

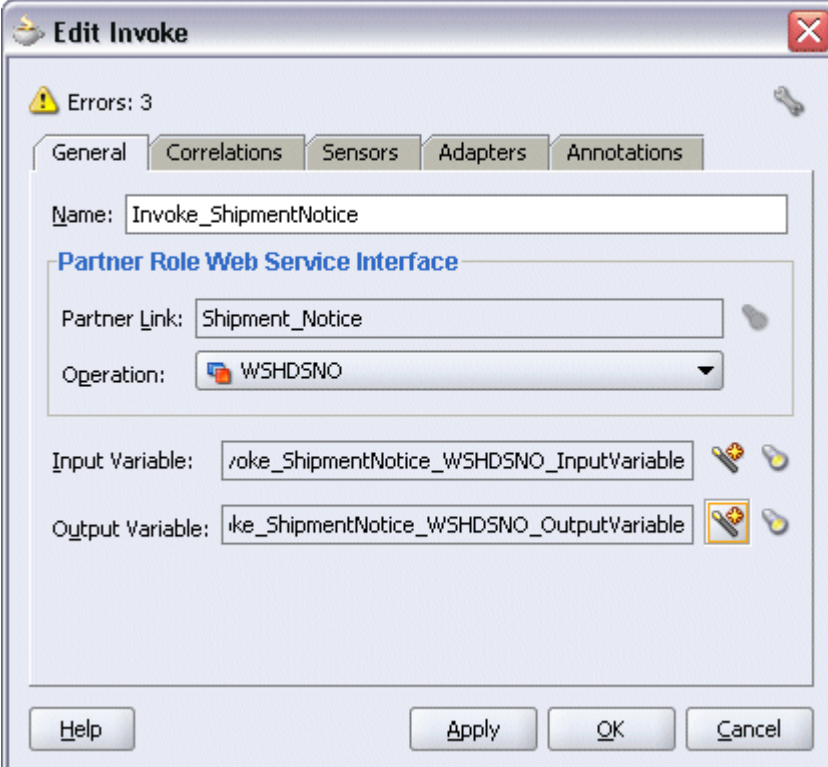
The Invoke activity appears in the process diagram.

### 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. The **Name** field contains "Invoke\_ShipmentNotice". Below this is a section titled **Partner Role Web Service Interface** containing a **Partner Link** field with "Shipment\_Notice" and an **Operation** dropdown menu set to "WSHDSNO". At the bottom of this section are **Input Variable** and **Output Variable** fields, both containing names starting with "Invoke\_ShipmentNotice\_WSHDSNO\_". Each variable field has a key icon and a lightbulb icon to its right. The bottom of the dialog has **Help**, **Apply**, **OK**, and **Cancel** buttons.

**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

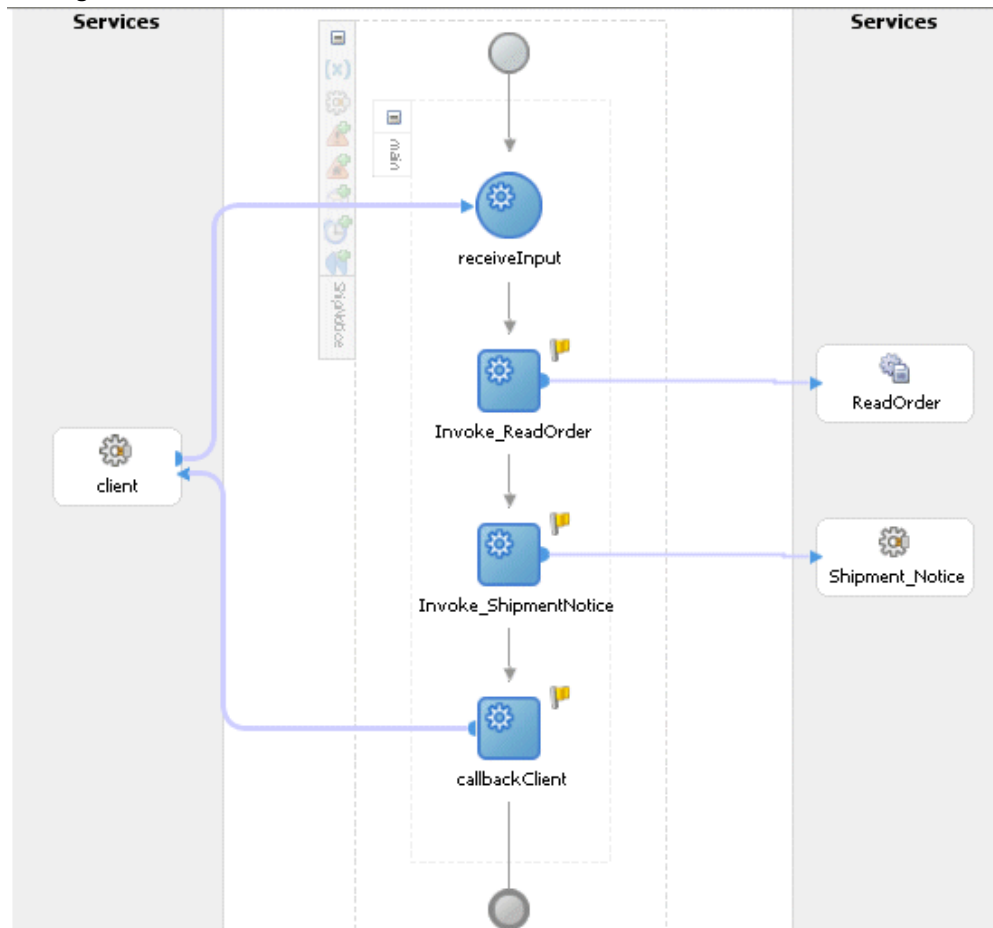
Output Variable: Invoke\_ShipmentNotice\_WSHDSNO\_OutputVariable

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 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.
2. To pass the payload of the concurrent program service to the input of the DSNO creation.
3. To pass the application context for SOAHeader in the invocation of the DSNO concurrent program service.

**To add the first 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 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 'SetOrderDetails'.
4. On the Copy Operation tab, click **Create**, 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, then navigate to **Variable > Process > Variables > Invoke\_Shipment\_Notice\_WSHDSNO\_OutVariable** and select an appropriate parameter.
  - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_Shipment\_Notice\_WSHDSNO\_InputVariable > Body > ns1:SOARequest** and select **ns3:InputParameters**. 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.

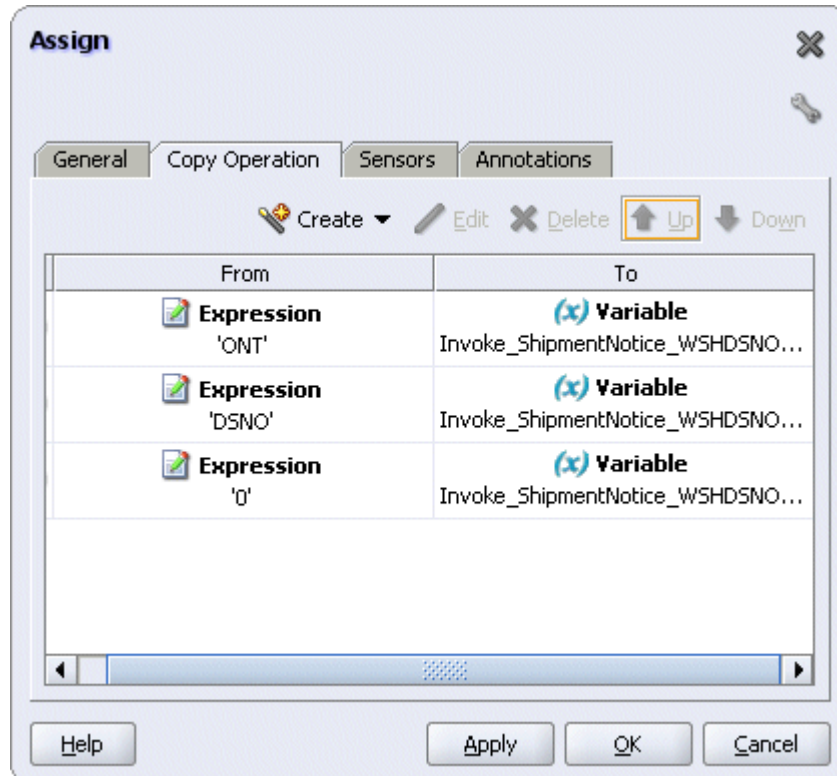
**To add the second Assign activity to pass payload of the concurrent program service to the Invoke\_ShipmentNotice Invoke activity:**

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 'SetCPdetails'.
3. Enter the first pair of parameters:
  - In the From navigation tree, select type Expression, then enter 'ONT' in the Expression box.
  - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_Shipment\_Notice\_WSHDSNO\_InputVariable > Body > ns1:InputParameters** and select **ns1:APPLICATION**. 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, then enter 'DSNO' in the Expression box.
    - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_Shipment\_Notice\_WSHDSNO\_InputVariable > Body > ns1:InputParameters** and select **ns1:PROGRAM**. 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, then enter '0' in the Expression box.
    - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_Shipment\_Notice\_WSHDSNO\_InputVariable > Body > ns1:InputParameters** and select **ns1:SUB\_REQUEST**. The XPath field should contain your selected entry.
    - Click **OK**.
  6. The Edit Assign dialog box appears.

### Assign Parameters



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

**To add the third Assign activity to pass SOAHeader variables used in the invocation of the DSNO concurrent program service:**

*Add the third Assign 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\_ShipmentNotice Invoke** activity.
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, then enter 'Order Management Super User, Vision Operations (USA)' in the Expression box.
  - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_Shipment\_Notice\_WSHDSNO\_InputVariable > header > ns1:SOAHeader** and select **ns1:ResponsibilityName**. 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, then enter 'OM' in the Expression box.
    - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_Shipment\_Notice\_WSHDSNO\_InputVariable > header > ns1:SOAHeader** and select **ns1:ResponsibilityAppName**. 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, then enter 'Standard' in the Expression box.
    - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_Shipment\_Notice\_WSHDSNO\_InputVariable > header > ns1:SOAHeader** and select **ns1:SecurityGroupName**. 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, then enter 'US' in the Expression box.
    - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > Invoke\_Shipment\_Notice\_WSHDSNO\_InputVariable > header > ns1:SOAHeader** and select **ns1:NLSLanguage**. The XPath field should contain your selected entry.
    - Click **OK**.
  7. 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 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.

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

1. Deploy the BPEL process, page 6-23

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 6-24

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

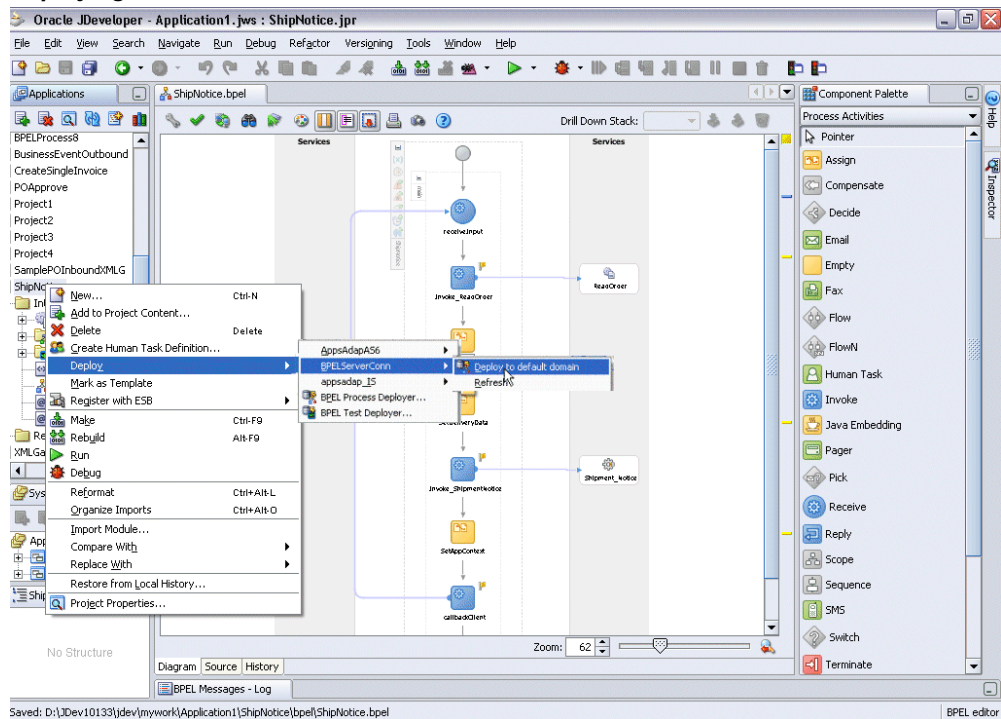
You must deploy the BPEL process (`ShipNotice.bpel`) that you created earlier before you can run it.

**To deploy the BPEL process:**

1. In the Applications Navigator of JDeveloper BPEL Designer, select the **ShipNotice** project.
2. Right-click the project and 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.

## 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 Applications 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 BPEL Process Manager, then select **BPEL Console**. The BPEL console login screen appears.
2. Select **Default** in the **Domain** box. Enter the password for the default domain and click **Login** to access the console.



Oracle BPEL console appears. The list of deployed processes is displayed.

### 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	

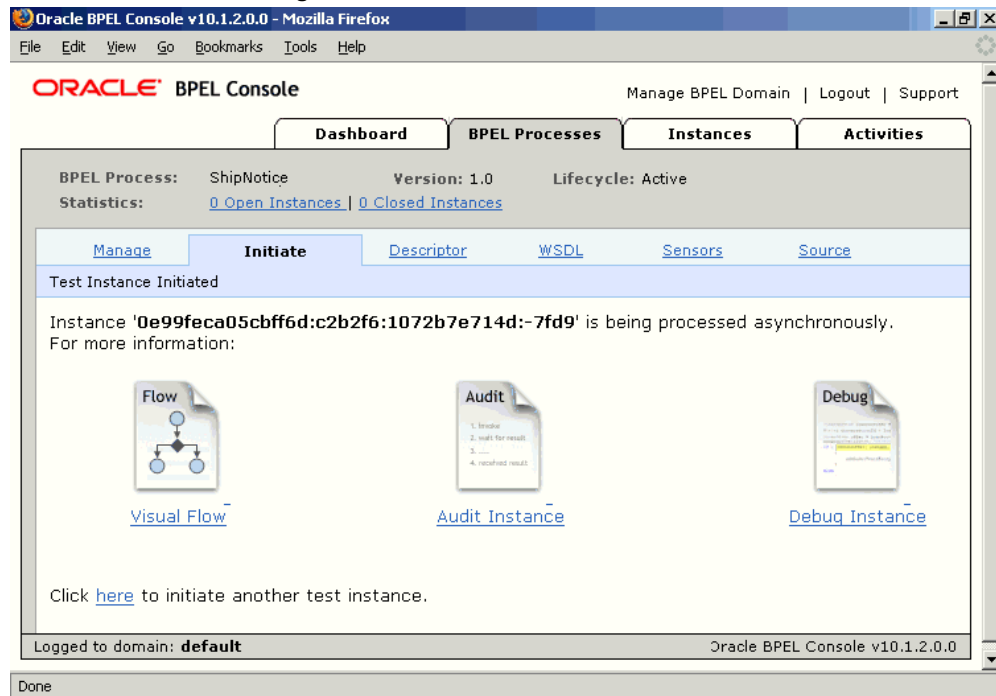
Deploy New Process

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 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 (such as 2967182).

### Verifying Records in Oracle Applications

Before verify the records in Oracle Applications, you must first ensure the concurrent request is completed successfully.

1. Log on to Oracle Applications 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, then click **Find**.

### Finding a Concurrent Request

Find Requests

☐ My Completed Requests

☐ My Requests In Progress

☐ All My Requests

☒ Specific Requests

Request ID: 2967182

Name:

Date Submitted:

Date Completed:

Status:

Phase:

Requestor:

☐ Include Request Set Stages in Query

Order By: Request ID

Select the Number of Days to View: 7

Submit a New Request... Clear Find

3. The request details are 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 Applications.

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 applications employ 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 applications 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 transform the interface definitions into 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 Applications if necessary.

To better understand how to use business service object interfaces in updating Oracle Applications, detailed design-time and run-time tasks are described in this chapter.

## Using Business Service Object WSDL in Creating a BPEL Process 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.

After deploying the BPEL process, you should notice the purchase order status changed from Incomplete to Approved.

### **Prerequisites to Create a BPEL Process Using Business Service Object Web Services**

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, 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, a confirmation message appears on top of the interface details page.

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

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

The screenshot displays the Oracle Integration Repository web application. The left sidebar shows a tree view of the 'Application Object Library' with categories like Business Event, Business Service Object, and Applications Technology. The main content area is titled 'Business Service Object : Purchase Order Service'. It includes a search bar and a 'Print' button. The interface is divided into several sections: 'Full Description' (describing the service as 'Approves a list of purchase orders'), 'Web Service - Web Service Provider' (providing abstract and deployed WSDL URLs), 'Source Information' (listing source file, version, product, and implementation), and 'Methods' (a table of service methods).

Select	Internal Name	Status	Description
<input type="checkbox"/>	Show updatePurchaseOrder	Active	Updates a PurchaseOrder.
<input type="checkbox"/>	Show acknowledgePurchaseOrder	Active	Acknowledges a purchase order.
<input type="checkbox"/>	Show approvePurchaseOrder	Active	approves a single purchase order.
<input type="checkbox"/>	Show approvePurchaseOrders	Active	Approves a list of purchase orders.
<input type="checkbox"/>	Show createPurchaseOrder	Active	Creates a new PurchaseOrder.
<input type="checkbox"/>	Show deletePurchaseOrder	Active	Deletes a PurchaseOrder.

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 scenario, the following design-time tasks are discussed in this chapter:

### 1. Create a new BPEL project, page 7-4

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 7-6

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 7-14

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 7-17

Use this step to create two Assign activities in order to:

1. Pass the purchase order details read from the File Adapter as an input to the Invoke activity for PurchaseOrderService Web service.

**Note:** You also need to populate certain variables in the BPEL process for SOAHeader elements to pass values that may be required to embed application context into SOAP envelopes for Web service authorization. These SOAHeader elements are *Responsibility Name*, *Responsibility Application Name*, *Security Group Name*, and *NLS Language*.

Detailed information on how to set SOAHeader for the SOAP request, see *Assign a SOAHeader Activity*, page 6-21.

2. 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 A-1 and *Oracle BPEL Process Manager Developer's Guide* for details.

## Creating a New BPEL Project

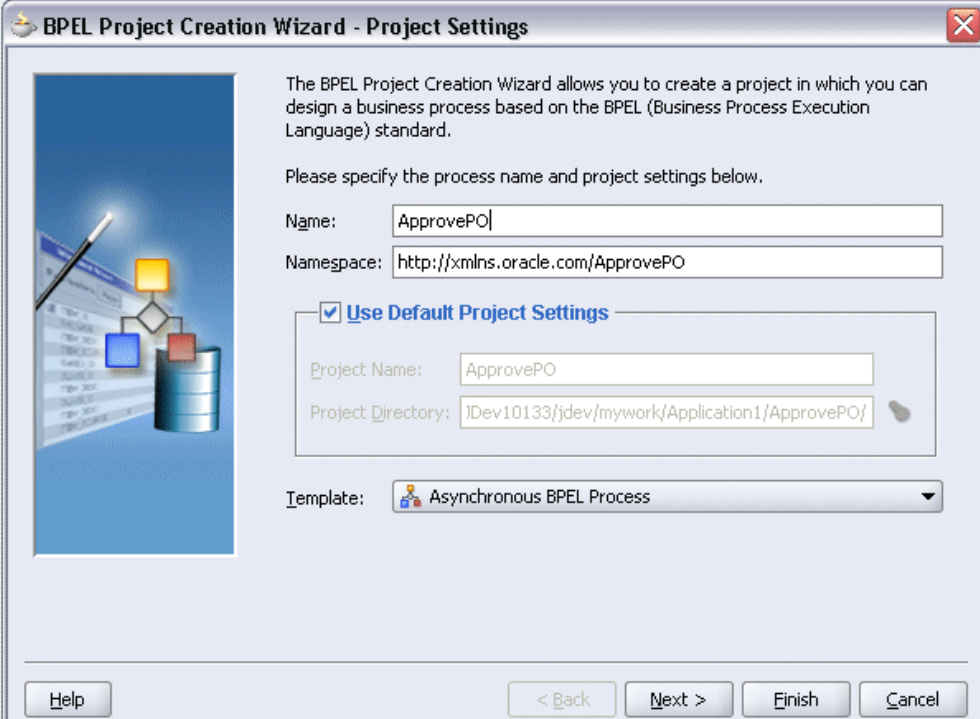
Use this step to create a new BPEL project that will contain various BPEL process activities.

**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, 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:

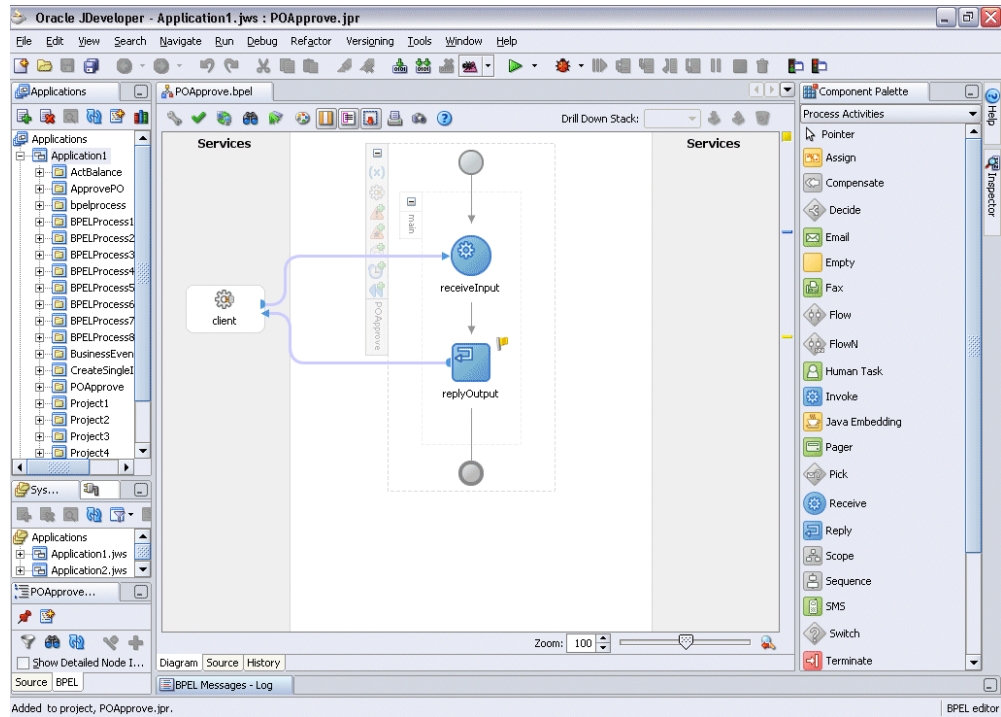
Template:

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

7. In the **Name** field, enter a descriptive name; for example, ApprovePO.
8. From the Template list, select **Synchronous BPEL Process**, 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. 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

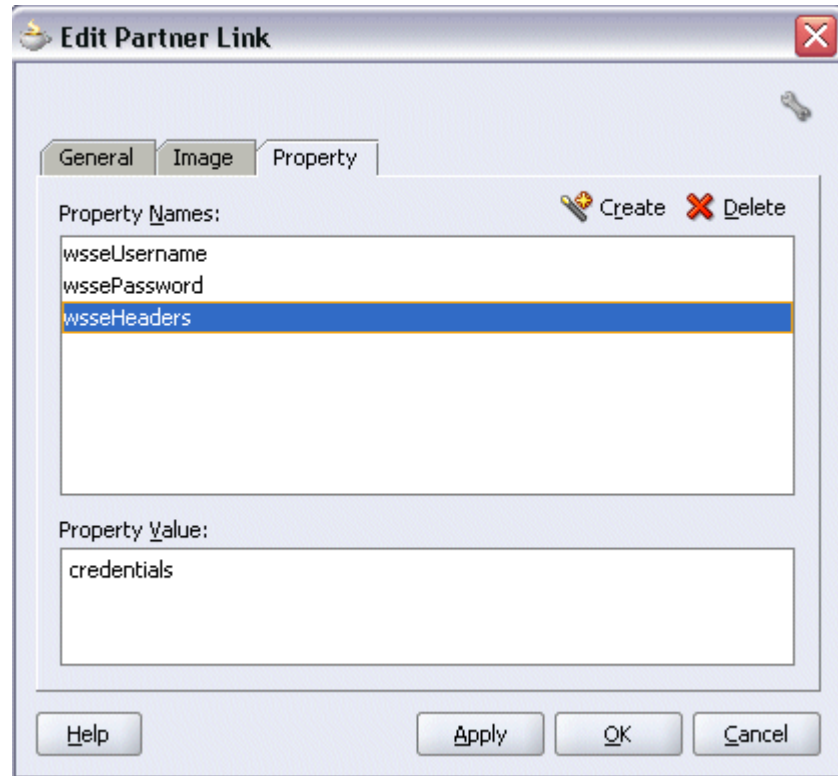
Specify the corresponding password for the username to be passed in the Property Value box.

- wsseHeaders

Enter `credentials` as the property value.

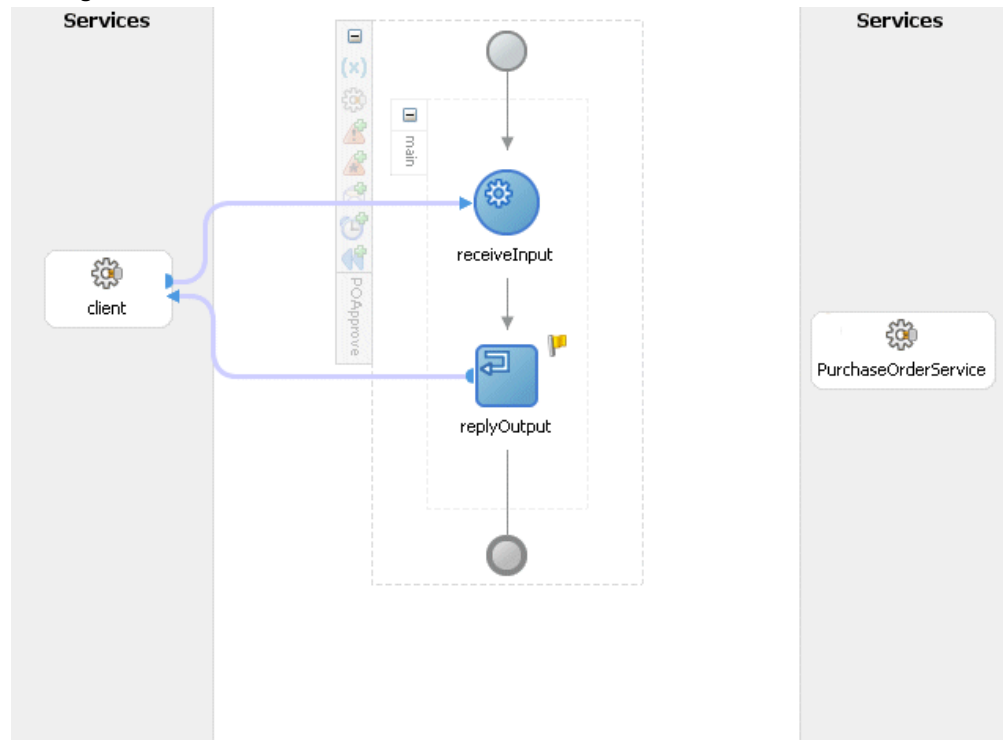
Click **Apply** to save the selected property values.

#### 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 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

**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:

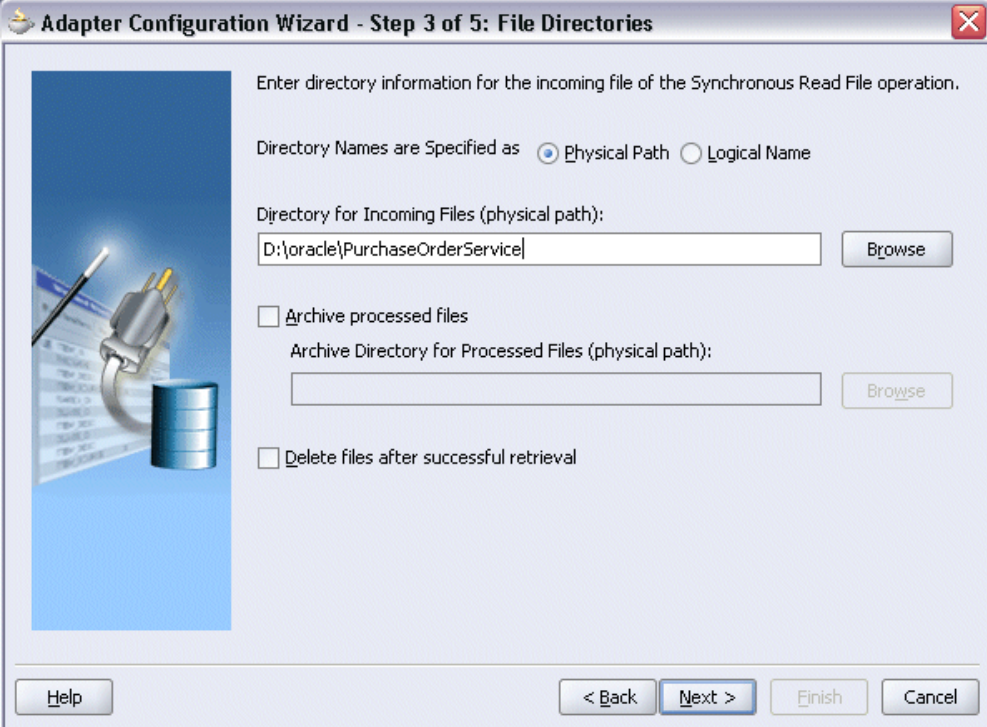
Help < Back Next > Finish Cancel

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



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

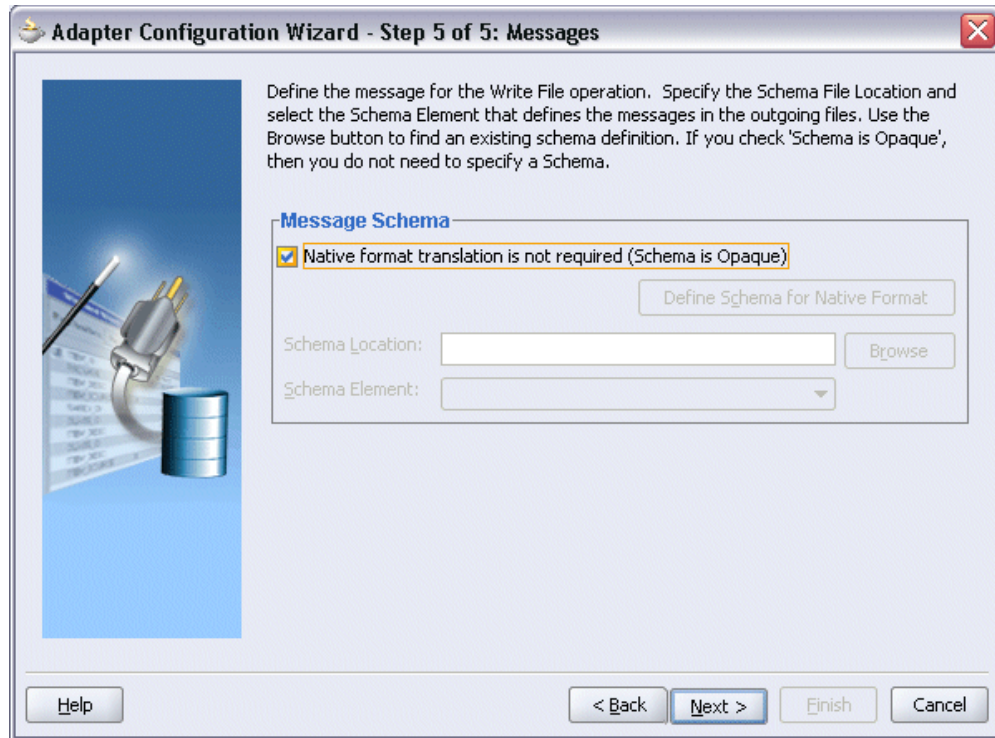
Directory for Incoming Files (physical path):  
D:\oracle\PurchaseOrderService

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

☐ Delete files after successful retrieval

6. Select **Physical Path** radio button and 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 'Input.xml'. 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 dialog box titled "Adapter Configuration Wizard - Step 5 of 5: Messages". On the left is a graphic of a USB cable and a database cylinder. The main text area contains instructions: "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 is a section titled "Message Schema" containing a checked checkbox "Native format translation is not required (Schema is Opaque)". To the right of the checkbox is a disabled button "Define Schema for Native Format". Below the checkbox are two fields: "Schema Location:" with a text box and a "Browse" button, and "Schema Element:" with a dropdown menu. At the bottom are buttons for "Help", "< Back", "Next >", "Finish", 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

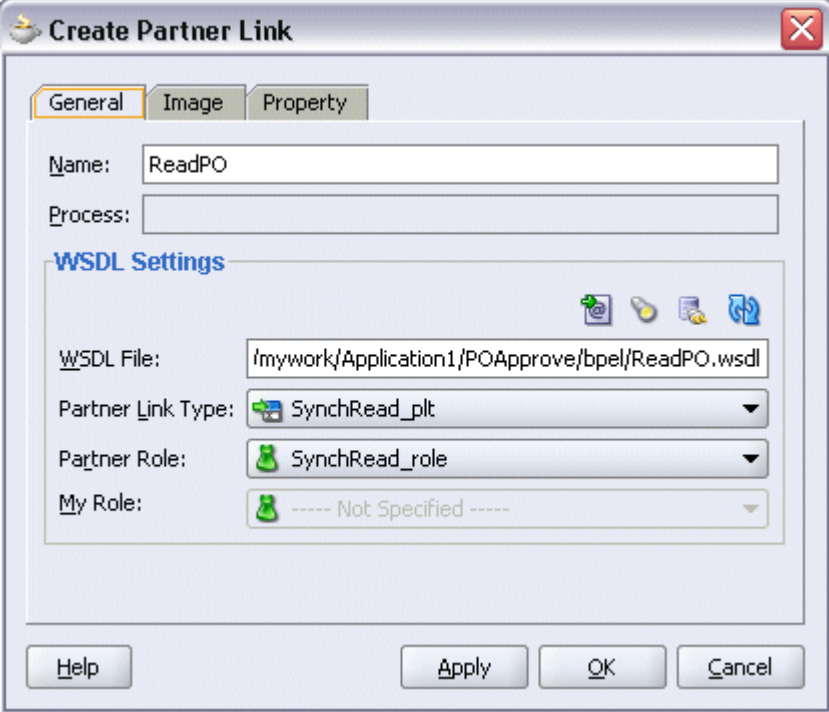
Schema Element:

Help < Back Next > Finish Cancel

9. Click **Next**, 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:

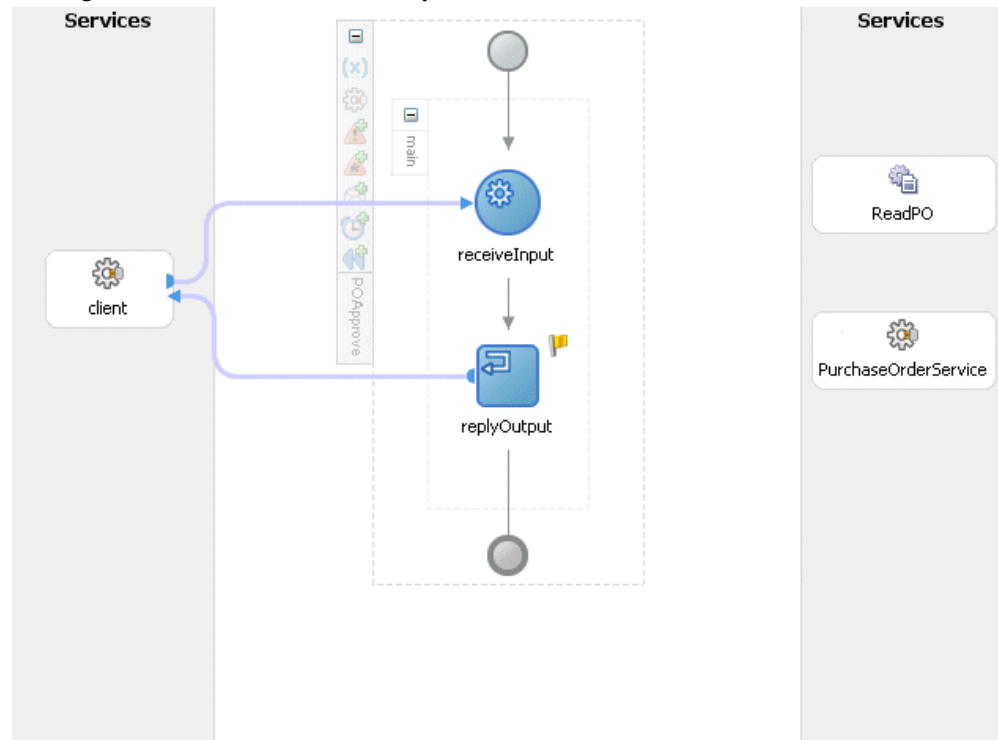
- Name:** ReadPO
- Process:** (empty text box)
- WSDL Settings:** A section with a toolbar containing icons for file operations (add, delete, save, open, print, help).
- WSDL File:** /mywork/Application1/POApprove/bpel/ReadPO.wsdl
- Partner Link Type:** SynchRead\_plt (dropdown menu)
- Partner Role:** SynchRead\_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 ReadPO Partner Link appears in the following BPEL process diagram:

### Adding the Partner Link for File Adapter



## Adding an Invoke activity

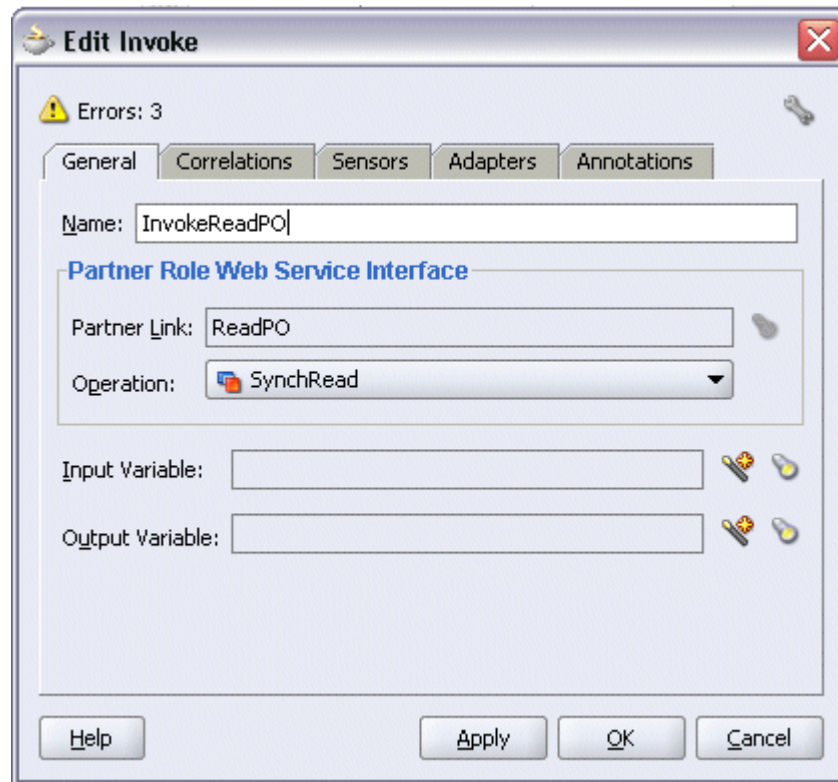
This step is to configure two Invoke activities:

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:

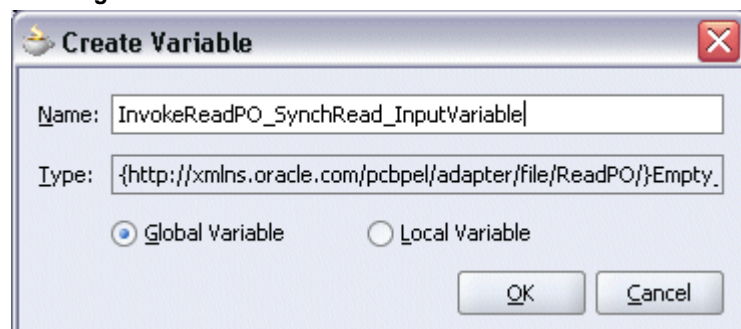
1. In JDeveloper BPEL Designer, drag and drop the **Invoke** activity from the Component Palette into the Activity box of the process diagram, between the **Receive** and **Reply** activities.
2. Link the Invoke activity to the `ReadPO` service. The Invoke activity will send event data to the partner link. The Edit Invoke dialog box appears.

### Editing the Invoke Activity



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

### Creating a Variable

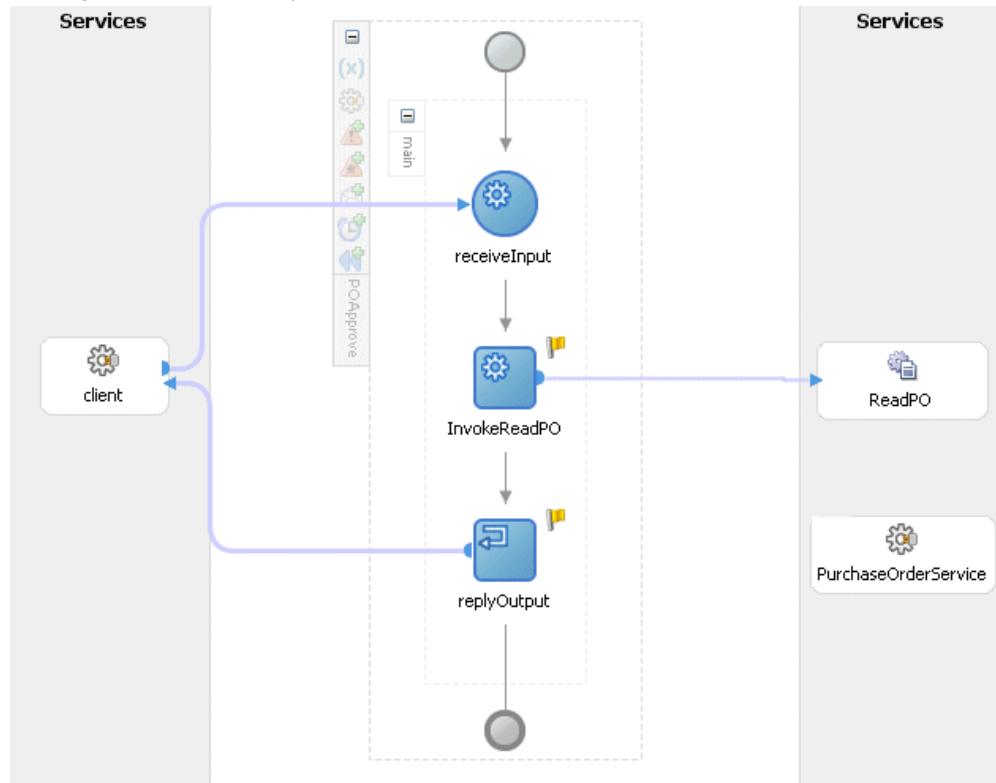


4. Select **Global Variable**, then enter a name for the variable. You can also accept the default name. Click **OK**.
5. Enter a name for the Invoke activity, then click the **Create** icon next to the **Output Variable** field to create a new variable. The Create Variable dialog box appears.

6. Select **Global Variable**, 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**

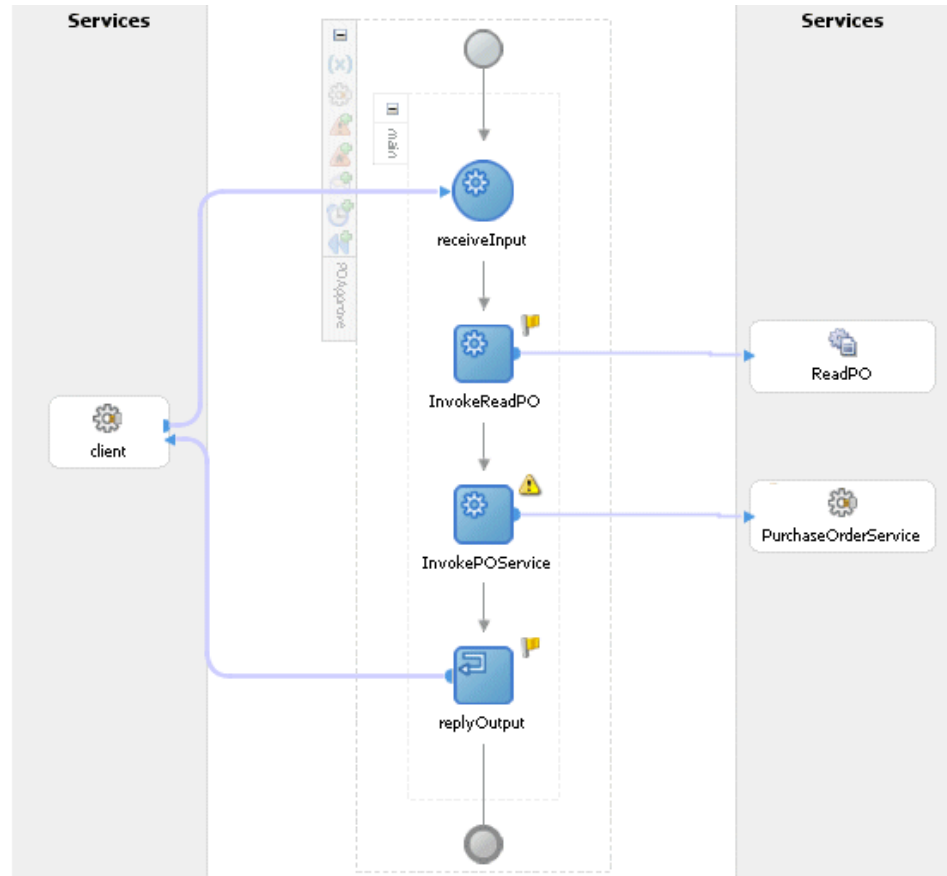


#### **To add an Invoke activity for PurchaseOrderService Partner Link:**

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' and then 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 pass the purchase order details read from the File Adapter as an input to the first Invoke activity for PurchaseOrderService partner link.

**Note:** You also need to populate certain variables in the BPEL process for SOAHeader elements to pass values that may be required to embed application context into SOAP envelopes for Web service authorization. These SOAHeader elements are *Responsibility Name*, *Responsibility Application Name*, *Security Group Name*, and *NLS Language*.

Detailed information on how to set SOAHeader for the SOAP request, see Assign a SOAHeader Activity, page 6-21.

2. To pass single purchase order approval information to the requestor through the dummy Reply activity.

**To enter the first 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 **Receive** activity and the **Invoke** 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 'SetPOApproval'.
4. On the Copy Operation tab, click **Create**, 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 > inputVariable > Payload > client:POApproveProcessRequest** and select **client:input**. The XPath field should contain your selected entry.
  - In the To navigation tree, select type Variable, then expand the **Variables** node by clicking the plus sign next to it to select an appropriate value. 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 second Assign activity to reply single purchase order approval information to requestor:**

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, expand the **Variables** node by clicking the plus sign next to it to select an appropriate value. The XPath field should contain your selected entry.
  - In the To navigation tree, select type Variable, then navigate to **Variable > Process > Variables > outputVariable > payload > client:POApproveProcessResponse** and select **client:result**. The XPath field should contain your selected entry.
  - 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.

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

1. Deploy the BPEL process, page 7-19

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 7-20

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:**

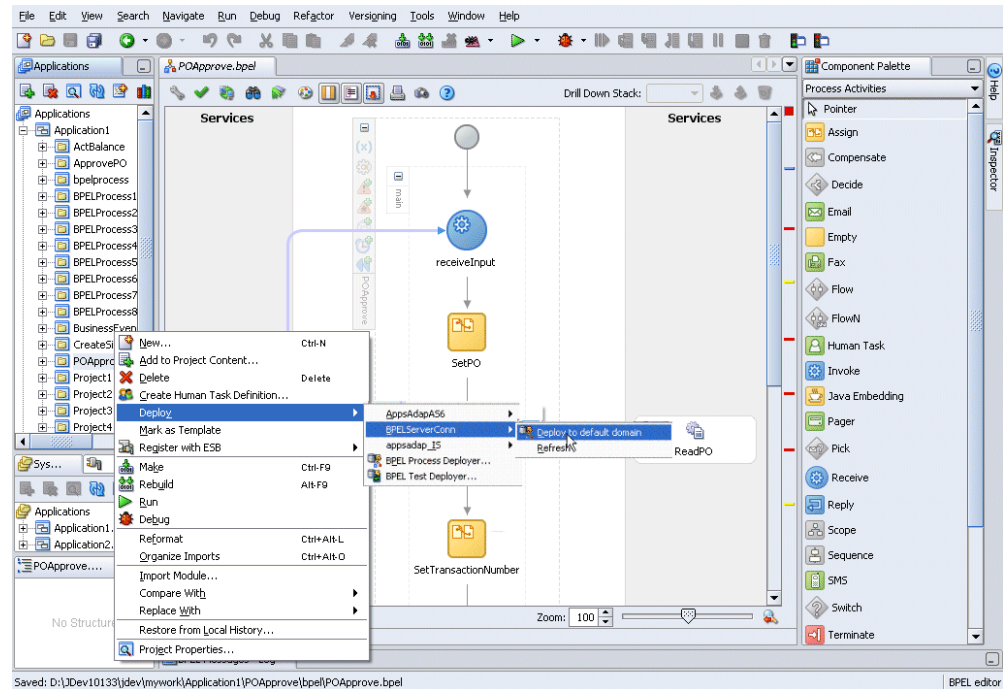
1. In the Applications Navigator of JDeveloper BPEL Designer, select the **POApprove** project.
2. Right-click the project and select **Deploy** from the menu.

Click on **Invoke Deployment Tool** and enter your BPEL Process Manager

information.

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



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.

## 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 Applications to manually initiate the processes.

### To test the BPEL process:

1. Log into Oracle BPEL Process Manager, then select **BPEL Console**. The BPEL console login screen appears.
2. Select **Default** in the **Domain** box. Enter the password for the default domain and



click **Login** to access the console.

3. In the BPEL console, confirm that 'POApprove' has been deployed.
4. Click the ApprovePO link to open the Initiate tab
5. Enter 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 a purchase order is approved successfully.

### Validating the Process in Oracle Applications

Additionally, you can validate the BPEL process in Oracle Applications. Log on to Oracle Applications with 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'.

#### Validating the PO Status

The screenshot displays the Oracle Applications 'Purchase Orders' form. The title bar indicates the environment is 'atqzdb2: Patch Test Env'. The form header shows the following details:

- PO, Rev: 4449
- Supplier: Advanced Network Devices
- Ship-To: M1- Seattle
- Buyer: Steck, Ms. Pat
- Type: Standard Purchase Order
- Site: SANTA CLARA-ERS
- Bill-To: V1- New York City
- Status: Approved
- Created: 19-JUL-2006 02:34:20
- Contact: [blank]
- Currency: USD
- Total: 150.39

Below the header, there are tabs for 'Lines', 'Price Reference', 'Reference Documents', 'More', and 'Approval'. The 'Lines' tab is active, showing a table with the following data:

Num	Type	Item	Rev	Category	Description	UOM	Quantity	Price	Promised
1	Goods	CM13139		PRODUCT/DRI	Hard Drive - 8GB	Each	1	150.393	20-JUL-2007 00:01

At the bottom of the form, there is a search bar with 'Item' set to 'CM13139' and 'Hard Drive - 8GB'. Below the search bar are buttons for 'Catalog...', 'Currency...', 'Terms', 'Shipments', and 'Approve...'.



---

# 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 8-2
- Downloading composite services, page 8-2
- Modifying and deploying BPEL processes, page 8-4

For general information and basic concept of a BPEL process, see Understanding BPEL Business Processes, page A-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 screenshot shows the Oracle Integration Repository interface. On the left is a navigation tree with categories like Business Event, Financials, Lease Management, Business Service Object, Applications Technology, Application Object Library, Service Suite, Depot Repair, Supply Chain Management, Composite - BPEL, Order Management Suite, Order Entry (selected), Supply Chain Management, Concurrent Program, EDI, Interface View, Java, Open Interface, PL/SQL, Service Data Object, Web Service, and XML Gateway Map. The main content area is titled 'Composite Service BPEL : Test Process'. It displays metadata: Internal Name (oracle.apps.oe.TestBPELProcess), Type (Composite - BPEL), Product (Order Entry), Status (Active), Business Entity (Sales Order), Scope (Public), and Interface Source (Oracle). Below this is a 'Full Description' section stating 'This is Abhishek's Test BPEL File.' and a 'BPEL Files' section with links for 'XML Representation' and 'Abstract WSDL'. The 'Abstract WSDL' link points to a URL: <http://rws60066rems.us.oracle.com:8040/webservices/SOAPProvider/bpel/oe/testbpeprocess/?wsdl>. Search and Printable Page buttons are present at the top right and bottom right of the main content area.

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 8-2.

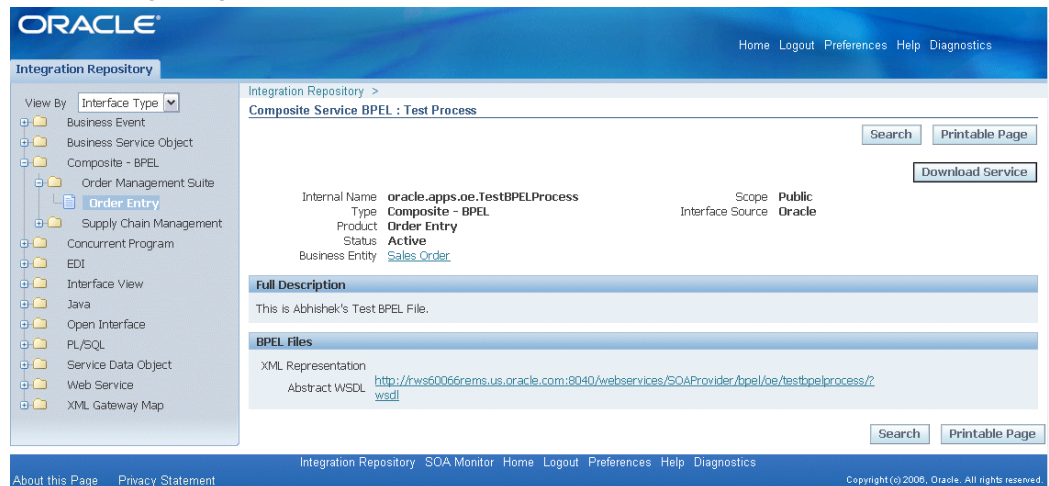
## Downloading Composite Services

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 8-4.

#### 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 A-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.

### 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.

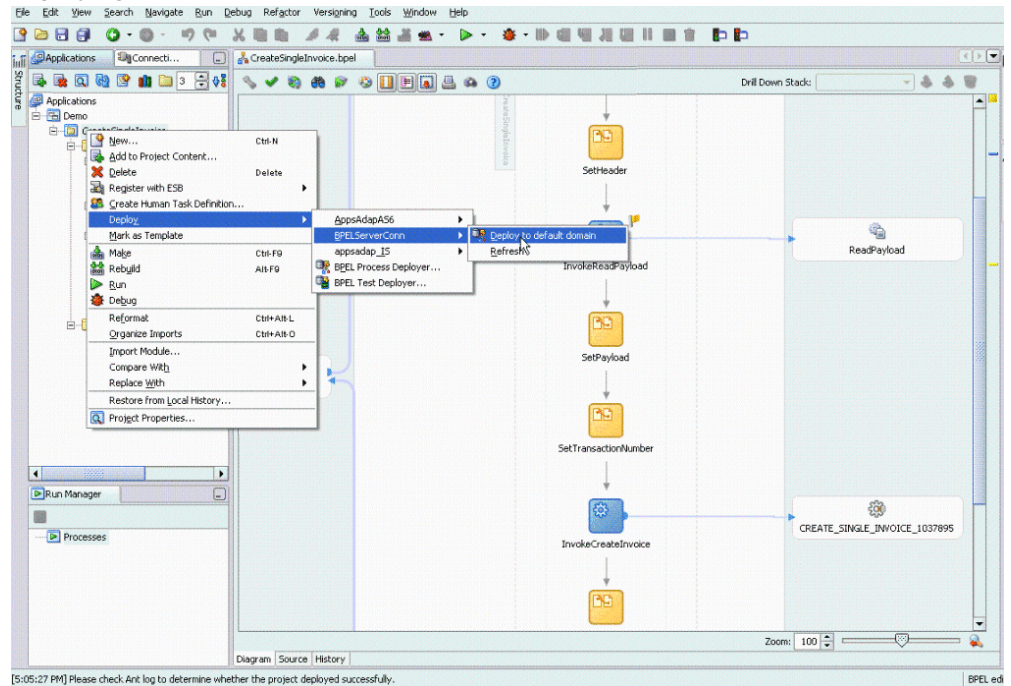
### To deploy a BPEL process:

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.





---

## 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 9-2
- Calling Back to Oracle E-Business Suite With Web Service Response, page 9-28
- Invoking Web Services, page 9-30
- Managing Errors, page 9-36
- Testing Web Service Invocation, page 9-37
- Troubleshooting Web Service Invocation Failure, page 9-43
- Extending Seeded Java Rule Function, page 9-48
- Other Invocation Usage Considerations, page 9-54

## Web Service Invocation Using Service Invocation Framework

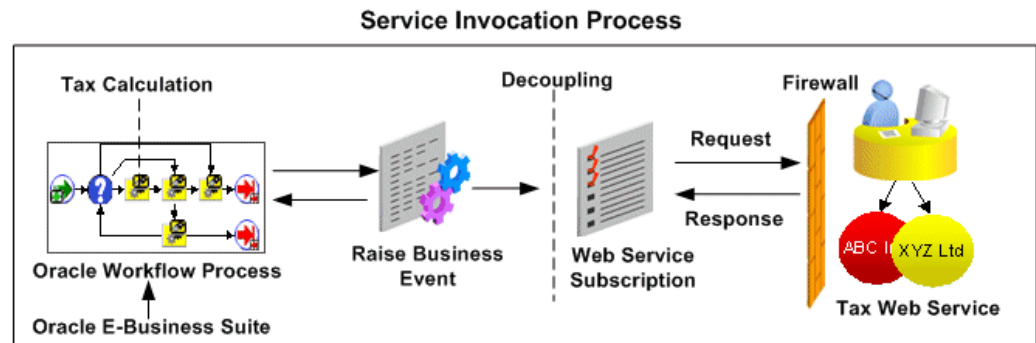
Service invocation framework provides an infrastructure allowing developers to interact with Web services through WSDL descriptions and to invoke Web services from Oracle E-Business Suite.

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 9-3
- Defining Web Service Invocation Metadata, page 9-5
  1. Creating a Web Service Invoker Business Event, page 9-6
  2. Creating Local and Error Event Subscriptions to the Invoker Event, page 9-8
  3. Creating a Receive Event and Event Subscription (Optional), page 9-17
- Understanding Web Service Input Message Parts, page 9-21
- Supporting WS-Security, page 9-26

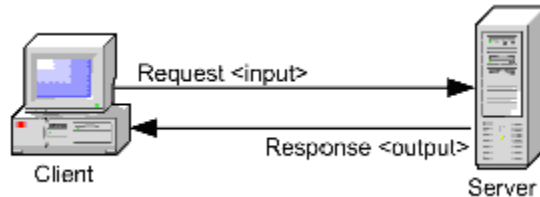
## Understanding Message Patterns in WSDL

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:

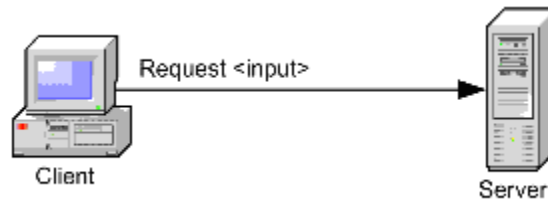
- A customer (the client) sends a request for an account balance to the Account Record Storage System (the server).
- The Account Record Storage System (the server) sends a reply to the customer (the client), specifying the dollar amount in the designated account.
- An *asynchronous* operation is one 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 9-6**

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 9-8**

After defining the Invoker event, you need to create the following two subscriptions:

- Create a Local subscription with 'Invoke Web Service' Action Type, page 9-8

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 9-15

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 9-17**

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 9-17.

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 9-8

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 9-15

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 9-48.

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 9-21.

- 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 9-26.

- 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 9-28.

### **Creating a Local Event Subscription with 'Invoke Web Service' Action Type**

## Create an Event Subscription

**ORACLE** Administrator Workflow

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 'Select Service' step in the Oracle Administrator Workflow. 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 the sequence: Load WSDL (completed), Select Service (current), Select Service Port, Select Operation, and Subscription Documentation. The main content area prompts the user to 'Select a service to consume in Business Event Subscription'. It includes a note: '\* Indicates required field'. The form fields are: WSDL Source (URL: http://rws60062rems.us.oracle.com:8077/OA\_HTML/XMLGatewayWSDL), WSDL Description (XMLGateway), and Service Name (a dropdown menu with 'XMLGateway' selected). Navigation buttons at the bottom right are 'Cancel', 'Back', 'Step 2 of 5', and 'Next'. The footer contains links for 'About this Page' and 'Privacy Statement', and a copyright notice: '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 'Select Service Port' step in the Oracle Administrator Workflow. 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 the sequence: Load WSDL (completed), Select Service (completed), Select Service Port (current), Select Operation, and Subscription Documentation. The main content area prompts the user to 'Select a service port to consume in Business Event Subscription'. It includes a note: '\* Indicates required field'. The form fields are: WSDL Source (URL: http://rws60062rems.us.oracle.com:8077/OA\_HTML/XMLGatewayWSDL), WSDL Description (XMLGateway), and Selected Service (XMLGateway). Below these is a table for 'Select Service Port' with two columns: 'Port End Point' and 'Port End Point'. The table contains one entry: 'XMLGatewayPort' with the URL 'http://rws60062rems.us.oracle.com:8077/webservices/AppsWSProvider/oracle/apps/fnd/XMLGateway'. Navigation buttons at the bottom right are 'Cancel', 'Back', 'Step 3 of 5', and 'Next'. The footer contains links for 'About this Page' and 'Privacy Statement', and a copyright notice: '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

ORACLE Administrator Workflow

Home | Developer Studio | Business Events | Status Monitor | Notifications | Administration

Events | Subscriptions | Agents | Systems

Load WSDL | Select Service | Select Service Port | **Select Operation** | Subscription Documentation

Select Service Operation

Select a service to consume in Business Event Subscription

WSDL Source URL: [http://rws60062rems.us.oracle.com:8077/OA\\_HTML/XMLGatewayWSDL](http://rws60062rems.us.oracle.com:8077/OA_HTML/XMLGatewayWSDL)

WSDL Description: XMLGateway

Selected Service: XMLGateway

Selected Port: XMLGatewayPort

Select Operation	Port Type	Operation Type
<input checked="" type="radio"/> ReceiveDocument	XMLGatewayPortType	

Cancel Back Step 4 of 5 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.

- 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

ORACLE Administrator Workflow

Home | Developer Studio | Business Events | Status Monitor | Notifications | Administration

Events | Subscriptions | Agents | Systems

Load WSDL | Select Service | Select Service Port | Select Operation | **Subscription Documentation**

Create Event Subscription - Invoke Web Service

Cancel Back Step 5 of 5 Apply

Action

Java Rule Function:

Subscription Parameters

Select Object:

Select All | Select None

Select Name	Value
<input type="checkbox"/> WFES_SOAP_USERNAME	<input type="text" value="sysadmin"/>
<input type="checkbox"/> WFES_SOAP_PASSWORD_MCD	<input type="text" value="FND"/>
<input type="checkbox"/> WFES_SOAP_PASSWORD_KEY	<input type="text" value="sysadmin"/>

Enter parameters and their values with no spaces

Documentation

\* Owner Name:

\* Owner Tag:

Customization Level:

Description:

Cancel Back Step 5 of 5 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.

- 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 9-36.

#### **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 ATG121U4.US.ORACLE.COM

**Triggering Event**

\* Source Type Error

\* 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 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.
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.



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 9-28

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 >

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 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**

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 `ReceiveDocument` in Oracle E-Business Suite generic XML Gateway service ([http://hostname:port/OA\\_HTML/XMLGatewayWSDL](http://hostname:port/OA_HTML/XMLGatewayWSDL)) as described below.

```

<definitions
targetNamespace="http://xmlns.oracle.com/oracle/apps/fnd/XMLGateway">
...
<message name="ReceiveDocument_Request">
  <part name="header" element="tnsl:XMLGateway_Header"/>
  <part name="body" element="tnsl:ReceiveDocument"/>
</message>
...
<binding name="XMLGatewaySOAPBinding" type="tns:XMLGatewayPortType">
<soap: binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="ReceiveDocument">
    <soap:operation
soapAction="http://hostname:port/webservices/AppsWSProvider/oracle/apps/
fnd/XMLGateway"/>
    <input>
      <soap:header message="tns:ReceiveDocument_Request" part="header"
use="literal"/>
      <soap:body parts="body" use="literal"/>
    </input>
    <output>
      <ssoap:body use="literal"/>
    </output>
  </operation>
</binding>
...
</definitions>

```

The operation `ReceiveDocument` requires input message `ReceiveDocument_Request`, which has two parts:

- **Body:** The value of `ReceiveDocument` type to be set as SOAP body is sent as business event payload.
- **Header:** The value of `XMLGateway_Header` 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 9-22
- [Other Web Service Input Message Parts](#), page 9-25

### 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, a 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:XMLGateway_Header
xmlns:ns1=\"http://xmlns.oracle.com/apps/fnd/XMLGateway\" \" \" +
    \"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:XMLGateway_Header>\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. 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 Applications 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 9-3.

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 9-17.

To better understand how to invoke a Web service, see Example of Invoking a Web Service From a Workflow Process, page 9-33

## 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:XMLGateway_Header
xmlns:ns1=\"http://xmlns.oracle.com/apps/fnd/XMLGateway\" " " +

```



```

"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:XMLGateway_Header>\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();
    }
}
}

```

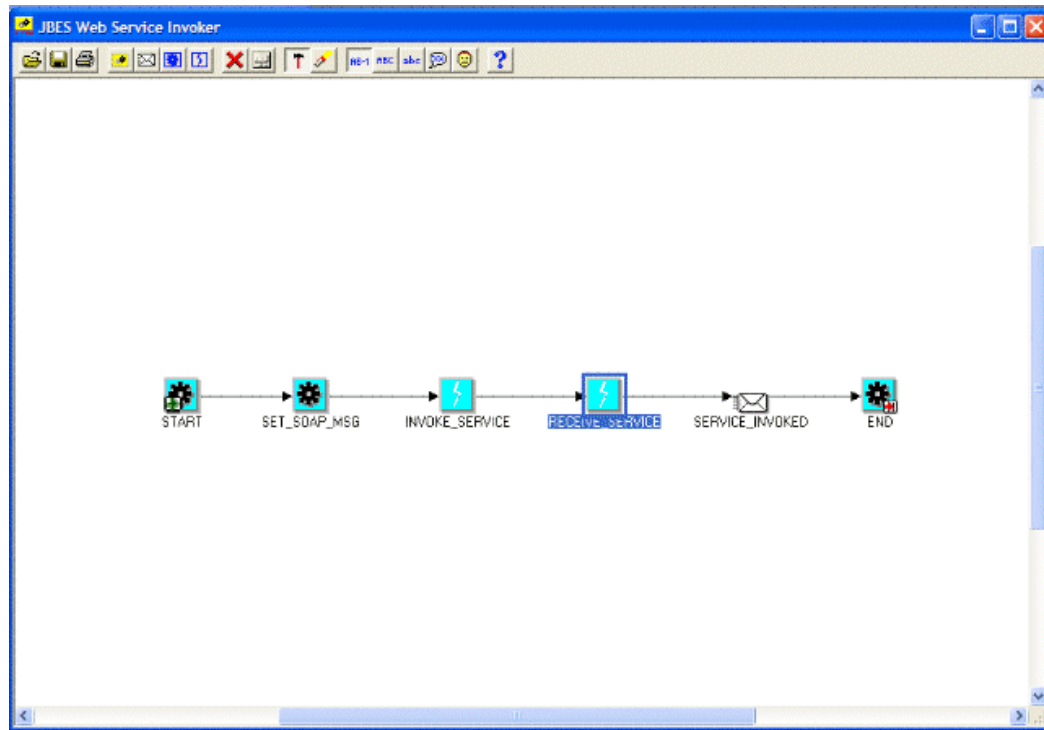
## Example of Invoking a Web Service From a Workflow Process

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 9-6.
2. Local and error event subscriptions to the invoker event. See: Creating Local and Error Event Subscriptions to the Invoker Event, page 9-8.
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 9-17.

### 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 of service components with columns for Name, Status, Type, Startup Mode, Container Type, Container, and Actions. Two components are listed: 'Workflow Inbound JMS Agent Listener' and 'Workflow Inbound Notifications Agent Listener', both with a status of 'Running'. The 'Workflow Inbound JMS Agent Listener' is selected. Below the table, there's a tip: 'TIP GSM = Generic Service Management'. At the bottom, there's a footer with 'Copyright 2001, 2006 Oracle Corporation. All Rights Reserved. About Oracle Applications Manager Version 2.3.1'.

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' text. Navigation links like 'Diagnostics', 'Home', and 'Logout' are visible. 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''. A 'Return to Worklist' link is at the bottom left. The footer contains 'About this Page', 'Privacy Statement', and 'Copyright (c) 2008 Oracle Corporation'.

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 header includes the Oracle logo and 'Administrator Workflow' title. Navigation links for 'Diagnostics', 'Home', 'Logout', 'Preferences', and 'Help' are visible. The breadcrumb trail indicates the path: 'Business Events: Subscriptions > Create Event Subscription >'. The main title is 'Create Event Subscription - Launch Workflow'. There are 'Cancel', 'Back', and 'Apply' buttons at the top right. A message states: 'An event subscription can be routed to a Workflow process. Please specify the Workflow Type and Workflow Process to be launched. \* Indicates required field'. The 'Action' section contains: '\* Workflow Type' (WFERROR), '\* Workflow Process' (DEFAULT\_EVENT\_ERROR2), '\* Priority' (Normal), and 'Additional Options'. The 'Subscription Parameters' section has a 'Select Object' dropdown (Delete), 'Select All' and 'Select None' links, a table with 'Select Name' and 'Value' columns, and an 'Add Another Row' button. The 'Documentation' section includes: '\* Owner Name' (Oracle Workflow), '\* Owner Tag' (FND), 'Customization Level' (User), and a 'Description' text area. 'Cancel', 'Back', and 'Apply' buttons are at the bottom right.

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 9-15.

## 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 9-38
- Using Command Line to Raise an Event, page 9-42

## Using the Test Business Event Page

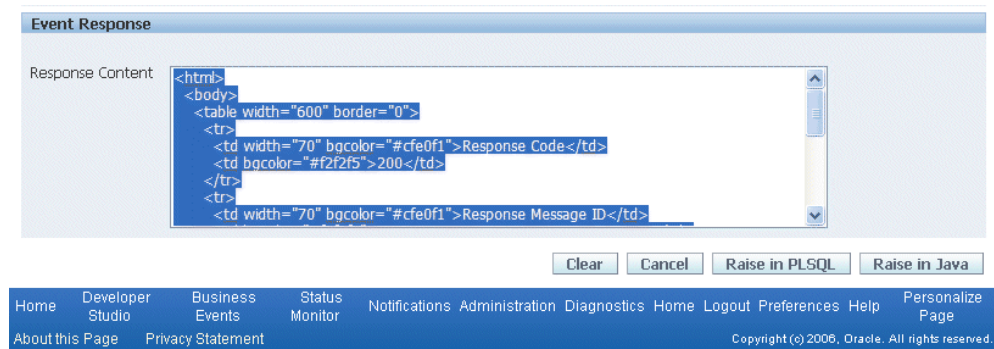
Use the Test Business Event page to test a event by raising it from both PL/SQL API and Java method.

- 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 Application 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 9-21
- Supporting WS-Security, page 9-26
- Calling Back to Oracle E-Business Suite With Web Service Response, page 9-28

#### **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 9-43.

#### **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/apps/fnd/XMLGateway">
  <PO_DOCUMENT>
    <PO_NUM>12345</PO_NUM>
    <PO_TYPE>standards</PO_TYPE>
  </DESCRIPTION>
</PO_DOCUMENT>
</ReceiveDocument>
```

**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: xmigkey2

Send Date:

Send Date must be in the Format: DD-MON-RRRR

**Event Parameters**

Select Object:

Select All | Select None

Select Label	Value
<input type="checkbox"/> WF8ES_INPUT_header	<ns1:XMLGateway_Header xmlns:ns1="http://xmlns.oracle.com/ap
<input type="checkbox"/> WF8ES_OUT_XSL_FILENAME	XMLGResponse.xsl:FND:1.0

**Event Data**

Upload Option:

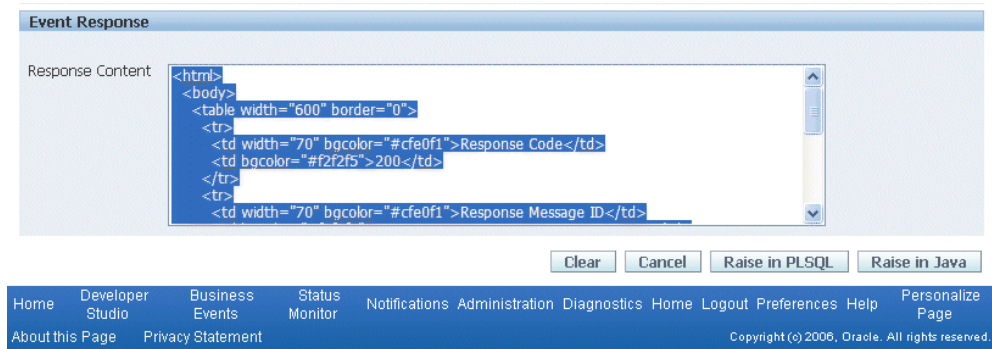
XML Content: 

```
<ReceiveDocument xmlns="http://xmlns.oracle.com/apps/fnd/XMLGateway">
<PO_DOCUMENT>
<PO_NUM>12345</PO_NUM>
<PO_TYPE>Standard</PO_TYPE>
<DESCRIPTION/>
</PO_DOCUMENT>
</ReceiveDocument>
```

If XML content is greater than 4000 characters, Select "Upload XML" option.

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 Application 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 9-43
- Troubleshooting invocation failure on Concurrent Manager (CM) Tier JVM, page 9-47
- Troubleshooting invocation failure on Standalone JVM, page 9-47

## 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 9-44
2. Displaying On-screen Logging, page 9-44

### 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'. It contains a table with the following columns: Profile Option Name, Site, Application, Responsibility, and User. The first row has 'FND: Diagnostics' in the 'Profile Option Name' column and 'Yes' in the 'Site' column. The other columns are empty. There are several empty rows below the first one. At the bottom of the window, there is a scroll bar.

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 9-44.

**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:

Show Log on Screen ▾

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 9-43.

9-46 Oracle E-Business Suite Integrated SOA Gateway Developer's Guide

## 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 9-36.

### 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 9-30.

## 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:XMLGateway_Header
xmlns:ns1=\"http://xmlns.oracle.com/apps/fnd/XMLGateway\" " +
"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:XMLGateway_Header>\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*.

---

# 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.

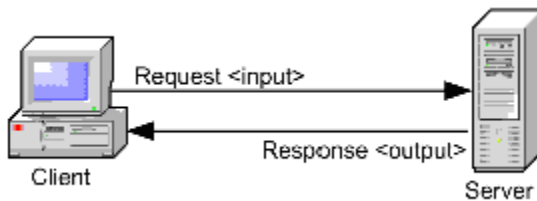
Based on the message patterns, appropriate activities representing various actions can be orchestrated in a meaningful way and enable the services.

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***



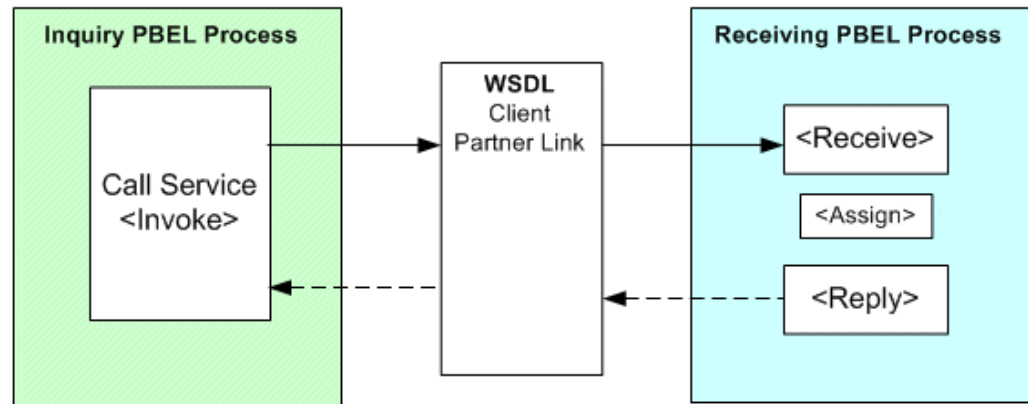
Many online banking tasks are programmed in request-response mode. For example, a request for an account balance is executed as follows:

- 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**

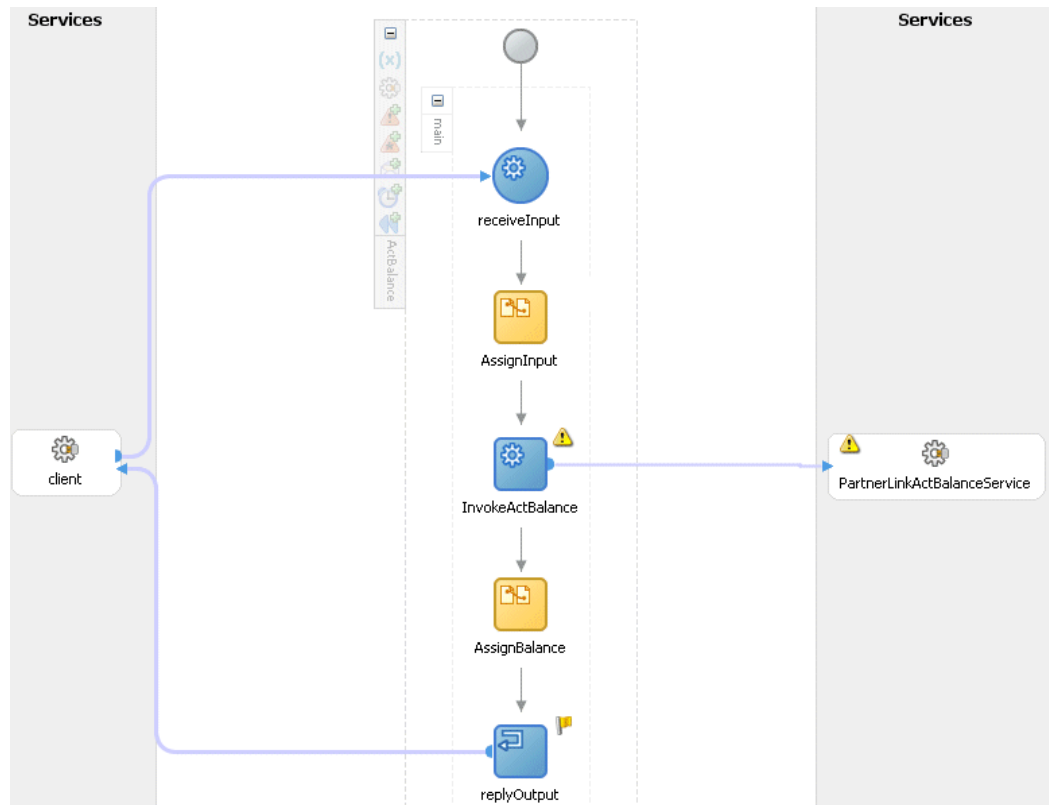
A Reply activity enables the business process to send a response message in reply to a message that was received through a Receive activity. For example, the Reply activity takes the account balance from the server as an output variable and sends it back to the requestor.

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

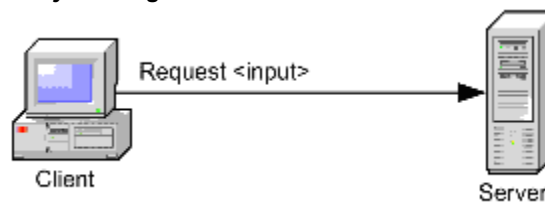
Once a Partner Link is successfully added to a synchronous BPEL project, a WSDL description URL that corresponds to the ActBalanceService business event service with appropriate event payload will be automatically generated. This ActBalanceService Partner Link serves as a bridge to communicate information 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.

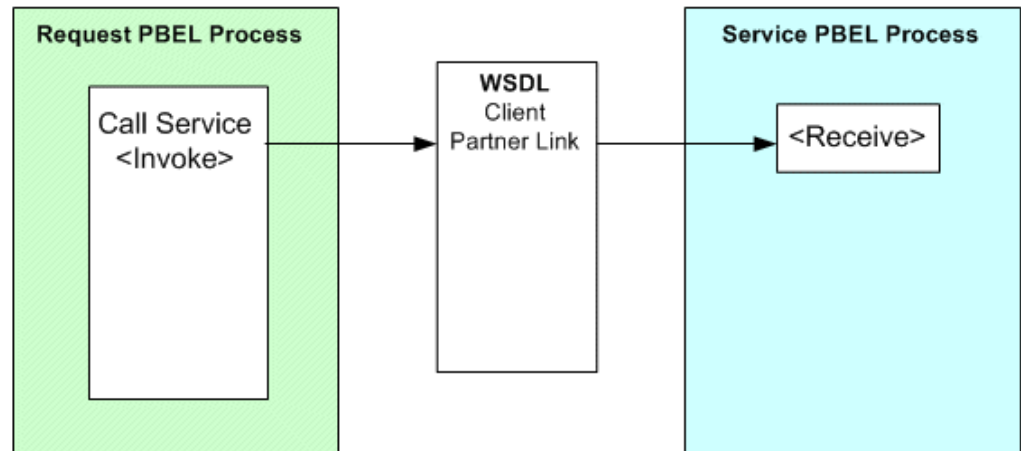
#### *Request Only Message Pattern*



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

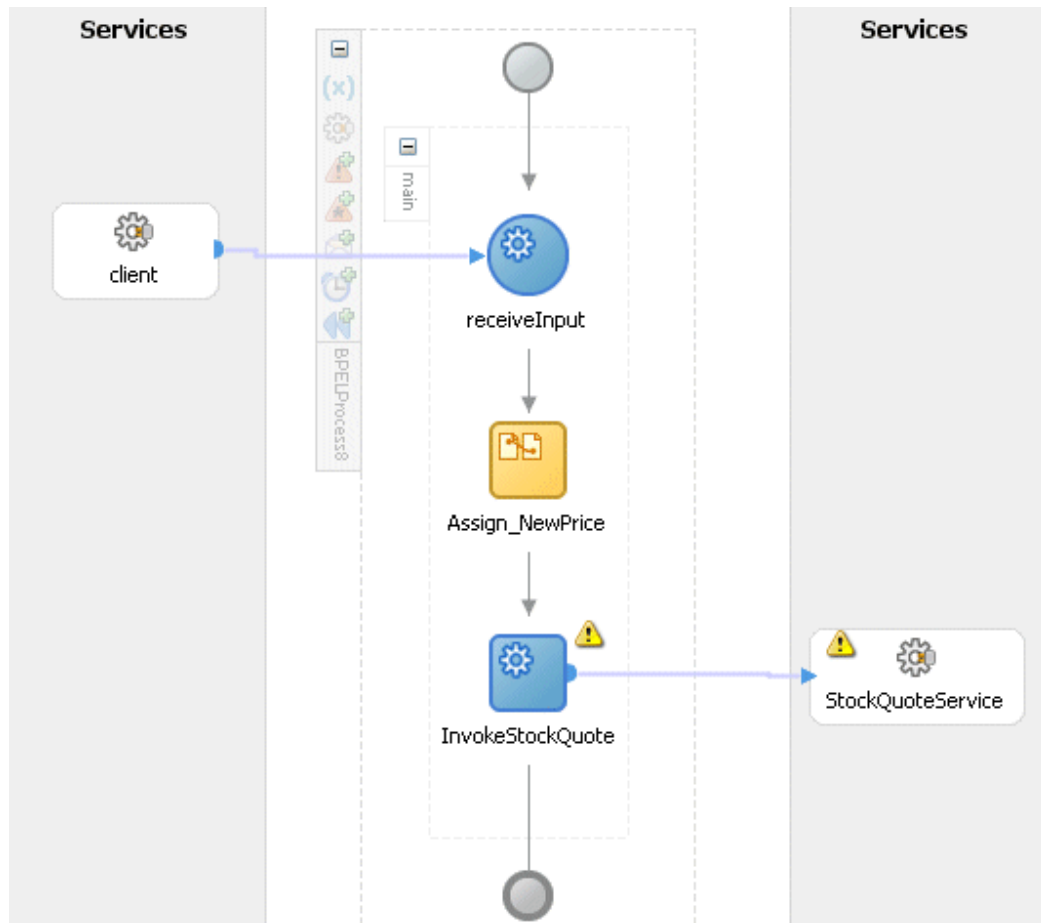


The following two BPEL processes are involved in this type of service:

- **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 Applications 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 Service Objects Design Tasks

- Adding an Assign Activity, 7-17
- Adding an Invoke Activity, 7-14
- Adding a Partner Link for File Adapter, 7-9
- Creating a New BPEL Project, 7-4
- Creating a Partner Link, 7-6

## C

---

### Composite Services

- download, 8-2
- modify, 8-4
- overview, 8-1
- view, 8-2

### create BPEL

- message pattern, A-1
- One-Way, A-6
- Synchronous Request-Response, A-2

### Creating BPEL Using Business Events

- Assign, 5-22
- Create a New BPEL Project, 5-4
- Create a Partner Link AQ Adapter, 5-5
- Create a Partner Link File Adapter, 5-14
- invoke, 5-20
- receive, 5-12

### Creating Invoker Event Subscription

- Creating Error Subscription, 9-15
- Creating Subscription with 'Invoke Web Service', 9-8

## D

---

### deploy and test bpel

- deploy bpel, 7-19
- test bpel, 7-20

### Deploy and Test Concurrent Program

- deploy bpel, 6-23, 6-24

### Deploy and Test Event BPEL

- deploy bpel, 5-25
- test bpel, 5-26

### Deploy and Test PL/SQL BPEL

- deploy bpel, 3-26
- test bpel, 3-27

### Discovering and Viewing Integration Interfaces

- overview, 2-1
- review details, 2-4
- review WSDL details, 2-6
- search and view interfaces, 2-1

## E

---

### Extensibility

- addWSSecurityHeader, 9-49
- invokeService, 9-49
- postInvokeService, 9-48
- preInvokeService, 9-48
- setInputParts, 9-50

## I

---

### Invoke Web service

- example, 9-33

### Invoke Web Services

- Calling Back to Oracle E-Business Suite With Web Service Responses, 9-28

### Invoke Web services through Oracle Workflow

- overview, 9-1
- Web Service Invocation Using SIF, 9-2
- Invoking Web service steps
  - Creating a receive Event, 9-17
  - Creating Invoke and Receive Events, 9-6
  - creating invoker local and error event subscriptions, 9-8

## O

---

- Oracle E-Business Suite Integrated SOA Gateway
  - component features, 1-2
  - Major Features, 1-2
  - Overview, 1-1

## S

---

- Synchronous Request-Response
  - Orchestrate Synchronous BPEL, A-4

## T

---

- Testing Service Invocation
  - Command Lines, 9-42
  - Test Business Event Page, 9-38
  - Troubleshooting Web Service Invocation Failure
    - Concurrent Manager (CM) Tier JVM, 9-47
    - OACORE OC4J, 9-43
    - Standalone JVM, 9-47

## U

---

- Using Business Events
  - deploy and test bpel, 5-24
  - overview, 5-1
  - using Business Events, 5-1
- Using Business Service Objects
  - deploy and test bpel, 7-19
  - 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, 6-9

- Assign activities, 6-18
- Creating a New BPEL Project, 6-4
- Creating a Partner Link, 6-6
- Invoke activities, 6-14
- Using PL/SQL
  - deploy and test bpel, 3-25
  - overview, 3-1
  - using PL/SQL WSDL, 3-1
- Using PL/SQL WSDL
  - Add an Assign activity, 3-18
  - Add an Invoke activity, 3-15
  - Adding a Partner Link for File Adapter, 3-10
  - Create a New BPEL Project, 3-5
  - Create a Partner Link, 3-7
- Using XML Gateway
  - deploy bpel, 4-57
  - overview, 4-1
  - test bpel, 4-58
  - 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, 4-20
  - Creating a New BPEL Project, 4-6
  - Creating a Partner Link, 4-8, 4-10
  - Invoke, 4-18
- Using XML Gateway Inbound SOA Provider
  - Run-time tasks, 4-30
- Using XML Gateway Inbound SOA Provider Run-Time Tasks
  - deploy, 4-30, 4-32
- Using XML Gateway outbound
  - using XML Gateway outbound, 4-35
- Using XML Gateway Outbound
  - deploy and test bpel, 4-56
- Using XML Gateway outbound design task
  - Add an Assign Activity, 4-54
  - Add an Invoke Activity, 4-52
  - Add a Partner Link for File Adapter, 4-46
  - Adding a Receive Activity, 4-44
  - create a new BPEL project, 4-37
  - create a Partner Link for AQ Adapter, 4-38
  - overview, 4-36

## W

---

- Web service invocation

- consideration, 9-54
- Extending Seeded Java Rule Function, 9-48
- Testing Web Service Invocation, 9-37
- Troubleshooting Web Service Invocation Failure, 9-43
- Web Service Invocation Using SIF
  - invoking Web services, 9-30
  - message patterns, 9-3
  - metadata definition, 9-5
  - Supporting WS-Security, 9-26
  - Web Service Input Message Parts, 9-21

