Oracle® Application Server

Adapter for J.D. Edwards OneWorld User's Guide

10*g* Release 3 (10.1.3.1.0)

B28996-01

March 2007

ORACLE®

Oracle Application Server Adapter for J.D. Edwards OneWorld User's Guide, 10*g* Release 3 (10.1.3.1.0)

B28996-01

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

Primary Author: Stefan Kostial

Contributing Authors: Sheela Vasudevan, Sunil Gopal, Marian Jones, Vikas Anand, Sunil Wadhwa, Vishal Saxena

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

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

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

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

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

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

Contents

Preface

This Preface contains these topics:

Audience

Oracle Application Server Adapter for J.D. Edwards OneWorld User's Guide is intended for those who perform the following tasks:

- Install applications
- Maintain applications

To use this document, you need to know how to install and configure Oracle BPEL Process Manager.

Documentation Accessibility

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

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

TTY Access to Oracle Support Services

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

Related Documents

For more information, refer to these Oracle resources:

- *Oracle Application Server Adapter Concepts*
- *Oracle Application Server Adapters Installation Guide*

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/OA_HTML/ibeCCtdMinisites.jsp?language=US`

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://www.oracle.com/technology/membership/index.html`

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://www.oracle.com/technology/documentation/index.html`

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples
- Conventions for Windows Operating Systems

Conventions in Text

We use the following conventions in text to help you more quickly identify special terms. The table also provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.

Convention	Meaning	Example
`UPPERCASE monospace (fixed-width) font`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, Recovery Manager keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, user names, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.
`lowercase monospace (fixed-width) font`	Lowercase monospace typeface indicates executable programs, filenames, directory names, and sample user-supplied elements. *Note:* Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to start SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run *`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
`[ ]`	Anything enclosed in brackets is optional.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
`{ }`	Braces are used for grouping items.	`{ENABLE | DISABLE}`

`|`	A vertical bar represents a choice of two options.	`{ENABLE | DISABLE}`
		`[COMPRESS | NOCOMPRESS]`

`...`	Ellipsis points mean repetition in syntax descriptions.
	In addition, ellipsis points can mean an omission in code examples or text.
		`CREATE TABLE ... AS `*`subquery`*`;`
		`SELECT `*`col1`*`, `*`col2`*`, ... , `*`coln`*` FROM employees;`

Other symbols	You must use symbols other than brackets ([ ]), braces ({ }), vertical bars (|), and ellipsis points (...) exactly as shown.	`acctbal NUMBER(11,2);`
		`acct    CONSTANT NUMBER(4) := 3;`

`Italics`	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/`*`system_password`*
		`DB_NAME = `*`database_name`*

Convention	Meaning	Example

`UPPERCASE`	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. Because these terms are not case sensitive, you can use them in either UPPERCASE or lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

`lowercase`	Lowercase typeface indicates user-defined programmatic elements, such as names of tables, columns, or files.
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Conventions for Windows Operating Systems

The following table describes conventions for Windows operating systems and provides examples of their use.

Convention	Meaning	Example
Click **Start**, and then choose the ***menu item***	How to start a program.	To start the Database Configuration Assistant, click **Start**, and choose **Programs**. In the Programs menu, choose **Oracle** - ***HOME_NAME*** and then click **Configuration and Migration Tools**. Choose **Database Configuration Assistant**.
File and directory names	File and directory names are not case sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (|), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotes. If the filename begins with \\, then Windows assumes it uses the Universal Naming Convention.	c:\winnt"\"system32 is the same as C:\WINNT\SYSTEM32
`C:\>`	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the *command prompt* in this manual.	`C:\oracle\oradata>`
Special characters	The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information on escape and special characters.	`C:\>exp HR/HR TABLES=employees QUERY=\"WHERE job_id='SA_REP' and salary<8000\"`

Convention	Meaning	Example
`HOME_NAME`	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	`C:\> net start Oracle`*`HOME_NAME`*`TNSListener`
`ORACLE_HOME` and *`ORACLE_BASE`*	In releases prior to Oracle8i release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *`ORACLE_HOME`* directory. This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level *`ORACLE_HOME`* directory. There is a top level directory called *`ORACLE_BASE`* that by default is `C:\oracle\product\10.1.0`. If you install the latest Oracle release on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is `C:\oracle\product\10.1.0\db_`*`n`*, where *`n`* is the latest Oracle home number. The Oracle home directory is located directly under *`ORACLE_BASE`*. All directory path examples in this guide follow OFA conventions. Refer to *Oracle Database Installation Guide for Windows* for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.	Change to the *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\rdbms\admin` directory.

Help Us to Serve You Better

To help our consultants answer your questions effectively, please be prepared to provide specifications and sample files and to answer questions about errors and problems.

The following list includes the specifications our consultants require.

- **Platform**:
- **Operating System**:
- **Operating System Version**:
- **Product List**:
- **Adapters**:
- **Adapter Deployment**:

  For example, *J2CA* or *Business Services Engine (BSE)*
- **Container Version**:

The following table lists components. Specify the version in the column provided.

Component	Version
Adapter	
EIS (DBMS/APP)	
HOTFIX/Service Pack	

In the following table, specify the JVM version and vendor.

JVM Version	Vendor

The following table lists additional questions to help us serve you better.

Request/Question	Error/Problem Details or Information
Provide usage scenarios or summarize the application that produces the problem.	
Has this happened previously?	
Can you reproduce this problem consistently?	
Any **change in the application environment**: software configuration, EIS/database configuration, application, and so on?	
Under what circumstance does the problem *not* occur?	
Describe the **steps** to reproduce the problem.	
Describe the **problem**.	
Specify the **error** message(s).	

The following is a list of error or problem files that might be applicable.

- XML schema
- XML instances
- Other input documents (transformation)
- Error screen shots
- Error output files
- Trace and log files
- Log transaction

1 Introduction

OracleAS Adapter for J.D. Edwards OneWorld provides connectivity and carries out interactions on a J.D. Edwards OneWorld system. This chapter provides information about OracleAS Adapter for J.D. Edwards OneWorld to help you accomplish your integration projects.

This chapter discusses the following topics:

Adapter Features

OracleAS Adapter for J.D. Edwards OneWorld provides a means to exchange real-time business data between J.D. Edwards OneWorld systems and other applications, databases, or external business partner systems. The **adapter** enables inbound and outbound processing with J.D. Edwards OneWorld.

OracleAS Adapter for J.D. Edwards OneWorld can be deployed as a J2EE Connector Architecture (J2CA) 1.0 resource adapter. This deployment is referred to as OracleAS J2CA adapter. It can also be deployed as a Web services servlet and is referred to as OracleAS Adapter Business Services Engine (BSE).

OracleAS Adapter for J.D. Edwards OneWorld uses XML messages to enable non-J.D. Edwards applications to communicate and exchange transactions with J.D. Edwards using services and events. Services and events are described as follows:

- Services: Enables applications to initiate a J.D. Edwards business event.
- Events: Enables applications to access J.D. Edwards data only when a J.D. Edwards business event occurs.

To support event functionality, the following two features are implemented:

- Port: A **port** associates a particular business object exposed by an adapter with a particular disposition. A disposition defines the protocol and location of the event data. The port defines the end point of the event consumption.

  The port is an Oracle adapter component that pushes the event received from the Enterprise Information System (EIS) to the adapter client.

Note: You are not required to create or configure ports for use with BPEL Process Manager. However, in this release you can associate an event schema to a port under a J2CA configuration.

The port validation feature is currently not available.

- Channel: A **channel** represents configured connections to particular instances of back-end or other types of systems. A channel binds one or more event ports to a particular **listener** managed by the adapter.

  The channel is the adapter component that receives events in real time from the Enterprise Information System (EIS) application. The channel component can be a File reader, an HTTP listener, a TCP/IP listener, or an FTP listener. A channel is always EIS specific. The adapter supports multiple channels for a particular EIS. This enables the user to choose the optimal channel component based on deployment requirements.

OracleAS Adapter for J.D. Edwards OneWorld provides:

- XML schemas for the J2CA 1.0 resource adapter.
- Web services for BSE.

See Also: *Oracle Application Server Adapter Concepts*

J.D. Edwards OneWorld Concepts

You can use OracleAS Adapter for J.D. Edwards OneWorld to call a J.D. Edwards OneWorld Master Business Function, such as Address Book, Purchase Order, and Sales Order. You can also use the adapter as a part of an integration effort to connect OneWorld with non-OneWorld systems.

OracleAS Adapter for J.D. Edwards OneWorld can receive an XML document, or it can run one or more J.D. Edwards Master Business Functions (MBF) by passing an XML document into OneWorld through the J.D. Edwards OneWorld ThinNet API.

Integration with J.D. Edwards OneWorld

J.D. Edwards OneWorld supports multiple methods and technologies to provide interoperability. The three supported entry points are:

- Flat files
- Database tables
- Master Business Function (MBF) interactive calls

You configure Oracle AS Adapter to send requests to J.D. Edwards OneWorld. The agent processes requests for J.D. Edwards OneWorld Master Business Functions (MBF), embedded in XML documents, and forwards them to a back-end J.D. Edwards OneWorld system. The resulting response information is then returned and processed for further routing.

OracleAS Adapter for J.D. Edwards OneWorld can receive an XML request document from a client and call a specific function in the target Enterprise Information System (EIS). OracleAS Adapter for J.D. Edwards OneWorld acts as a consumer of request messages and provides a response. An agent performs the following functions:

- Receives requests from a legacy system, another EIS, or a non-EIS client.
- Transforms the XML request document into the EIS-specific format.

  The request document conforms to a request XML schema.
  The schema is based on metadata in the EIS.
- Calls the underlying function in the EIS and waits for its response.
- Transforms the response from the EIS-specific data format to an XML document.

  The response document conforms to a response XML schema for the agent.
  The schema is based on metadata in the EIS.

You can configure a listener, known as a channel, for the adapter to receive messages from J.D. Edwards OneWorld. The information the listener receives is used to build an XML record and is forwarded to any specified disposition for further processing.

Listeners are consumers of EIS-specific messages and may or may not provide a response. A listener performs the following functions:

- Receives messages from an EIS client
- Transforms the EIS-specific message format into an XML format.

  The XML format conforms to an XML schema.
  The schema is based on metadata in the EIS.

Propagating External Listeners into J.D. Edwards OneWorld

When integrating external listeners into OneWorld using flat file input, the files are imported through a batch program and placed on an unedited transaction table. The records on the transaction table are processed by a batch program that makes calls to the appropriate MBF.

The database table method bypasses the first step in the flat file method, and records are written directly to the unedited transaction table. The records on the transaction table are processed by a batch program that makes calls to the appropriate MBF.

The third method, calling the MBF directly, bypasses the batch processing completely and provides synchronous access to OneWorld.

Propagating Internal Listeners out of J.D. Edwards OneWorld

Integrating a J.D. Edwards OneWorld listener with external systems is similar to the inbound process, except in reverse. The Data Export Control table maintains the determination of whether a transaction must be integrated with an external system. When a transaction must be integrated, the MBF handles logging of all additions, changes, and deletions to the unedited transaction table. After the transaction information is written to the table, a key for that record is sent from the MBF to the subsystem data queue.

The subsystem data queue triggers the processing of the new record by launching an outbound subsystem batch process that is generic and handles all outbound transactions. The outbound subsystem then accesses the Data Export Control table to determine the configured external subscriber to run.

J.D. Edwards OneWorld Interoperability Framework

J.D. Edwards OneWorld enables integration with systems through its interoperability framework. The adapter uses the OneWorld framework and leverages various integration access methods to provide the greatest amount of flexibility and functionality.

OracleAS Adapter for J.D. Edwards OneWorld supports the following integration access methods:

- J.D. Edwards OneWorld ThinNet API
- J.D. Edwards OneWorld XML
- J.D. Edwards unedited transaction tables (Z tables)

Figure 1–1 illustrates the inbound processing framework.

The agent uses the J.D. Edwards OneWorld ThinNet API to communicate with the OneWorld application. Using the ThinNet API, the agent can run one or more MBF in a single Unit Of Work (UOW). When any of the MBF fail, the entire UOW fails, preventing partial updates. Validation of data, business rules, and communications to the underlying database are handled by the OneWorld application because the agent runs the MBF.

Figure 1–1 J.D. Edwards OneWorld Inbound Processing

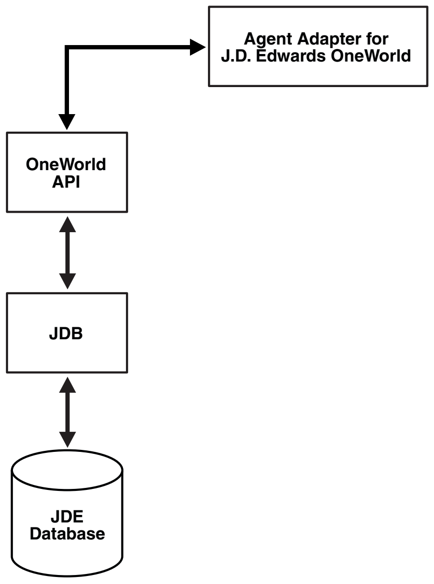
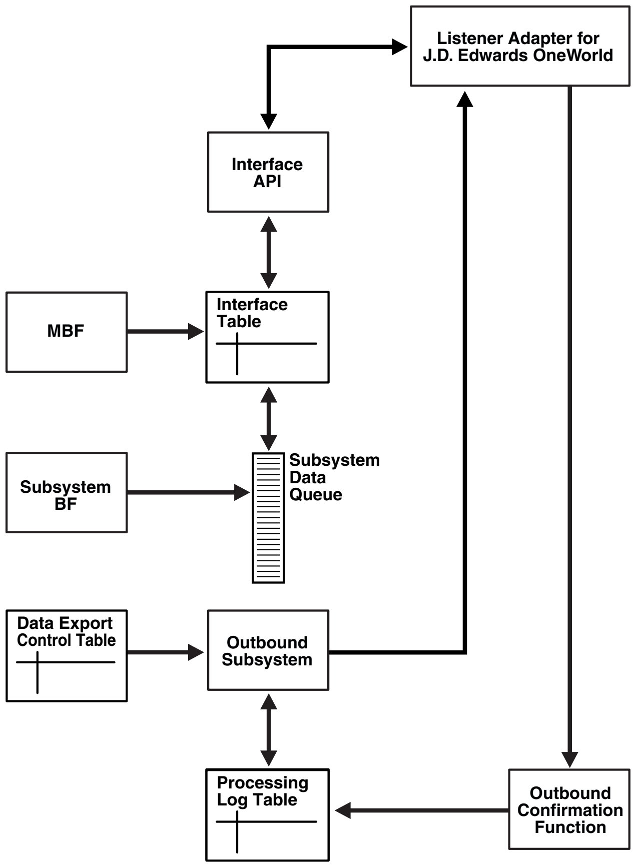


Figure 1–2 illustrates the outbound processing framework.

Figure 1–2 J.D. Edwards OneWorld Outbound Processing



In the outbound process, the event starts when a specific MBF is executed in the J.D. Edwards OneWorld environment. The MBF writes the required information for the event into the appropriate interface table and then notifies the subsystem Batch Function (BF) that an event occurred. The subsystem BF then places an entry about the event on the Subsystem Data Queue.

The outbound subsystem retrieves the data queue entry and looks in the Data Export Control table for the external processes to notify. The outbound subsystem then calls the Oracle Application Server Adapter for J.D. Edwards OneWorld listener with notification. The listener passes the notification to the generator. The generator then uses the J.D. Edwards OneWorld ThinNet API to retrieve the appropriate information from the interface table.

Adapter Architecture

OracleAS Adapter for J.D. Edwards OneWorld works with Application Explorer in conjunction with one of the following components:

- Oracle Application Server Adapter Business Services Engine (BSE)
- Enterprise Connector for J2EE Connector Architecture (J2CA)

OracleAS Adapter Application Explorer (Application Explorer)

Application Explorer is used to configure database connections and create Web services and events. It can be configured to work in a Web services environment in conjunction with BSE or with the Enterprise Connector for J2EE Connector Architecture (J2CA). When working in a J2CA environment, the connector uses the Common Client Interface (CCI) to provide fast integration services using Adapters instead of using Web services.

Both BSE and the connector for J2CA are deployed to an application server with Application Explorer and the adapters.

Application Explorer uses an explorer metaphor for browsing the J.D. Edwards OneWorld system for business functions. Application Explorer enables you to create XML schemas and Web services for the associated business function.

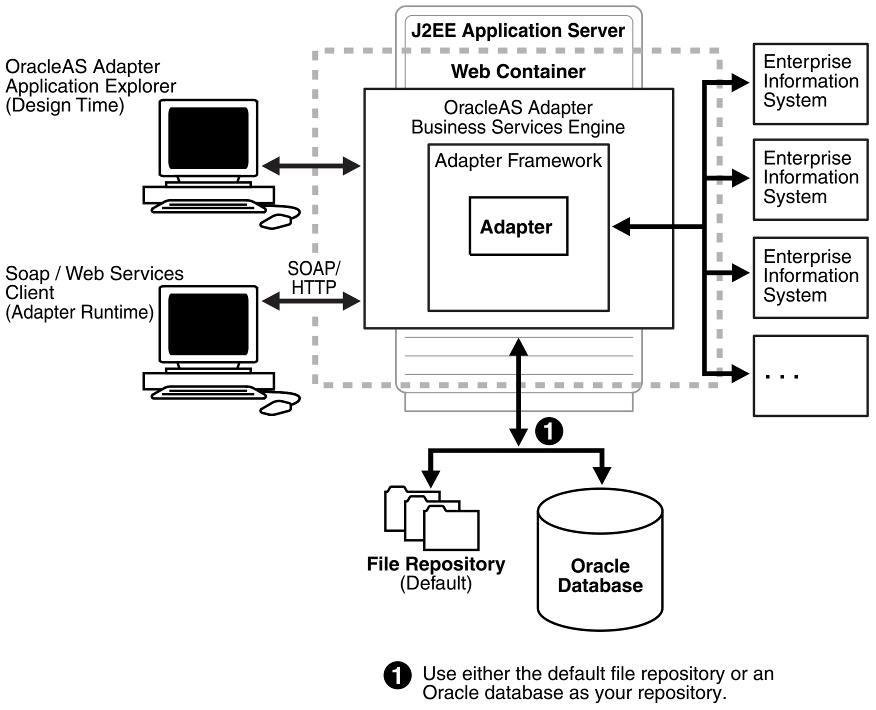
Resource Adapters

OracleAS Adapter for J.D. Edwards OneWorld is a J2CA-based component also known as resource adapter. Resource adapters connect applications that were not originally designed to communicate with each other. Adapters are bidirectional, that is, they can send requests to an Enterprise Information System (EIS), as well as receive notification of events occurring in an EIS.

Oracle Application Server Adapter Business Services Engine (BSE) Architecture

Figure 1–3 shows the generic architecture for the Oracle Web service adapter for packaged applications. The adapter works in conjunction with BSE, as deployed to a Web container in a J2EE application server.

Figure 1–3 OracleAS Adapter Business Services Engine (BSE) Architecture


J2EE Application Server
Web Container
OracleAS Adapter Application Explorer (Design Time)
OracleAS Adapter Business Services Engine
Adapter Framework
Adapter
Soap / Web Services Client (Adapter Runtime)
SOAP/ HTTP
Enterprise Information System
Enterprise Information System
Enterprise Information System
. . .
1
File Repository (Default)
Oracle Database
1 Use either the default file repository or an Oracle database as your repository.

Note: Do not use a file repository for BSE in production environments.

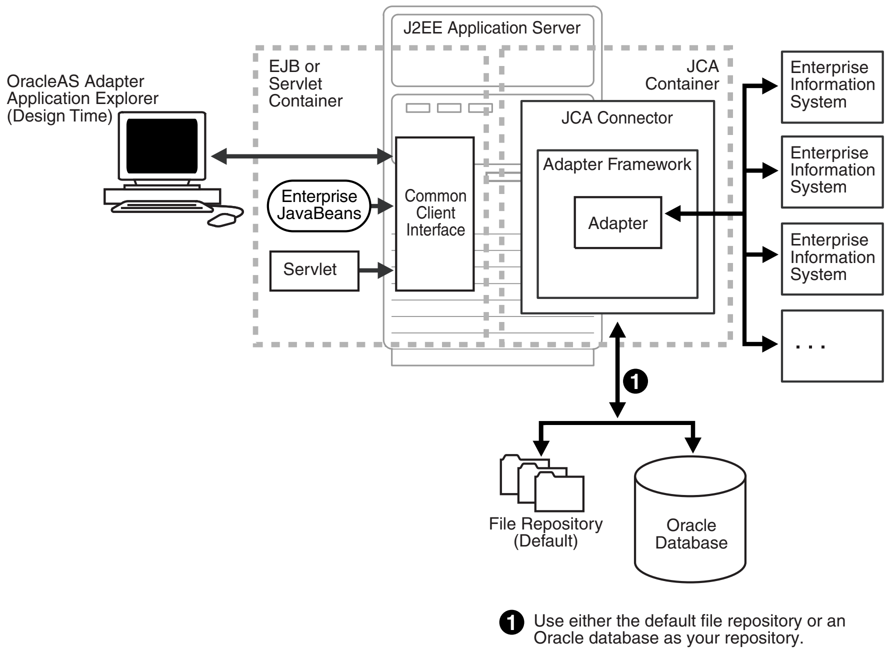
Application Explorer, a design-time tool deployed along with BSE, is used to configure adapter connections, browse EIS objects, configure services, and configure listeners to listen for EIS events. Metadata created while you perform these operations are stored in the repository by BSE.

BSE uses SOAP as a protocol for receiving requests from clients, interacting with the EIS, and sending responses from the EIS back to clients. BSE receives the adapter response, wraps the response XML in a SOAP envelope, and returns it to the adapter bridge. The bridge then strips the SOAP envelope, strips the namespace prefix, if present, and passes the DTD-compliant XML to the IC Adapter agent.

Oracle Application Server Adapter Generic J2CA Architecture

Figure 1–4 shows the generic architecture for Oracle J2CA adapter for packaged applications. The J2CA connector is deployed to a standard J2CA Container and serves as host container to the adapters. The connector is configured with a repository.

Figure 1–4 OracleAS Adapter Generic J2CA Architecture


J2EE Application Server
OracleAS Adapter Application Explorer (Design Time)
EJB or Servlet Container
JCA Container
Enterprise Information System
Enterprise Information System
Enterprise Information System
...
JCA Connector
Adapter Framework
Adapter
Enterprise JavaBeans
Servlet
Common Client Interface
1
File Repository (Default)
Oracle Database
1 Use either the default file repository or an Oracle database as your repository.

Application Explorer, a design tool that works in conjunction with the connector, is used to configure adapter connections, browse EIS objects, configure services, and configure listeners to listen for EIS events. Metadata created during these operations is stored in the repository by the connector. The repository can be a file system or an Oracle database. It is deployed as a RAR file and has an associated deployment descriptor called `ra.xml`. You can create multiple connector factories by editing the OC4J deployment descriptor `oc4j-ra.xml`. See Chapter 3, "OC4J Deployment and Integration" for more information on OC4J deployment.

Processing Business Functions

OracleAS Adapter for J.D. Edwards OneWorld enables the processing of OneWorld business functions through the J.D. Edwards OneWorld ThinNet API. Using the API eliminates the requirement of creating complex and impractical batch processes. In addition, a transport layer, such as IBM MQSeries, File, or HTTP is not required because an **agent** or a listener is defined through a TCP connection.

External applications that access OneWorld through OracleAS Adapter for J.D. Edwards OneWorld use either XML schemas or Web services to pass data between the external application and the adapter. Chapter 2, "Configuring OracleAS Adapter for J.D. Edwards One World" describes how to use Application Explorer to create XML schemas and Web services for the J.D. Edwards Master Business Functions (MBF) used with the adapter.

See Also:

- *Oracle Application Server Adapter Concepts*
- *Oracle Application Server Adapters Installation Guide*

BSE Versus OracleAS Adapter J2CA Deployment

If you are using OracleAS Adapter for J.D. Edwards OneWorld with BPEL Process Manager, please note that:

- Only OracleAS Adapter J2CA deployment supports inbound integration (event notification) with BPEL Process Manager.
- Both OracleAS Adapter J2CA and BSE deployments support outbound integration (request-response service) with BPEL Process Manager.

The following three factors explain the differences between deploying BSE and OracleAS Adapter J2CA. Understanding the factors can help in selecting a deployment option.

1. BSE is the preferred deployment option because it:
   - Can be deployed in a separate instance of Oracle Application Server.
   - Provides better distribution of load.
   - Provides better isolation from any errors from third party libraries.
   - Provides better capability to isolate issues for debugging purposes.
   - Conforms more closely to the Service Oriented Architecture (SOA) model for building applications.
2. OracleAS Adapter J2CA provides slightly better performance.

   OracleAS Adapter J2CA does provide slightly better performance than BSE. However, the difference decreases as the transaction rate increases.
3. OracleAS Adapter J2CA and the BSE option both provide identity propagation at runtime.

   The BSE option provides the capability to pass identity using the SOAP header. For OracleAS Adapter J2CA, user name and password can be passed using the connection specification of the CCI.

2 Configuring OracleAS Adapter for J.D. Edwards One World

This chapter describes how to use OracleAS Adapter Application Explorer (Application Explorer) to define a target to connect to a J.D. Edwards OneWorld system, view system objects, and create XML schemas and Web services. This chapter also explains how to configure an event adapter.

This chapter discusses the following topics:

Starting Application Explorer

To start Application Explorer:

1. Start the server where Application Explorer is deployed.
2. From the Windows **Start** menu, select **Programs**, ***OracleAS_home*** **Adapters**, and then **Application Explorer**.

   On Windows, `iaexplorer.bat` is located under `*OracleAS_home*\adapters\application\tools`, where `OracleAS_home` is the directory where Oracle Application Server is installed.

   On UNIX, load the iwae script, `iwae.sh`, located under `*OracleAS_home*/adapters/application/tools`, where `OracleAS_home` is the directory where Oracle Application Server is installed.

Application Explorer starts. You can now define new targets to your Enterprise Information System (EIS).

Configuring Settings for BSE or J2CA

Before a repository configuration can be created, you must configure OracleAS Adapter Business Services Engine (BSE). You need not configure the Connector for J2CA because the `ra.xml` file is configured automatically during installation.

Configuring BSE

After BSE is deployed to Oracle Application Server, you can configure it through the BSE configuration page.

To configure BSE:

1. Display the following page in your browser:

   http://*hostname*:*port*/ibse

   Where `hostname` is the host name of Oracle Application Server and`port` is the HTTP port for Oracle Application Server.

   For example,

   ```
   http://localhost:7777/ibse
   ```

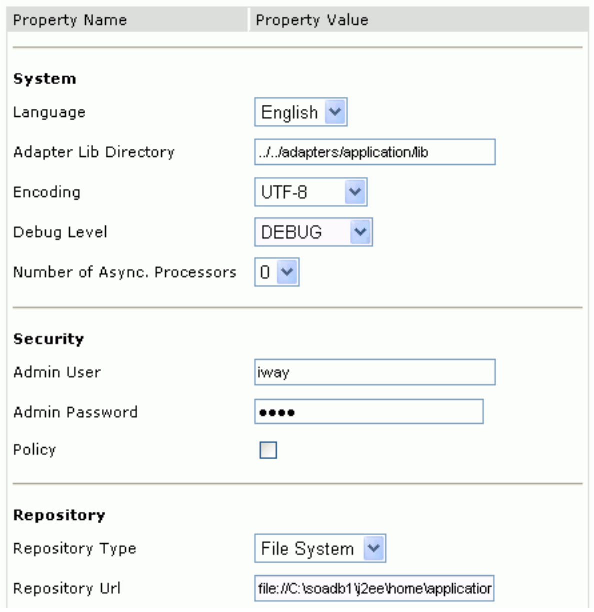
   Note: The first time you access this page, it may take time to load.

2. Log on when prompted.

   When first installed, the user ID and the password are:

   - User name: iway
   - Password: iway

   The BSE configuration page is displayed.



Property Name	Property Value
System	
Language	English
Adapter Lib Directory	../../adapters/application/lib
Encoding	UTF-8
Debug Level	DEBUG
Number of Async. Processors	0
Security	
Admin User	iway
Admin Password	●●●●
Policy	☐
Repository	
Repository Type	File System
Repository Url	file://C:\soadb1\j2ee\home\applicatior

3. Ensure that the Adapter Lib Directory parameter specifies the path to the lib directory, for example:

   OracleAS_home\adapters\application\lib

After you specify the path, adapters in the lib directory are available to BSE.

4. For security purposes, enter a new password in the **Admin Password** field.

> **Note:** The **Repository URL** field specifies where the file system repository is located. To use a database repository, you must enter the repository connection information. For the initial verification, use a file system repository. See "Configuring an Oracle Repository" on page 2-6 for information on switching to a database repository.

5. Click **Save**.

Configuring BSE System Settings

To configure BSE system settings:

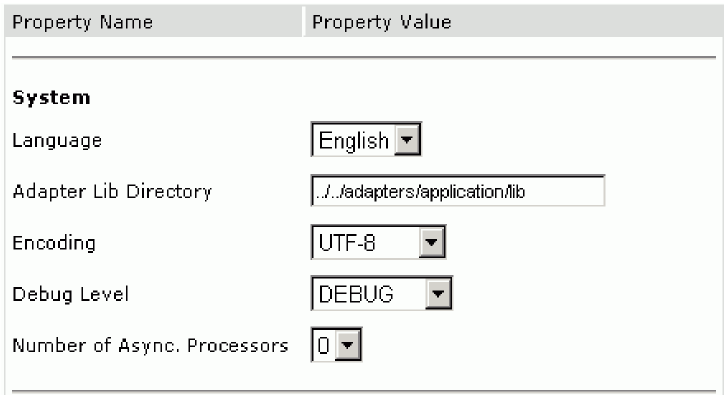
1. Display the BSE configuration page by entering the following URL in a browser:

   ```
   http://hostname:port/ibse/IBSEConfig
   ```

   Where `hostname` is the machine where BSE is installed and `port` is the port number on which BSE is listening.

   > **Note:** The server to which BSE is deployed must be running.

The BSE configuration page is displayed, as shown in the following figure.

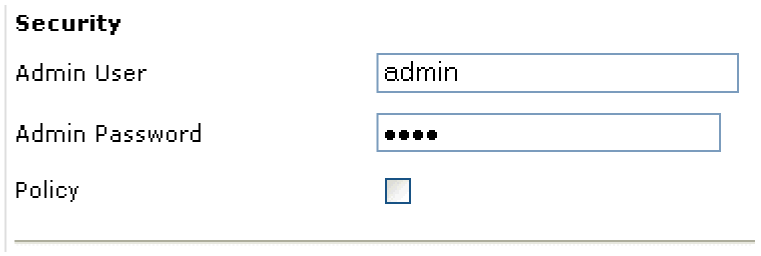


Property Name	Property Value
System	
Language	English
Adapter Lib Directory	../../adapters/application/lib
Encoding	UTF-8
Debug Level	DEBUG
Number of Async. Processors	0

2. Configure the system settings according to the information in the following table.

Parameter	Description
Language	Specify the required language.
Adapter Lib Directory	Enter the full path to the directory where the adapter jar files reside.
Encoding	Only UTF-8 is supported.
Debug Level	Specify the debug level from one of the following options: ■ None ■ Fatal ■ Error ■ Warning ■ Info ■ Debug
Number of Async. Processors	Select the number of asynchronous processors.

The following image shows the Security pane of the BSE configuration page.

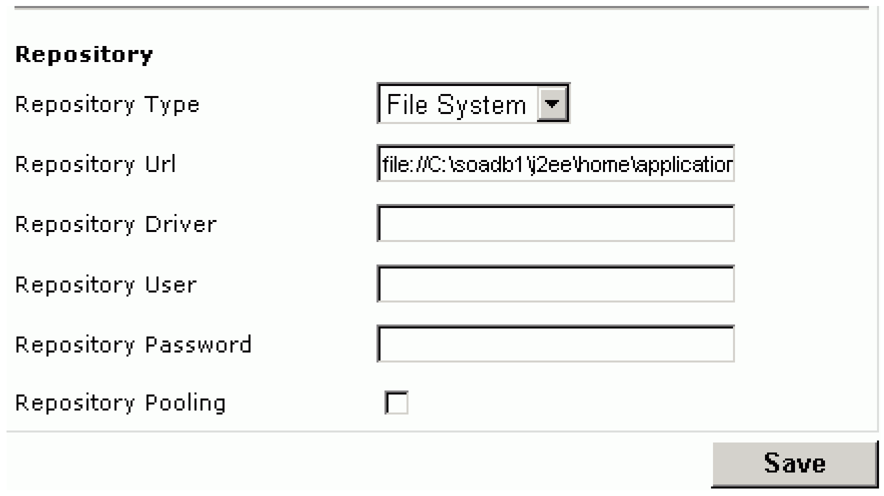

Security

Admin User	admin
Admin Password	●●●●
Policy	☐

3. Configure the security settings according to the information in the following table.

Parameter	Description
Admin User	Provide a BSE administrator ID.
Admin Password	Enter the password associated with the BSE administrator ID.
Policy	Select the check box to enable policy security.

The following image shows the Repository pane of the BSE configuration page.


Repository

Repository Type	File System
Repository Url	file://C:\soadb1\j2ee\home\applicatior
Repository Driver	
Repository User	
Repository Password	
Repository Pooling	☐

BSE requires a repository to store transactions and metadata required for the delivery of Web services. For more information, see "Configuring a File System Repository" on page 2-5 and "Configuring an Oracle Repository" on page 2-6.

4. Configure the repository settings according to the information in the following table.

Parameter	Description
Repository Type	Select one of the following repositories from the list: ■ Oracle. ■ File. (Do not use a file repository for BSE in production environments.)
Repository URL	Enter the URL to use when opening a connection to the database.
Repository Driver	Provide the driver class to use when opening a connection to the database (optional).
Repository User	Enter the user ID to use when opening a connection to the database.
Repository Password	Enter the password associated with the user ID.
Repository Pooling	Select the check box to enable pooling.

5. Click **Save**.

Configuring a File System Repository

If you do not have access to a database for the repository, you can store repository information in an XML file on your local machine. However, a file system repository is less secure and efficient than a database repository. When BSE is first installed, it is automatically configured to use a file system repository.

Note: Do not use a file repository for BSE in production environments.

The default location for the repository on Windows is:

```
OracleAS_home\config\base\ibserepo.xml
```

On other platforms, use the corresponding location.

If you are using a file system repository, you are not required to configure any additional BSE components.

Configuring an Oracle Repository

To configure an Oracle repository:

1. Contact your database administrator to obtain an Oracle user ID and password to create the BSE repository.

   This user ID should have rights to create and modify tables, as well as the ability to create and execute stored procedures.

2. Open a command prompt and navigate to the setup directory. Its default location on Windows is:

   ```
   OracleAS_home\iWay55\etc\setup
   ```

   For other platforms, navigate to the corresponding location.

   This directory contains SQL to create the repository tables in the following file:

   ```
   iwse.ora
   ```

3. Enter the following command:

   ```
   sqlplus userid/password @database @ iwse.ora
   ```

Configuring J2CA

During the J2CA deployment of OracleAS Adapter for J.D. Edwards OneWorld, OC4J generates a deployment descriptor called `oc4j-ra.xml`. This descriptor provides OC4J-specific deployment information for resource adapters. See Chapter 3, "OC4J Deployment and Integration" for more information on J2CA deployment and configuration.

No configuration changes are necessary if you are using the default file based repository with J2CA deployment.

Configuring a Database Repository for J2CA

To configure a database repository for J2CA:

1. Execute the `iwse.ora` SQL statement on the machine where the database is installed.

2. Create the `jcatransport.properties` file and save it in the following directory:

   ```
   OracleAS_HOME\adapters\application\config\jca_sample
   ```

3. Enter values for `iwafjca.repo.url`, `iwafjca.repo.user` and `iwafjca.repo.password` fields in the newly created `jcatransport.properties` file. For example:

   ```
   iwafjca.repo.url=jdbc:oracle:thin:@90.0.0.51:1521:orcl
   iwafjca.repo.user=scott
   iwafjca.repo.password=scott1
   ```

4. Open the `oc4j-ra.xml` file in a text editor.
5. Provide the JDBC connection information as a value for the IWAYRepo_URL property.
6. Provide a valid user name for the IWAYRepo_User property.
7. Provide a valid password for the IWAYRepo_Password property.
8. Save your changes to the `oc4j-ra.xml` file.
9. Alter the JDBC driver path in Application Explorer's lcp. For example:

```
lcp=..\lib\orabpel-adapters.jar;C:\jdev\jdbc\lib\ojdbc14.jar;C:\jdev\jdbc\lib\n
ls_charset12.jar;%lcp%
to
lcp=..\lib\orabpel-adapters.jar;..\..\..\jdbc\lib\ojdbc14.jar;..\..\..\jdbc\lib
\nls_charset12.jar;%lcp%
```

Password Encryption

When creating J2CA configurations, you can also encrypt a password using Application Explorer and use this value in the jcatransport.properties and oc4j-ra.xml files for added security.

Configuring Password Encryption

To encrypt a password:

1. Open Application Explorer.
2. Click **Help** and select **Encryption**.

   The Encryption dialog box opens.
3. Type a password in the Password field and click OK.

   An encrypted version of the password displays in the Encryption field.
4. Copy the password.
5. In the jcatransport.properties file, which is used during design time, replace the existing password with the encrypted value.

   The following is a sample of the jcatransport.properties file where the password is replaced:

```
iwafjca.log.level=DEBUG
iwafjca.repo.url=jdbc:oracle:thin:@172.30.166.100:1521:orcl
iwafjca.repo.user=scott
iwafjca.repo.password=ENCR (3189319731832973218312931643233321232227)
```
6. In the oc4j-ra.xml file, which is used during run time, replace the existing password with the encrypted value for the IWayRepoPassword element.
7. Restart the Oracle Application Server.

Creating a Repository Configuration

Before you use Application Explorer with OracleAS Adapter for J.D. Edwards OneWorld, you must create a repository configuration. You can create two kinds of repository configurations, Web services and J2CA, depending on the container to which the adapter is deployed. During design time, the repository is used to store metadata created when using Application Explorer to configure adapter connections, browse EIS objects, configure services, and configure listeners to listen for EIS events. The information in the repository is also referenced at runtime.

A default J2CA repository is created for the default ManagedConnectinFactory. The name of this configuration is `jca_sample`.

See "Adapter Features" on page 1-1 for more information.

Creating a Configuration for BSE

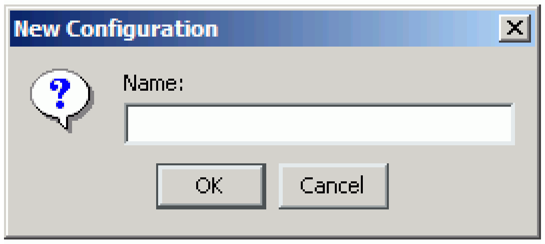
To create a repository configuration for BSE using Application Explorer, you must first define a new configuration.

Defining a New Configuration for BSE

To define a new configuration for BSE:

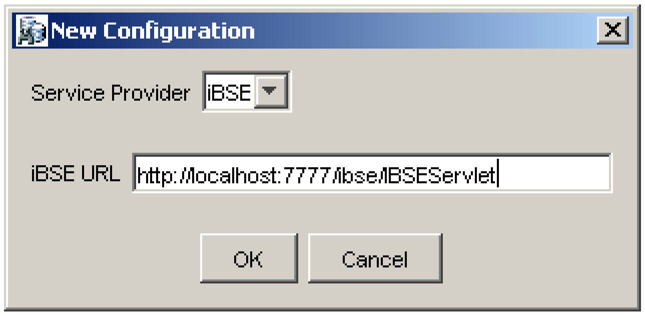
1. Right-click **Configurations** and select **New**.

   The New Configuration dialog box is displayed.



2. Enter a name for the new configuration (for example, myConfig) and click **OK**.

   The New Configuration dialog box is displayed.



Service Provider: iBSE

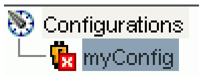
iBSE URL: http://localhost:7777/ibse/IBSEServlet

3. From the Service Provider list, select **iBSE**.
4. In the **iBSE URL** field, accept the default URL or replace it with a different URL using the following format:

   ```
   http://hostname:port/ibse/IBSEServlet
   ```

   Where `hostname` is the machine where your application server resides.
5. Click **OK**.

   A node representing the new configuration appears beneath the root Configurations node.



Creating a Configuration for J2CA

To create a configuration for J2CA using Application Explorer, you must first define a new configuration.

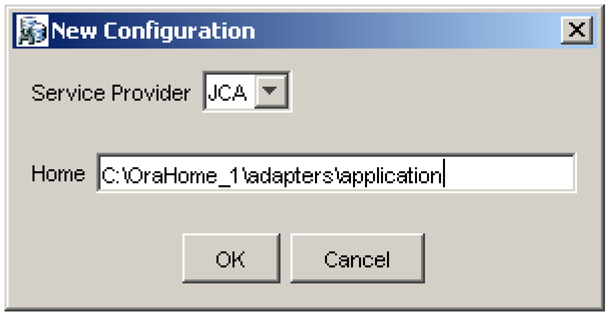
Defining a New Configuration for J2CA

To define a new configuration for J2CA:

1. Right-click **Configurations** and select **New**.

   The New Configuration dialog box is displayed.

2. Enter a name for the new configuration (for example, myConfig) and click **OK**.



New Configuration

Service Provider: JCA

Home: C:\OraHome_1\adapters\application

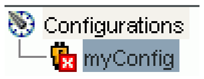
3. From the **Service Provider** list, select **JCA**.

4. In the **Home** field, enter a path to your J2CA configuration directory where the repository, schemas, and other information are stored. This configuration directory is located as follows:

   ```
   OracleAS_home\adapters\application
   ```

5. Click **OK**.

   A node representing the new configuration appears beneath the root Configurations node.



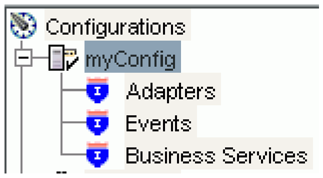
Connecting to a BSE or J2CA Configuration

To connect to a BSE or J2CA configuration:

1. Right-click the configuration to which you want to connect, for example, myConfig.

2. Select **Connect**.

Nodes appear for Adapters, Events, and Business Services (also known as Web services). The Business Services node is only available for BSE configurations. If you are connected to a J2CA configuration, you will not see the Business Services node. The following is an example of a BSE configuration named myConfig:



- Use the **Adapters** folder to create inbound interaction with J.D. Edwards. For example, you use the J.D. Edwards node in the Adapters folder to configure a service that updates J.D. Edwards.
- Use the **Events** folder to configure listeners that listen for events in J.D. Edwards OneWorld.
- Use the **Business Services** folder (available for BSE configurations only) to test Web services created in the Adapters folder. You can also control security settings for the Web services by using the security features of the Business Services folder.

You can now define new targets to J.D. Edwards OneWorld.

Establishing a Connection (Target) for J.D. Edwards OneWorld

Part of the application definition includes adding a target for the adapter. Setting up the target in Application Explorer requires information which is specific to the target.

To browse the available Master Business Functions (MBF), you must first define a target to the system you use. After you define the target, it is automatically saved. You must connect to the system every time you start Application Explorer or after you disconnect.

When you launch Application Explorer, the left pane displays (as nodes) the application systems supported by Application Explorer, based on the adapters that are installed.

Defining a Target to J.D. Edwards OneWorld

To connect to an application system for the first time, you must define a new target.

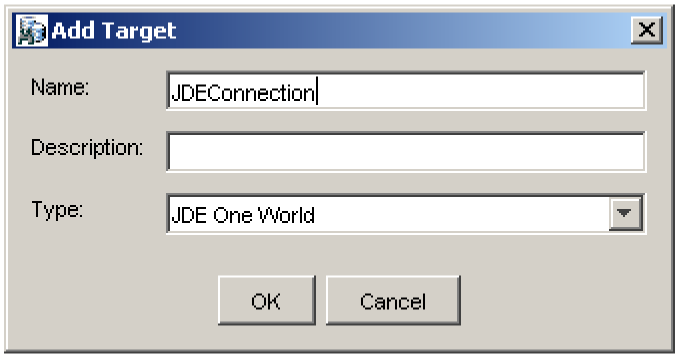
To define a target:

1. In the left pane, expand the **Adapters** node.

   The applications systems supported by Application Explorer appear as nodes based on the adapters that are installed.

2. Right-click the **JDEdwards** node and select **Add Target**.

   The Add Target dialog box is displayed.



Name: JDEConnection
Description:
Type: JDE One World

   Perform the following steps:

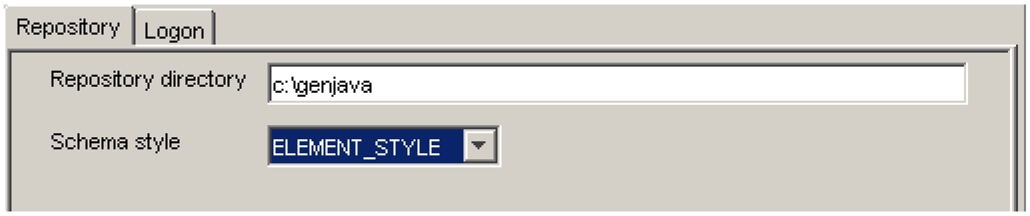
   a. In the **Name** field, enter a descriptive name, for example, JDEConnection.

   b. In the **Description** field, enter a description for the target (optional).

   c. From the **Type** list, select **JDE One World**.

3. Click **OK**.

   The JDE One World dialog box appears.



Repository directory: c:\genjava

Schema style: ELEMENT_STYLE

   a. In the **Repository** tab, enter the path to the GenJava repository in the Repository directory field.

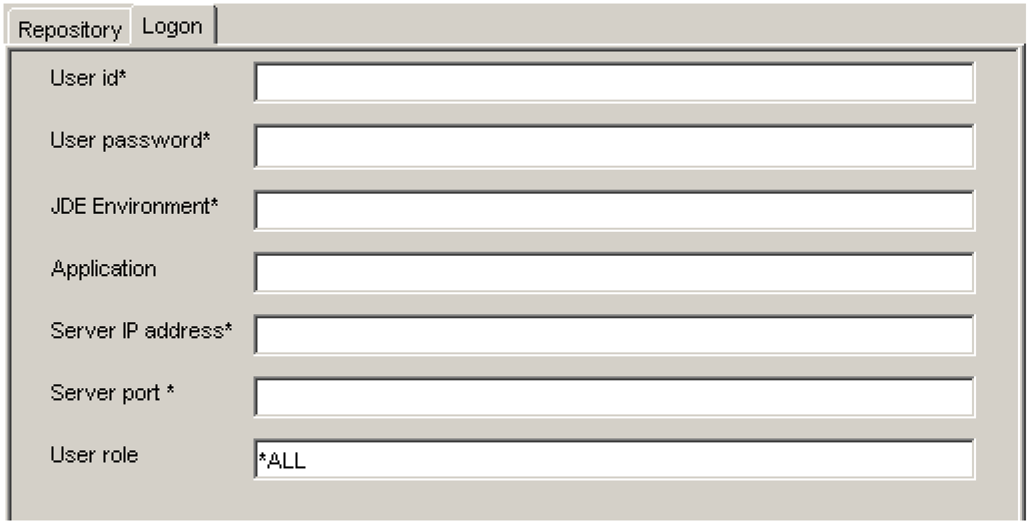
      This is the location of the Java files created by the GenJava program.

      Note: Generating agent schemas requires the GenJava repository. For more information on building the J.D. Edwards OneWorld Master Business Function repository, see the *J.D. Edwards Interoperability Guide for OneWorld Xe.*

      ---

   b. From the **Schema style** list, select **ELEMENT_STYLE** or **ATTRIBUTE_STYLE**.

   c. Click the **Logon** tab and enter the appropriate information for your target type based on the information in the following table. Fields marked with an asterisk are required.



User id*

User password*

JDE Environment*

Application

Server IP address*

Server port *

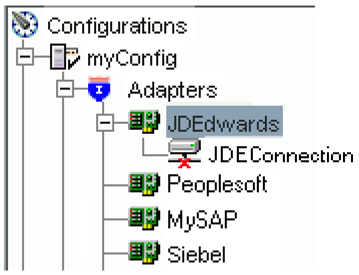
User role: *ALL

Parameter	Description
User id*	A valid user ID for J.D. Edwards OneWorld.
User password*	The password associated with the user ID.
JDE environment*	The J.D. Edwards OneWorld environment, for example, DU7333. For more information about this parameter, see your J.D. Edwards OneWorld documentation or ask your OneWorld system administrator.
Application	XMLInterop or the application name in J.D. Edwards OneWorld. Optional.

Parameter	Description
Server IP address*	The name of the server on which J.D. Edwards OneWorld is running. This can be the name of the server, for example, JDEOW, or its IP address.
Server Port*	The port number on which the server is listening, for example, 6009.
User role	This property setting depends on your J.D. Edwards system version. ▪ For **B7333**: You can leave this field blank. ▪ For **EnterpriseOne 8.10**: You must specify ***ALL**.

4. Click **OK**.

   After the extraction finishes, the new target, JDEConnection, appears under the JDEdwards node.



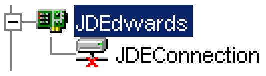
   For information on how to create schemas for the adapter, see "Creating an XML Schema" on page 2-13.

Connecting to a Defined J.D. Edwards OneWorld Target

To connect to a target:

1. Expand the **Service Adapters** node.
2. Expand the **JDEdwards** node.

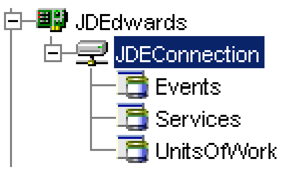
   The disconnected target is displayed.



3. Click the target name (for example, JDEConnection) under the JDEdwards node.
4. Click the **Logon** tab on the right.

   The Logon tab displays the values you entered for connection parameters.
5. Verify your connection parameters.
6. Provide the correct password.
7. Right-click the target name and select **Connect**.

   The x icon disappears, indicating that the node is connected.

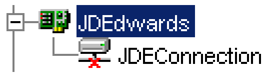


Disconnecting from J.D. Edwards OneWorld

To disconnect from a target:

1. Expand the **Adapters** node.
2. Expand the **JDEdwards** node.
3. Right-click the target to which you are connected (for example, JDEConnection), and select **Disconnect**.

   The x icon appears, indicating that the node is disconnected.



Editing a Target

To edit a target:

1. In the left pane, ensure that the target you wish to edit is disconnected.
2. Right-click the target and select **Edit**.

   A window is displayed that enables you to edit the existing connection parameters.
3. Modify the target information.
4. Click **OK**.

Deleting a Target

You can delete a target, rather than just disconnecting and closing it. When you delete the target, the node disappears from the list of J.D. Edwards OneWorld targets in the left pane of the explorer.

To delete a target:

1. Expand the **Adapters** node.
2. Expand the **JDEdwards** node.
3. Right-click the target to which you are connected (for example, JDEConnection), and select **Delete**.

   The node disappears from the list of available connections.

For information on how to view application system objects, see *J.D .Edwards Interoperability Guide Release OneWorld XE*.

Creating an XML Schema

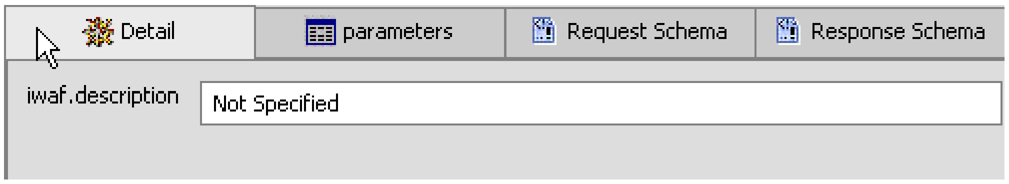
To execute an MBF, the adapter must receive a request document through the J.D. Edwards OneWorld ThinNet API. The agent processes the request and sends an XML response document indicating the result. Application Explorer creates both the XML request schema and the XML response schema.

Creating a Request and a Response Schema

The following procedure explains how to create request and response schemas for a J.D. Edwards OneWorld business function. Application Explorer enables you to create XML schemas for this function.

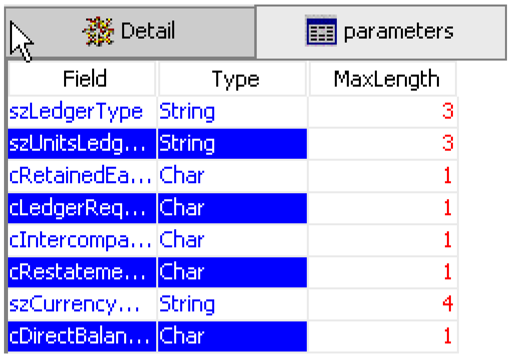
1. Connect to a J.D. Edwards OneWorld target as described in "Connecting to a Defined J.D. Edwards OneWorld Target" on page 2-12.
2. Expand the **Services** node.
3. Expand the node of the MBF for which you want to create the schema.
4. Expand and then select the node beneath the MBF.

   The following image shows the tabs that appear on the right.



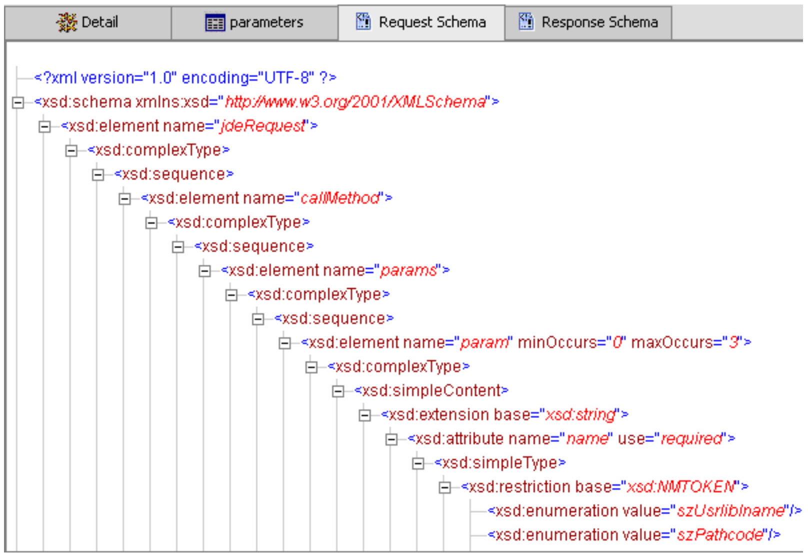
iwaf.description: Not Specified

5. Click the **parameters** tab to view the parameter information.



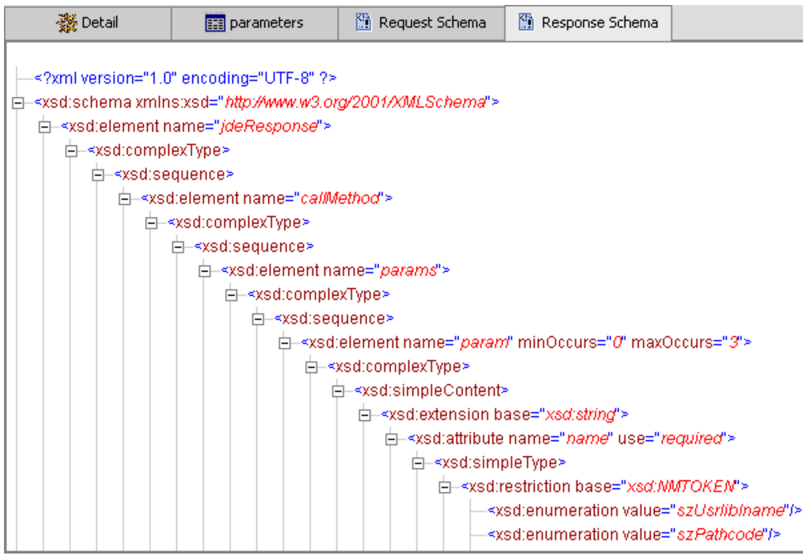
Field	Type	MaxLength
szLedgerType	String	3
szUnitsLedg...	String	3
cRetainedEa...	Char	1
cLedgerReq...	Char	1
cIntercompa...	Char	1
cRestateme...	Char	1
szCurrency...	String	4
cDirectBalan...	Char	1

6. Click **Request Schema** to view the request schema information.



```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="jdeRequest">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="callMethod">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="params">
        <xsd:complexType>
         <xsd:sequence>
          <xsd:element name="param" minOccurs="0" maxOccurs="3">
           <xsd:complexType>
            <xsd:simpleContent>
             <xsd:extension base="xsd:string">
              <xsd:attribute name="name" use="required">
               <xsd:simpleType>
                <xsd:restriction base="xsd:NMTOKEN">
                 <xsd:enumeration value="szUsrlibIname"/>
                 <xsd:enumeration value="szPathcode"/>
```

7. Click **Response Schema** to view the response schema information.



```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="jdeResponse">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="callMethod">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="params">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="param" minOccurs="0" maxOccurs="3">
                      <xsd:complexType>
                        <xsd:simpleContent>
                          <xsd:extension base="xsd:string">
                            <xsd:attribute name="name" use="required">
                              <xsd:simpleType>
                                <xsd:restriction base="xsd:NMTOKEN">
                                  <xsd:enumeration value="szUsrlibIname"/>
                                  <xsd:enumeration value="szPathcode"/>
```

Using GenJava to Generate a Schema

To create schemas for the adapter, you must use GenJava wrappers. You create the GenJava wrappers using the OneWorld utility called GenJava. You use Application Explorer to generate schemas against OneWorld GenJava wrappers. GenJava is supplied as a command line process with several runtime options.

See Also: *J.D. Edwards Interoperability Guide for OneWorld Xe*

Generating WSDL (J2CA Configurations Only)

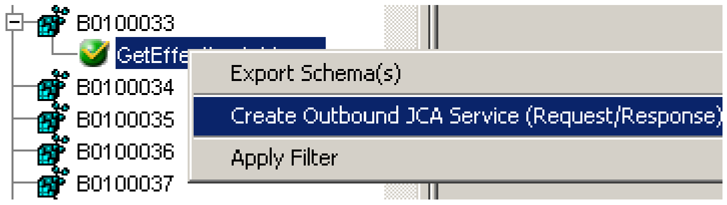
The procedure for generating WSDL (Web Service Definition Language) for request-response (outbound) services differs from that of generating WSDL for event notification (inbound) J2CA services of the adapter. Each procedure is described in detail in the following sections.

Generating WSDL for Outbound Interaction

To generate a WSDL file for request-response service:

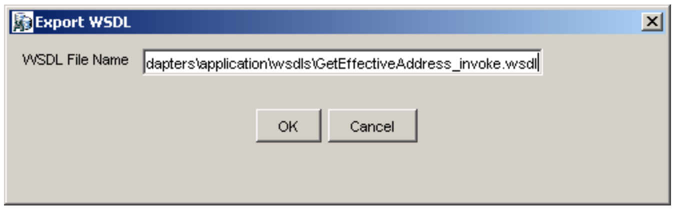
1. Start Application Explorer and connect to a defined J.D. Edwards target.
2. Expand **Services**, **JDEJAVA_CFIN**, and then **B0100033**. Select **GetEffectiveAddress**.
3. Right-click **GetEffectiveAddress**.

   The following menu is displayed.



4. Select **Create Outbound JCA Service (Request/Response)**.

The Export WSDL dialog box is displayed.



5. Accept the default name and location for the file.

   The **.wsdl** file extension is added automatically. By default, the names of WSDL files generated for request-response services end with `_invoke`, while those generated for event notification end with `_receive`.

   > **Note:** You can organize your WSDL files in subfolders, creating your own WSDL hierarchy structure. Create the folders under `*OracleAS_home*\adapters\application\wsdls\`. The WSIL browser in JDeveloper will display the full tree structure of your WSDL hierarchy.

6. Click **OK**.

   The WSDL file is saved in the specified location.

Generating WSDL for Inbound Interaction

You can create inbound J2CA service only if the node you have selected supports events. To generate a WSDL file:

1. Create a channel in Application Explorer under the J.D. Edwards events node.
2. Start the channel.

   Important: Do not restart the BPEL PM Server or Oracle Application Server after the channel is started.
3. Send an inbound message from J.D. Edwards OneWorld.
4. Capture the inbound message payload in the log file located under `*OracleAS_home*\adapters\application\config\jca_sample\log\iwaf_jca15.log`

   Alternatively, you can create a port with the File protocol under Event in Application Explorer to dispose the event message to the file system.
5. Use a third party tool such as XMLSpy to create the XSD schema using the XML payload captured in the previous step.
6. Verify that `orabpel-adapters.jar` is in your classpath.
7. Open a command prompt and change directory to: `*OracleAS_home*\adapters\application\tools`
8. Run `obadapter.bat` to set the environment.
9. Enter the following command to generate WSDL:

```
java com.iwaysoftware.af.container.tools.wsdl.IWayWSILBrowser jcaHome adapter
target channel schemaPrefix wsdlFileName
```

Where:

- `jcaHome` is the path to your J2CA configuration. For example, *`OracleAS_home`*`\adapters\application\`
- `adapter` is the name of the adapter. For example, JDEdwards.
- `target` is the name of the adapter target you created under Adapter in Application Explorer
- `channel` is the name of the channel you have created under Event in Application Explorer
- `schemaPrefix` is the prefix of the XSD schema. The schema file must be in the same directory where the Java command is executed, for example, *`OracleAS_home`*`\adapters\application\config\jca_sample\schemas\JDEdwards\bvision02`
- `wsdlFileName` is the name of the WSDL file being created in this procedure

For example:

```
C:\soaga\adapters\application\config\jca_
sample\schemas\JDEdwards\bvision02>java
com.iwaysoftware.af.container.tools.wsdl.IWayWSILBrowser
c:\soaga\adapters\application\  JDEdwards bvision02 JDE PO purchaseorder.wsdl
```

Once the command is successfully executed, the following output will be shown in the command window:

```
Running Inbound WSDL generation tool...
  -> user.dir = C:\soaga\adapters\application\config\jca_
sample\schemas\JDEdwards\bvision02
  -> Generating WSDL...
  -> Done.
  -> Writing WSDL 'c:\soaga\adapters\application\\wsdls\purchaseorder.wsdl' to
disk...
  -> Done.
```

Note: It is good practice to append `_receive` to the names of WSDL files generated for event notification services. This will allow you to easily differentiate between them and those generated for request-response services.

10. Stop the channel in Application Explorer.

Note: You can organize your WSDL files in subfolders, creating your own WSDL hierarchy structure. Create the folders under *`OracleAS_home`*`\adapters\application\wsdls\`. The WSIL browser in JDeveloper will display the full tree structure of your WSDL hierarchy.

The WSDL you generate using Application Explorer is used in a BPEL process as a Partner Link. This is described in detail in "J.D. Edwards OneWorld Event Integration" on page 5-13.

Creating and Testing a Web Service (BSE Configurations Only)

You can generate a Web service (also known as a **business service**) using Application Explorer. You can explore the business function repository and generate Web services for the functions you want to use with the adapter. The following procedure uses an example called BusinessUnitExistenceCheck.

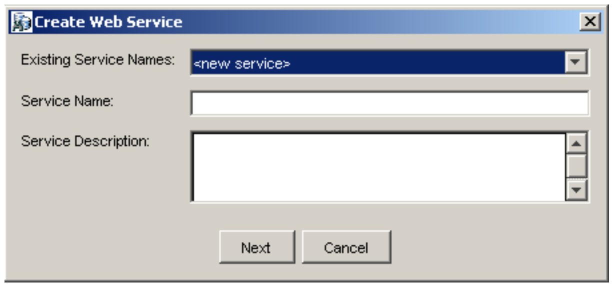
Note: In a J2EE Connector Architecture (J2CA) implementation, Web services are not available. When the adapters are deployed to use J2CA, the Common Client Interface (CCI) provides integration services.

Creating a Web Service

To create a Web service for a business function:

1. Expand the **JDEdwards** node and then the **Services** node.
2. Expand the MBF, **B1000012**, also called BusinessUnitExistenceCheck.
3. Right-click the node from which you want to create a business service and select **Create Web Service**.

   The Create Web Service dialog box is displayed.



Create Web Service

Existing Service Names: <new service>

Service Name:

Service Description:

   You can add the business function as a method for a new Web service or as a method for an existing one.

   a. From the **Existing Service Names** list, select either **<new service>** or an existing service.
   b. In the **Service Name** field, specify a service name if you are creating a new service. This name identifies the Web service in the list of services under the Business Services node.
   c. Enter a description for the service (optional).

4. Click **Next**.

   Perform the following steps:

   a. In the **License Name** field, select one or more license codes to assign to the Web service.
   b. In the **Method Name** field, enter a descriptive name for the method or accept the default name.
   c. In the **Description** field, enter a brief description of the method (optional).
   d. In the **DTD Directory** field, specify a location where the Web service will be saved. If you wish to select a location different than the default, click **Browse** and navigate to the desired location.

5. Click **OK**.

   Application Explorer switches the view to the Business Services node, and the new Web service appears in the left pane.

Testing a Web Service

After a Web service is created, you can test it to ensure it functions properly. A test tool is provided for testing the Web service.

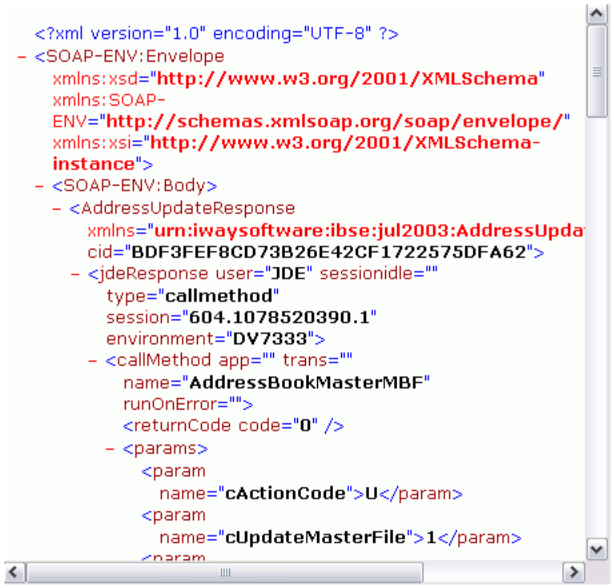
To test a Web service:

1. Click the **Business Services** node to access your Web services.
2. Expand the **Services** node.
3. Select the name of the business service you want to test.

   The business service name appears as a link in the right pane.
4. In the right pane, click the named business services link.

   The test option appears in the right pane. If you are testing a Web service that requires XML input, an input field appears.
5. Enter the appropriate input.
6. Click **Invoke**.

   Application Explorer displays the results.



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:SOAP-
    ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance">
  - <SOAP-ENV:Body>
    - <AddressUpdateResponse
        xmlns="urn:iwaysoftware:ibse:jul2003:AddressUpda
        cid="BDF3FEF8CD73B26E42CF1722575DFA62">
      - <jdeResponse user="JDE" sessionidle=""
          type="callmethod"
          session="604.1078520390.1"
          environment="DV7333">
        - <callMethod app="" trans=""
            name="AddressBookMasterMBF"
            runOnError="">
            <returnCode code="0" />
          - <params>
              <param
                name="cActionCode">U</param>
              <param
                name="cUpdateMasterFile">1</param>
```

Identity Propagation

If you test or execute a Web service using a third party XML editor, for example XMLSPY, the Username and Password values that you specify in the SOAP header must be valid and are used to connect to J.D. Edwards OneWorld. The user name and password values that you provided for J.D. Edwards OneWorld during target creation using Application Explorer are overwritten for this Web service request. The following is a sample SOAP header that is included in the WSDL file for a Web service:

```
<SOAP-ENV:Header>
  <m:ibsinfo xmlns:m="urn:schemas-iwaysoftware-com:iwse">
    <m:service>String</m:service>
    <m:method>String</m:method>
    <m:license>String</m:license>
    <m:disposition>String</m:disposition>
    <m:Username>String</m:Username>
    <m:Password>String</m:Password>
    <m:language>String</m:language>
  </m:ibsinfo>
</SOAP-ENV:Header>
```

You can remove the `<m:disposition>` and `<m:language>` tags from the SOAP header, since they are not required.

Configuring an Event Adapter

Events are generated as a result of activity in a database or in an application system. You can use events to trigger an action in your application. For example, an update to a database can reflect an update to customer information. If your application must perform when this happens, your application is a consumer of this event.

After you create a connection to your application system, you can add events using Application Explorer. To create an event, you must create a port and a channel.

Note: If you are using a J2CA configuration, you must create a new channel for every event and select this channel when you generate WSDL. Additionally, you are not required to create or configure ports for use with BPEL Process Manager.

- A **port** associates a particular business object exposed by the adapter with a particular disposition. A disposition is a URL that defines the protocol and location of the event data. The port defines the end point of the event consumption. For example, you can use the MSMQ protocol to route the result of a Purchase Order update in the J.D. Edwards OneWorld system to a queue hosted by your application server. See "Creating and Editing an Event Port" on page 2-20 for more information.
- A **channel** represents configured connections to particular instances of back-end systems. A channel binds one or more event ports to a particular listener managed by the adapter. See "Creating and Editing a Channel" on page 2-22 for more information.

Note: Oracle Containers for J2EE (OC4J) currently conforms to J2CA 1.0, which does not call for event capabilities. When conforming to J2CA 1.0, only service interactions are supported.

Creating and Editing an Event Port

You create event ports from the Events node in Application Explorer.

Note: You are not required to create event ports for J2CA configurations. You must create event ports for BSE configurations only.

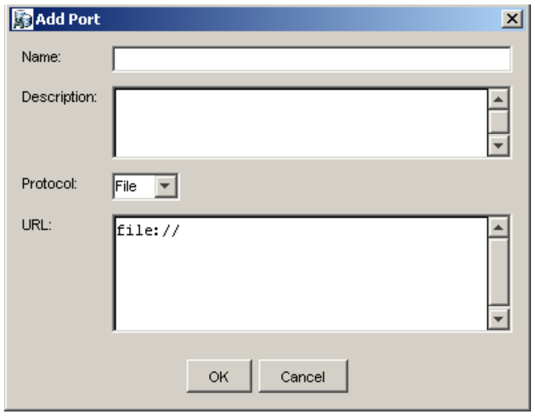
Creating an Event Port for File

The following procedure describes how to create an event port for File from the Events node using Application Explorer.

1. Expand the **Events** node.
2. Expand the **JDEdwards** node.

3. Right-click **Ports** and select **Add Port**.

   The Add Port dialog box is displayed.



   Perform the following steps:

   a. Enter a name for the event port.
   b. Provide a brief description (optional).
   c. From the disposition **Protocol** list, select **File**.
   d. In the **URL** field, update the File URL. For example:

      `file://`*`location`*

   The following table defines the parameters for the File disposition.

Parameter	Description
location	Full directory path and file name to which the data is written..

4. Click **OK**.

   The port appears under the ports node in the left pane. In the right pane, a table appears that summarizes the information associated with the event port you created.

   You are ready to associate the event port with a channel. For more information, see "Creating and Editing a Channel" on page 2-22.

Editing an Event Port

To edit an event port using Application Explorer:

1. Expand the **Event Adapters** node.
2. Expand the **JDEdwards** node.
3. Right-click the event port you want to edit and select **Edit**.

   The Edit Port pane is displayed.
4. Make the necessary edits and click **OK**.

Deleting an Event Port

To delete an event port using Application Explorer:

1. Expand the **Event Adapters** node.

2. Expand the **JDEdwards** node.
3. Right-click the event port you want to delete and select **Delete**.

   A confirmation dialog box is displayed.
4. To delete the event port you selected, click **OK**.

   The event port disappears from the list in the left pane.

Creating and Editing a Channel

The following procedure describes how to create a channel for your event. All defined event ports must be associated with a channel.

Note: If you are using a J2CA configuration, you must create a new channel for every event and select this channel when you generate WSDL. Creating a channel is required for both BSE and J2CA configurations.

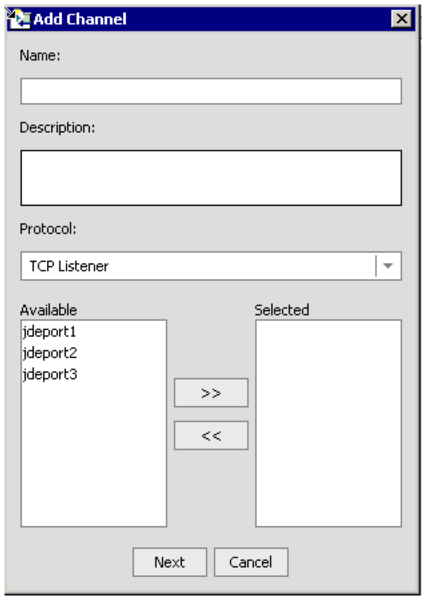
Creating a Channel

To create a channel:

1. Click **Event Adapters**.
2. Expand the **JDEdwards** node.

   The ports and channels nodes appear in the left pane.
3. Right-click **Channels** and select **Add Channel**.

   The Add Channel dialog box is displayed.


Add Channel

Name:

Description:

Protocol: TCP Listener

Available: jdeport1, jdeport2, jdeport3

Selected

   Perform the following steps:

   a. Enter a name for the channel, for example, NewChannel.
   b. Enter a brief description.
   c. From the disposition **Protocol** list, select **TCP Listener**.

   d. Select an event port from the list of available ports. To select more than one, hold down the **Ctrl** key and select the ports.
   e. Click >> to transfer the port(s) to the list of selected ports.

4. Click **Next**.

   The TCP Listener dialog box is displayed with the Basic tab active. Perform the following steps:

   a. Enter the parameters that are specific to your J.D. Edwards environment.
   b. Click the **preparser** tab.
   c. Enter the required parameters.

   The following table lists the parameters with their descriptions. Parameters with an asterisk are required.

Parameter	Description
Host*	Name or URL of the machine where the database resides.
Port Number*	Port on which the Host database is listening.

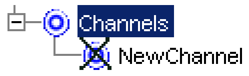
Synchronization Type	Possible values are:
	`RECEIVE_REPLY`
	`RECEIVE_ACK`
	`RECEIVE`

Is Length Prefix	For J.D. Edwards OneWorld events that send data back that is not in XML format. The TCP/IP event application must prefix the data with a 4-byte binary length field when writing the data to the TCP/IP port.
Is XML	For J.D. Edwards OneWorld events that send data back in XML format. No preparser is required.
Is Keep Alive	Maintains continuous communication between the event transaction and the channel.
User id*	A valid user ID for J.D. Edwards OneWorld.
User password*	The password associated with the user ID.
JDE Environment*	The J.D. Edwards OneWorld environment, for example, DU7333. For more information about this parameter, see your J.D. Edwards OneWorld documentation or ask your OneWorld system administrator.
Application	XMLInterop or the application name in J.D. Edwards OneWorld. Optional.
Server IP address*	Name of the server on which J.D. Edwards OneWorld is running. This can be the name of the server, for example, JDEOW, or its IP address, for example, 123.45.67.89.
Server port*	Port number on which the server is listening, for example, 6009.

   For additional parameters, see your J.D. Edwards OneWorld Administrator.

5. Click **OK**.

   The channel appears under the channels node in the left pane.

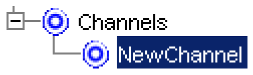


   An X over the icon indicates that the channel is currently disconnected. You must start the channel to activate your event configuration.

Note: If you are using OracleAS Adapter for J.D. Edwards OneWorld with BPEL Process Manager, do not start the channel, as it is managed by the BPEL PM Server. If you start the channel for testing and debugging purposes, stop it before runtime.

6. Right-click the channel node and select **Start**.

   The channel becomes active.



   The X over the icon disappears.

7. To stop the channel, right-click the connected channel node and select **Stop**.

   The channel becomes inactive and an X appears over the icon.

Editing a Channel

To edit a channel:

1. In the left pane, locate the channel you want to edit.
2. Right-click the channel and select **Edit**.

   The Edit channels pane is displayed.

3. Make the required changes to the channel configuration and click **Finish**.

Deleting a Channel

To delete a channel:

1. In the left pane, locate the channel you want to delete.
2. Right-click the channel and select **Delete**.

   A confirmation dialog box is displayed.

3. To delete the channel you selected, click **OK**.

   The channel disappears from the list in the left pane.

The OneWorld Event Listener

OracleAS Adapter for J.D. Edwards OneWorld Event Listener is designed specifically to provide J.D. Edwards approved access to your OneWorld business events. The OneWorld Event Listener refers to a specialized application that runs in conjunction with OneWorld business functions and is called by the OneWorld application system.

The OneWorld application system provides the Event Listener with the information required to retrieve the event information for only the desired events. For information about configuring the OneWorld environment, see the *J.D. Edwards Interoperability Guide for OneWorld.*

The OneWorld Event Listener is called directly from the OneWorld application and is passed a Z-file record identifier. This identifier then generates a request document that is passed to the server for processing. The server retrieves the event information from the J.D. Edwards OneWorld system and propagates the information for integration with other application systems.

Configuring the OneWorld Event Listener

The OneWorld Event Listener is installed as part of the basic installation. The OneWorld Adapter is automatically installed in the appropriate directory. If the integration server is not installed on the same computer as the J.D. Edwards application server, you must configure the OneWorld Event Listener.

The OneWorld Event Listener is invoked by J.D. Edwards for specific transactions as configured in the OneWorld environment.

The OneWorld Event Listener includes the following components:

- The listener exit `(IWOEvent),` located under *`adapters_home`*`\etc\jde`, where `adapters_home` is *`OracleAS_home`*`\adapters\application\`. For example:

  ```
  C:\OraHome_1\adapters\application\etc\jde\Iwoevent.dll
  ```

  The file extension varies depending on your operating system:

  – For **Windows**, the exit is `Iwoevent.dll`.

  – For **Sun Solaris**, the exit is `libiwoevent.so`.

  – For **HP-UX**, the exit is `libiwoevent.sl`.

  – For **AS/400**, the exit is `iwaysav.sav`.

  – For **IBM AIX**, the exit is `libiwoevent.so`.

- The listener configuration file `(iwoevent.cfg).`

  See "Creating the iwoevent.cfg File" on page 2-25 and "Adding Connection Information" on page 2-26 for more information on this file.

- The outbound agent `(XDJdeOutboundAgent).`

The OneWorld Event listener exit is the function that passes the key fields for a record in the OneWorld outbound transaction tables to the integration server for processing by the outbound agent. The OneWorld Event listener is deployed under the J.D. Edwards OneWorld Enterprise Server. The Java class for the OneWorld Event listener is called IWOEvent (the file extension depends on the operating system) and is case-sensitive.

Creating the iwoevent.cfg File

After OneWorld starts the OneWorld Event listener, the listener accesses the configuration file, called `iwoevent.cfg` (case-sensitive). Based on the information in the configuration file, the listener sends the event notification to the integration server. If the integration server is unavailable or some exception occurs, the OneWorld Event listener saves the event information in a file called `batch.log`. After the server becomes available, the listener sends the information. All log information is saved in a file called `iwoevent.log`.

To create the `iwoevent.cfg` file:

1. On the J.D. Edwards OneWorld Server, create an `iwoevent.cfg` file in the defined directory. See "Adding Connection Information" for information about the contents of this file.

2. Create an environment variable, *`IWOEVENT_HOME`*, to point to the directory containing the `iwoevent.cfg` file.

   - On Windows: Add *`IWOEVENT_HOME`* to the system environment variables.

   - On UNIX: Add the following command to your start-up script:

```
export IWOEVENT_HOME =/directory_name
```

Adding Connection Information

The OneWorld Event listener requires connection information for the associated adapter to initiate events properly. This information is contained in the `iwoevent.cfg` file. You must create this file and add the connection information to it. The OneWorld Event Listener requires connection information for the associated integration server to function properly. This information is contained in the `iwoevent.cfg` file. A sample `iwoevent.cfg` file is installed on the J.D. Edwards server and is in the root path. The `iwoevent.cfg` file has three distinct sections:

- **Common**

  The common section of the configuration file contains basic configuration options. Currently, only the trace option is supported.

- **Alias**

  The alias section of the configuration file contains the connection information required to send transactions to specific servers. The alias values to these entries are as follows:

  ```
  Alias.aliasname={ipaddress|dsn}:port, trace={on|off}
  ```

  Where:

  `aliasname` is the symbolic name given to the connection.

  `ipaddress|dsn` is the IP address or DSN name for the server containing OracleAS Adapter for J.D. Edwards OneWorld (required).

  `port` is the port defined for OracleAS Adapter for J.D. Edwards OneWorld (required).

  `trace={on|off}` sets the tracing to on for the particular alias.

- **Trans**

  The trans section of the configuration file contains transaction information required to route J.D. Edwards OneWorld transactions to specified servers.

  If a particular J.D. Edwards OneWorld transaction is not defined to an alias, it is sent to all aliases. The trans values to these entries are as follows:

  ```
  trans.jdeTransactionName=alias1,alias2,aliasn
  ```

  Where `jdeTransactionName` is the JDE-defined name for the outbound transaction and `alias1,alias2,aliasn` is the list of aliases to which the transactions are sent.

Adding Connection Information to iwoevent.cfg

To add connection information to the iwoevent.cfg file:

1. Add the server and port entries to the `iwoevent.cfg` file.
2. To set the trace option, select **on** or **off**.

   ```
   common.trace=on|off
   ```

   Where `on` sets the tracing to on and `off` sets the tracing to off. Off is the default value.

The following is a sample entry from `iwoevent.cfg` that supplies connection information:

```
common.trace=on
alias.edamcs1=172.1.1.1:3694
alias.edamcs1t=172.1.1.1:3694, trace=on
alias.edamcs2=222.2.2.2:1234
trans.JDESOW=edamcs1t,edamcs2
trans.JDEPOOUT=edamcs1
```

3 OC4J Deployment and Integration

This chapter describes Oracle Containers for J2EE (OC4J) deployment and integration with OracleAS Adapter for J.D. Edwards OneWorld.

This chapter discusses the following topics:

See Also:

- *Oracle Application Server Adapter Concepts*

Adapter Integration with OC4J

OracleAS Adapter for J.D. Edwards OneWorld is deployed within an OC4J container during installation. All client applications run within the OC4J environment. In J2CA deployment, the Common Client Interface (CCI) integrates an OC4J client application with a resource adapter.

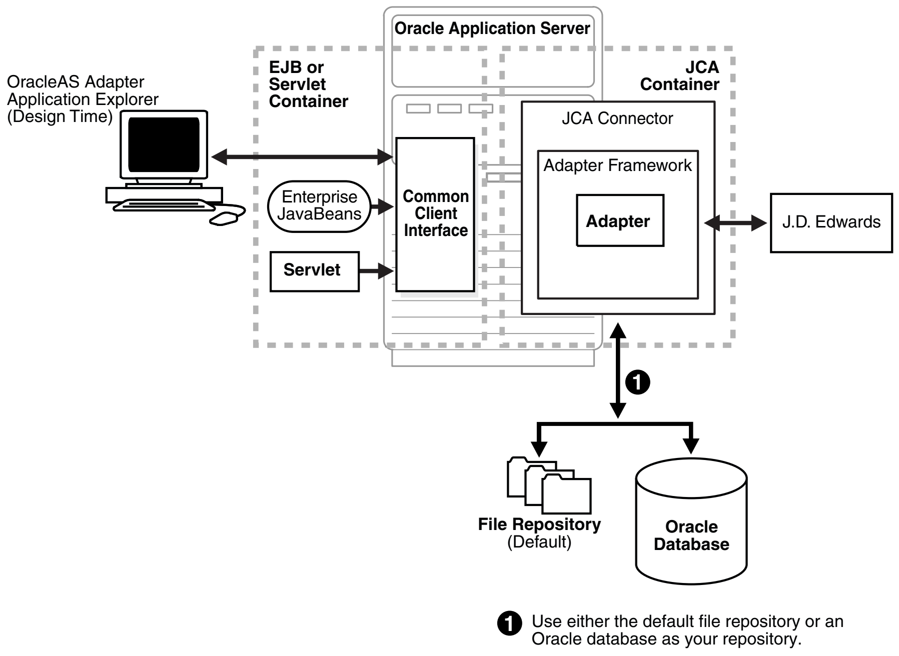
See Also:

- Oracle Application Server Adapters Integration with OC4J in *Oracle Application Server Adapter Concepts*

Deployment of Adapter

Figure 3–1 shows deployment of the Connector to the Oracle Application Server. In a runtime service scenario, an Enterprise Java Bean (EJB), Servlet, or Java program client makes CCI calls to J2CA resource adapters. The adapters process the calls as requests and send them to the EIS. The EIS response is then sent back to the client.

Figure 3–1 Oracle Application Server Adapter J2CA Architecture


Oracle Application Server
OracleAS Adapter Application Explorer (Design Time)
EJB or Servlet Container
JCA Container
JCA Connector
Adapter Framework
Adapter
J.D. Edwards
Enterprise JavaBeans
Common Client Interface
Servlet
1
File Repository (Default)
Oracle Database
1 Use either the default file repository or an Oracle database as your repository.

See Also:

- *Oracle Application Server Adapter Concepts*

Updating Adapter Configuration

During the J2CA deployment of OracleAS Adapter for J.D. Edwards OneWorld, OC4J generates a deployment descriptor called `oc4j-ra.xml`, located in *`OC4J_home`*`\j2ee\home\application-deployments\default\jca_app_adapter`.

Note: Your installation directory contains more than one file named `oc4j-ra.xml`. The OC4J deployment descriptor described in this section is located in the directory specified earlier.

Creating a Managed Connector Factory Object

The `oc4j-ra.xml` descriptor provides OC4J-specific deployment information for resource adapters. For example, the default `jca_sample` configuration in Application Explorer is represented in the `oc4j-ra.xml` file as follows:

```
<?xml version="1.0"?>
<!DOCTYPE oc4j-connector-factories PUBLIC "-//Oracle//DTD Oracle Connector
9.04//EN"
"http://xmlns.oracle.com/ias/dtds/oc4j-connector-factories-9_04.dtd">
<oc4j-connector-factories>
   <connector-factory location="eis/OracleJCAAdapter/DefaultConnection"
connector-name="IWAFJCA10">
   <config-property name="IWayHome" value="../../adapters/application"/>
   <config-property name="IWayConfig" value="jca_sample"/>
   <config-property name="IWayRepoURL" value=""/>
   <config-property name="IWayRepoUser" value=""/>
   <config-property name="IWayRepoPassword" value=""/>
   <config-property name="logLevel" value="debug"/>
   </connector-factory>
</oc4j-connector-factories>
```

The parameters defined in the `oc4j-ra.xml` file are described in the following table:

Parameter Name	Description
IWayHome	The base installation directory for the OracleAS packaged application adapter.
IWayConfig	The adapter configuration name as defined in Application Explorer. For example, OracleAS Adapter for J.D. Edwards OneWorld has a preconfigured `jca_sample` configuration in Application Explorer.
IWayRepoURL	The URL to use when opening a connection to the database. This is necessary only when using an Oracle database as the BSE repository. See "Configuring BSE System Settings" on page 2-3 for more information.
IWayRepoUser	User name to use when connecting to the database. This is necessary only when using an Oracle database as the BSE repository. See "Configuring BSE System Settings" on page 2-3 for more information.
IWayRepoPassword	Password. If provided, it overwrites configuration. This is necessary only when using an Oracle database as the BSE repository. See "Configuring BSE System Settings" on page 2-3 for more information.
loglevel	It overwrites the level set by the ManagedConnectionFactory property.

Creating Multiple Managed Connector Factory Objects

To establish multiple managed connector factory objects, you must edit the `oc4j-ra.xml` file and add more `<connector-factory>` nodes. For example, the default `jca_sample` configuration in Application Explorer is represented in the `oc4j-ra.xml` file as follows:

```
<?xml version="1.0"?>
<!DOCTYPE oc4j-connector-factories PUBLIC "-//Oracle//DTD Oracle Connector
9.04//EN"
"http://xmlns.oracle.com/ias/dtds/oc4j-connector-factories-9_04.dtd">
<oc4j-connector-factories>
   <connector-factory location="eis/OracleJCAAdapter/DefaultConnection"
connector-name="IWAFJCA10">
   <config-property name="IWayHome" value="../../adapters/application"/>
   <config-property name="IWayConfig" value="jca_sample"/>
   <config-property name="IWayRepoURL" value=""/>
   <config-property name="IWayRepoUser" value=""/>
   <config-property name="IWayRepoPassword" value=""/>
   <config-property name="logLevel" value="debug"/>
   </connector-factory>
</oc4j-connector-factories>
```

To create multiple managed connector factory objects, you must add new <connector-factory> nodes in the file. For example:

```
<?xml version="1.0"?>
```

```
<!DOCTYPE oc4j-connector-factories PUBLIC "-//Oracle//DTD Oracle Connector
9.04//EN"
"http://xmlns.oracle.com/ias/dtds/oc4j-connector-factories-9_04.dtd">
<oc4j-connector-factories>
   <connector-factory location="eis/OracleJCAAdapter/DefaultConnection1"
connector-name="IWAFJCA10">
   <config-property name="IWayHome" value="../../adapters/application"/>
   <config-property name="IWayConfig" value="jca_sample"/>
   <config-property name="IWayRepoURL" value=""/>
   <config-property name="IWayRepoUser" value=""/>
   <config-property name="IWayRepoPassword" value=""/>
   <config-property name="logLevel" value="debug"/>
   </connector-factory>
<connector-factory location="eis/OracleJCAAdapter/DefaultConnection2"
connector-name="IWAFJCA10">
   <config-property name="IWayHome" value="../../adapters/application"/>
   <config-property name="IWayConfig" value="jca_sample2"/>
   <config-property name="IWayRepoURL" value=""/>
   <config-property name="IWayRepoUser" value=""/>
   <config-property name="IWayRepoPassword" value=""/>
   <config-property name="logLevel" value="debug"/>
   </connector-factory>
</oc4j-connector-factories>
```

How to Write a Java Application Client Using the CCI API

The following example shows the code structure for using CCI with packaged application adapters. The code sample is shown in four steps.

Step 1. Obtain the Connection Factory

The connection factory is obtained by JNDI lookup.

```
InitialContext context = new InitialContext();
ConnectionFactory cf = (ConnectionFactory)context.lookup(iwayJndi)
```

Step 2. Obtaining a Connection for the Adapter

IWAFConnectionSpec is an implementation of ConnectionSpec used for creating a design time or runtime service adapter connection. The ConnectionSpec has seven parameters. Connection Pooling is fully supported and established based on these parameters, except log level.

Parameter Name	Description
adapterName	Name of the packaged application adapter.
config -	Adapter configuration name. NOT REQUIRED FOR IWAEAdapter.
language	Default is `en`.
country	Default is `us`.
userName	User name. If provided, it overwrites configuration.
password	Password. If provided, it overwrites configuration.
logLevel	It overwrites the level set by the ManagedConnectionFactory property.

Note: Currently the OracleAS Adapter J2CA supports only basic security mapping. The DEBUG log level provides detailed information on the mapping behavior. It functions as follows:

- If the user name and password are not set, and no security is provided by the application server, the OracleAS Adapter J2CA will still let it pass and rely on the adapter configuration security information.
- If the user name and password are set, these values will overwrite the adapter configuration. The OracleAS Adapter J2CA compares this information with the security information provided by the application server and log in case the values do not match. However, it still allows the information through.

The iWAFConnectionSpec can initiate an interaction with J.D. Edwards OneWorld if the adapter name and configuration parameters are specified in the ConnectionSpec. For example,

```
iWAFConnectionSpec cs = new IWAFConnectionSpec();
  cs.setAdapterName(ADAPTER);
  cs.setConfig(TARGET);
  cs.setLogLevel(LOG_LEVEL);  // Adapter layer log level
  Connection c = cf.getConnection(cs);// where cf is the connection factory
```

In this snippet, `ADAPTER` and `TARGET` refer to the adapter being deployed, in this case J.D. Edwards OneWorld, and the name of a target defined in Application Explorer. See "Complete Code Sample" on page 3-7 for more information.

Step 3. Create interaction with interactionSpec for runtime

```
Interaction i = c.createInteraction();
  IWAFInteractionSpec is = new IWAFInteractionSpec();
  is.setFunctionName(IWAFInteractionSpec.PROCESS);
```

Two functions can be set: PROCESS and IWAE. PROCESS is used at runtime. IWAE is used when you are using the IAEAdapter at design time.

Step 4. Create Input Record and Execute Interaction

In this case, to complete the EIS invocation, a schema is provided by Application Explorer.

A standard J2CA Indexed Record is used in this example:

```
// Use JCA IndexRecord, named "input" for runtime processing.
IndexedRecord rIn = cf.getRecordFactory().createIndexedRecord("input");
rIn.add(msg_run);
  IndexedRecord rOut = (IndexedRecord)i.execute(is, rIn);
System.out.println((String)rOut.get(0));
```

A special record is supported in this example:

```
//IWAFRecord rIn = new IWAFRecord("input");
//rIn.setRootXML(msg_run);
//IWAFRecord response = executeRunInteraction(c, rIn);
  //IWAFRecord rOut = (IWAFRecord)i.execute(is, rIn);
//System.out.println(rOut.getRootXML());
```

Where *msg_run* is an instance XML document generated from the schema created by Application Explorer. For example, the following is a sample J.D. Edwards OneWorld request XML document.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<jdeRequest type="callmethod" user="JDE" pwd="JDE" environment="DV7333" session=""
sessionidle="">
<callMethod name="AddressBookMasterMBF" app="" runOnError="" trans="">
<params>
  <param name="cActionCode">A</param>
  <param name="cUpdateMasterFile">1</param>
  <param name="cProcessEdits">1</param>
  <param name="cSuppressErrorMessages"/>
  <param name="szErrorMessageID" />
  <param name="szVersion">ZJDE0001</param>
  <param name="mnSameAsExcept" />
  <param name="mnAddressBookNumber" id="1" />
  <param name="szLongAddressNumber" />
  <param name="szTaxId" />
  <param name="szSearchType">C</param>
  <param name="szAlphaName">John J. Smith</param>
  <param name="szSecondaryAlphaName">John J. Smith</param>
  <param name="szMailingName">John J. Smith</param>
  <param name="szSecondaryMailingName">John J. Smith</param>
  <param name="szDescriptionCompressed" />
  <param name="szBusinessUnit"/>
  <param name="szAddressLine1">1 Main St.</param>
  <param name="szAddressLine2">Apt 101</param>
  <param name="szAddressLine3"/>
  <param name="szAddressLine4"/>
  <param name="szPostalCode">75000</param>
  <param name="szCity">AnyTown</param>
  <param name="szCounty"/>
  <param name="szState">TX</param>
  <param name="szCountry">US</param>
  <param name="szCarrierRoute" />
  <param name="szBulkMailingCenter" />
  <param name="szPrefix1" />
  <param name="szPhoneNumber1">917-123-4567</param>
  <param name="szPhoneNumberType1" />
  <param name="szPhoneAreaCode2" />
  <param name="szPhoneNumber2" />
  <param name="szPhoneNumberType2" />
  <param name="cPayablesYNM">Y</param>
  <param name="cReceivablesYN">N</param>
  <param name="cEmployeeYN">N</param>
  <param name="cUserCode" />
  <param name="cARAPNettingY">N</param>
  <param name="cPersonCorporationCode" />
  <param name="szCertificate" />
  <param name="szAddlIndTaxID" />
  <param name="szCreditMessage" />
  <param name="szLanguage" />
  <param name="szIndustryClassification" />
  <param name="cEMail" />
  <param name="szCategoryCode01" />
  <param name="szRemark"/>
  <param name="szUserReservedCode"/>
  <param name="jdUserReservedDate"/>
  <param name="mnUserReservedAmount"/>
  <param name="mnUserReservedNumber"/>
  <param name="szUserReservedReference"/>
  <param name="jdDateEffective" />
  <param name="szRemark1" /> </params>
<onError abort=""/>
</callMethod>
</jdeRequest>
```

Complete Code Sample

The following is a sample of the complete code:

```
import javax.resource.cci.*;
import com.ibi.afjca.cci.*;
import com.ibi.afjca.spi.*;

/**
 * The purpose of this sample is to illustrate how to use the IWAF Universal
 * JCA connector.
 */
public class IWAFJCASimple {

private static String  HOME    = "c:/iway/xfoc/components/iwafcont/dist";
private static String  CONFIG    = "base";
private static String  LOG_LEVEL = "FATAL";

private static String ADAPTER = "JDE";
private static String TARGET  = "JDE_connection";

// Input Message
private static String msg_run = "<JDE/>";

  public static void main(String[] args) throws Exception {

// 1. Getting the Connection factory through JNDI lookup
// ---------------------------------------------------------
  InitialContext context = new InitialContext();
  ConnectionFactory cf = (ConnectionFactory)context.lookup(iwayJndi)
  // 2. Getting a connection for a particular adapter target, in this case JDE
// ---------------------------------------------------------
  IWAFConnectionSpec cs = new IWAFConnectionSpec();
  cs.setAdapterName(ADAPTER);
  cs.setConfig(TARGET);
  cs.setLogLevel(LOG_LEVEL);  // Adapter layer log level
  Connection c = cf.getConnection(cs);// where cf is the connection factory

  // 3. Create interaction with interactionSpec for RUNTIME
// ---------------------------------------------------------
  Interaction i = c.createInteraction();
  IWAFInteractionSpec is = new IWAFInteractionSpec();
  is.setFunctionName("PROCESS");

  // 4. Create input Record and execute interaction
// ---------------------------------------------------------

  // 4.1 Using JCA standard Indexed Record
// Use JCA IndexRecord, named "input" for runtime processing.
IndexedRecord rIn = cf.getRecordFactory().createIndexedRecord("input");
rIn.add(msg_run);
  IndexedRecord rOut = (IndexedRecord)i.execute(is, rIn);
System.out.println((String)rOut.get(0));

  // 4.2 Our own Record is supported here
//IWAFRecord rIn = new IWAFRecord("input");
//rIn.setRootXML(msg_run);
//IWAFRecord response = executeRunInteraction(c, rIn);
  //IWAFRecord rOut = (IWAFRecord)i.execute(is, rIn);
//System.out.println(rOut.getRootXML());

  } // main()

}
```

4 Integration with Oracle BPEL Process Manager

OracleAS Adapter for J.D. Edwards OneWorld integrates seamlessly with Business Process Execution Language (BPEL) Process Manager to facilitate Web service integration. Oracle BPEL Process Manager is based on the Service-Oriented Architecture (SOA). It consumes adapter services exposed as Web Service Definition Language (WSDL) documents.

This chapter includes the following topics:

- Overview of Adapter Integration with Oracle BPEL Process Manager
- Deployment of Adapter
- Design Time
- Invoking Adapter Request-Response Service from Oracle BPEL Process Manager
- Listening to Adapter Events Inside Oracle BPEL Process Manager

Overview of Adapter Integration with Oracle BPEL Process Manager

To integrate with Oracle BPEL Process Manager, OracleAS Adapter for J.D. Edwards OneWorld must be deployed in the same OC4J container as Oracle BPEL Process Manager. The underlying adapter services must be exposed as WSDL files, which are generated during design time in Oracle Application Server Adapter Application Explorer (Application Explorer) for both request-response (outbound) and event notification (inbound) services of the adapter. See "Generating WSDL (J2CA Configurations Only)" on page 2-15 for more information.

The generated WSDL files are used to design the appropriate BPEL processes for inbound or outbound adapter services. A completed BPEL process must be successfully compiled in a BPEL designer and deployed to a BPEL server. Upon deployment to the BPEL server, every newly built process is automatically deployed to the Oracle BPEL Console, where you run, monitor, and administer BPEL processes, as well as listen to adapter events.

When using the adapter with Oracle BPEL Process Manager installed on OracleAS Middle Tier, your middle-tier BPEL PM home directory is OC4J_SOA, located as follows:

```
OracleAS_home\j2ee\OC4J_SOA
```

See Also:

- *Oracle Application Server Adapter Concepts*
- *Oracle BPEL Process Manager Developer's Guide*

Deployment of Adapter

During installation, OracleAS Adapter for J.D. Edwards OneWorld is deployed as a J2CA 1.0 resource adapter within the OC4J J2CA container. The adapter must be deployed in the same OC4J container as Oracle BPEL Process Manager.

See Also: *Oracle Application Server Adapter Concepts*

Design Time

The following tools are required to complete your adapter design-time configuration:

- OracleAS Adapter Application Explorer (Application Explorer)
- Oracle JDeveloper BPEL Designer (JDeveloper) or Eclipse

Note: The examples in this chapter demonstrate the use of JDeveloper.

Before you design a BPEL process, you must generate WSDL using Application Explorer. See "Generating WSDL (J2CA Configurations Only)" on page 2-15 for more information. The WSDL generated in Application Explorer is used in the BPEL process as Partner Links.

Namespace Requirements

The purpose of an XML namespace is to allow the deployment of XML vocabularies (where element and attribute names are defined) in a global environment and to reduce the risk of name collisions in a given document when vocabularies are combined. Qualified namespaces are used for stricter schema validation. In documents conforming to this specification, element and attribute names appear as qualified names. Syntactically, they are either prefixed names or unprefixed names. An attribute-based declaration syntax is provided to bind prefixes to namespace names and to bind a default namespace that applies to unprefixed element names. These declarations are scoped by the elements on which they appear so that different bindings may apply in different parts of a document. Processors conforming to this specification must recognize and act on these declarations and prefixes.

In the 10.1.3.1.0 SOA release, the recommendations for BPEL integrations is to perform stricter name space validations. As a result, Application Explorer generates Web services for the backend with the namespace marked as "Qualified". This means that during testing or usage phases of this service by BPEL, the request XML document that is used should adhere to the schema and WSDL document. Once again, it is important to remember that the namespaces are qualified. To further understand this point, the difference is illustrated with the following example:

1. Input XML for BPEL based on unqualified namespaces:

   ```
   <?xml version='1.0' encoding='utf-8' ?>
   <jdeRequest type="callmethod" user="PSFT" pwd="PSFT" environment="DV810"
   session="">
   <callMethod name="GetEffectiveAddress">
   ```

```
    <params>
      <param name="mnAddressNumber">1001</param>
    </params>
  </callMethod>
  </jdeRequest>
```

2. Input XML for BPEL based on qualified namespaces:

```
<jdeRequest
xmlns="urn:iwaysoftware:jde/services/CALLBSFN/AddressBook/GetEffectiveAddress"
type="callmethod" session="" sessionidle="">
<callMethod runOnError="" trans="String" name="GetEffectiveAddress"
app="string" returnNullData="yes">
  <params>
    <mnAddressNumber>1001</mnAddressNumber>
  </params>
  <onError abort="Yes"/>
 </callMethod>
</jdeRequest>
```

Note: If you are passing an unqualified input against a WSDL document that is expecting qualified namespaces, BPEL will throw the exception as “Unable to process input xml...."

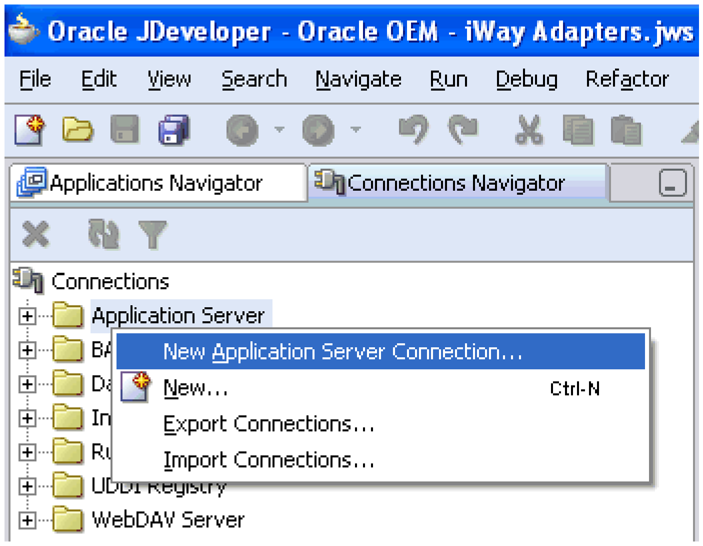
Design a BPEL Process for Request-Response Service (Outbound)

An outbound BPEL process consists of PartnerLink, Invoke, and Assign process activities. You must first create a new Application Server connection, Integration Server connection, and a synchronous BPEL process template.

Create a New Application Server Connection

To create a new Application Server connection:

1. Display the connections by clicking the **Connections Navigator** tab at the top of the upper left pane in JDeveloper.

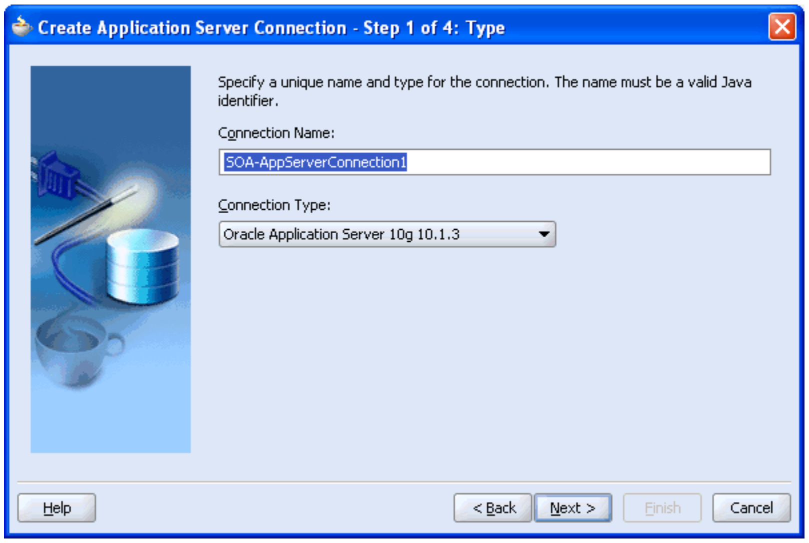


2. Right-click **Application Server** and select **New Application Server Connection**.

   The Create Application Server Connection - Welcome dialog box is displayed.

3. Click **Next**.

The Create Application Server Connection - Step 1 of 4: Type dialog box is displayed.


Create Application Server Connection - Step 1 of 4: Type

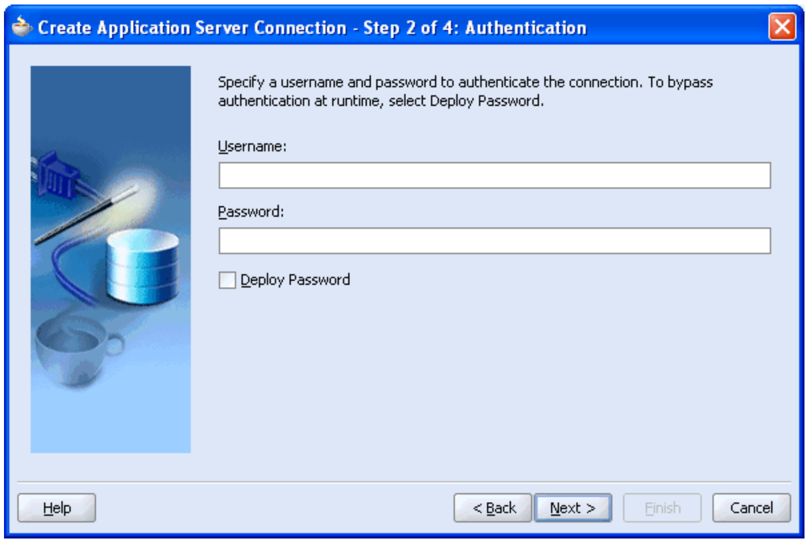
Specify a unique name and type for the connection. The name must be a valid Java identifier.

Connection Name: SOA-AppServerConnection1

Connection Type: Oracle Application Server 10g 10.1.3

4. Specify a unique name and select a connection type for your Application Server connection and click **Next**.

   The Create Application Server Connection - Step 2 of 4: Authentication dialog box is displayed.


Create Application Server Connection - Step 2 of 4: Authentication

Specify a username and password to authenticate the connection. To bypass authentication at runtime, select Deploy Password.

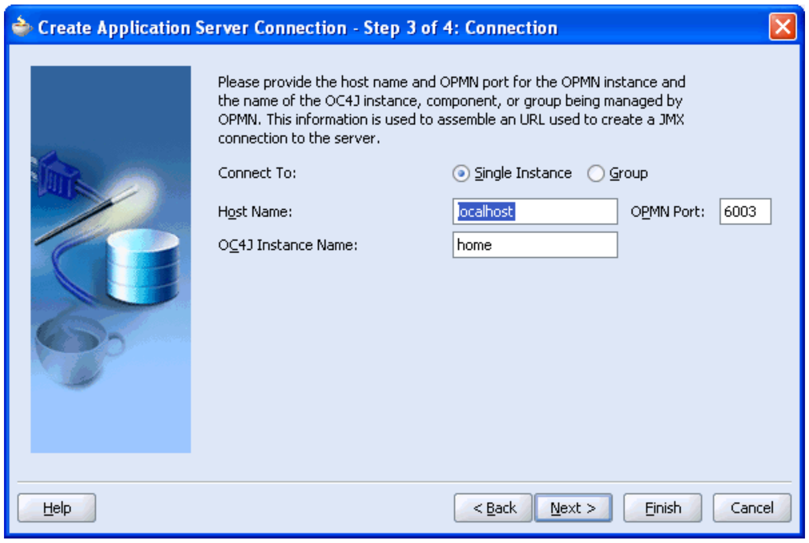
Username:

Password:

Deploy Password

5. Specify a valid user name and password for the Application Server you wish to connect to.
6. Select **Deploy Password**.
7. Click **Next**.

   The Create Application Server Connection - Step 3 of 4: Connection dialog box is displayed.


Create Application Server Connection - Step 3 of 4: Connection

Please provide the host name and OPMN port for the OPMN instance and the name of the OC4J instance, component, or group being managed by OPMN. This information is used to assemble an URL used to create a JMX connection to the server.

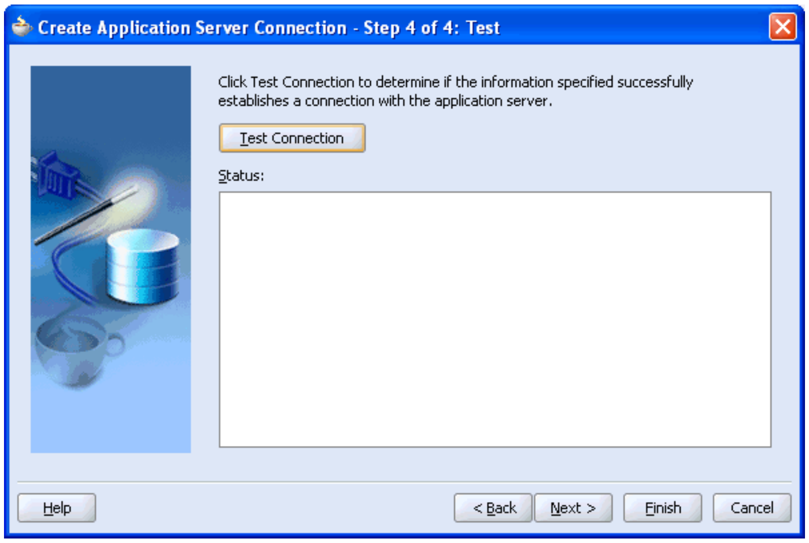
Connect To: [✓ Single Instance] Group

Host Name: localhost OPMN Port: 6003

OC4J Instance Name: home

8. Select the **Single Instance** connection option.
9. Enter **localhost** as the host name and **6003** for the OPMN port.
10. Enter **home** as the OC4J instance name
11. Click **Next**.

    The Create Application Server Connection - Step 4 of 4: Test dialog box is displayed

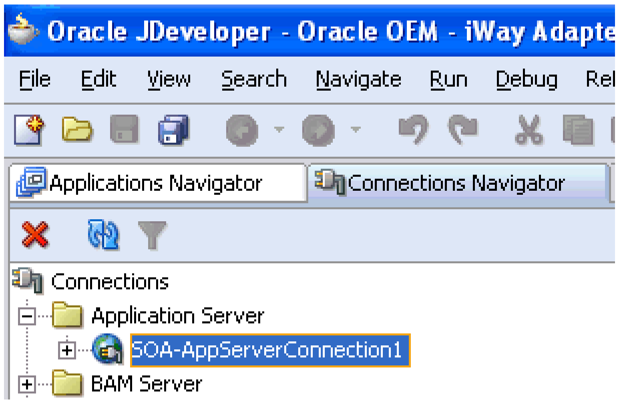

Create Application Server Connection - Step 4 of 4: Test

Click Test Connection to determine if the information specified successfully establishes a connection with the application server.

12. Click **Test Connection**.

    When the test is complete and the connection is successful, a **Success!** message appears in the status area.
13. Click **Finish**.

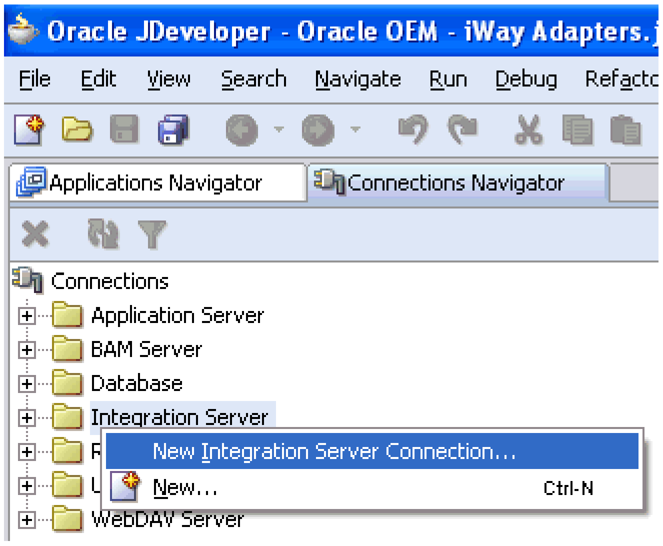
    Your newly created Application Server connection is displayed in the Connections Navigator tab under the Application Server node.



Create a New Integration Server Connection

To create a new Integration Server connection:

1. Display the connections by clicking the **Connections Navigator** tab at the top of the upper left pane in JDeveloper.

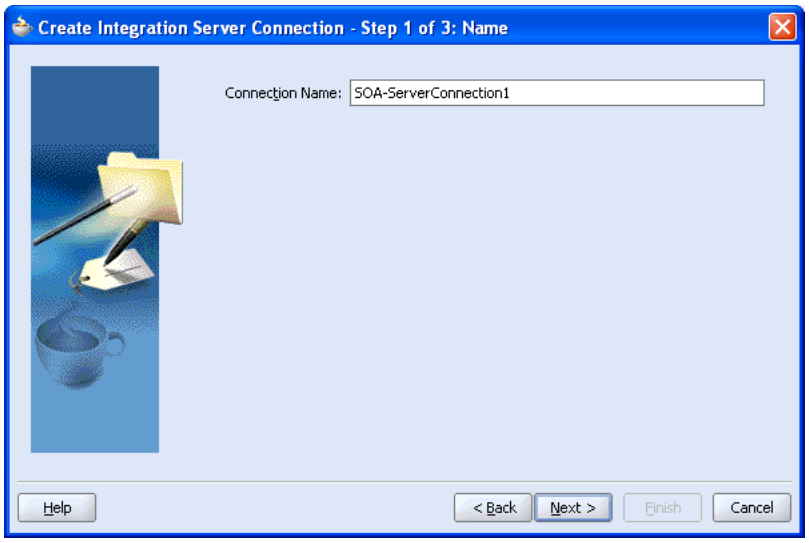


2. Right-click **Integration Server** and select **New Integration Server Connection**.

   The Create Integration Server Connection - Welcome dialog box is displayed.

3. Click **Next**.

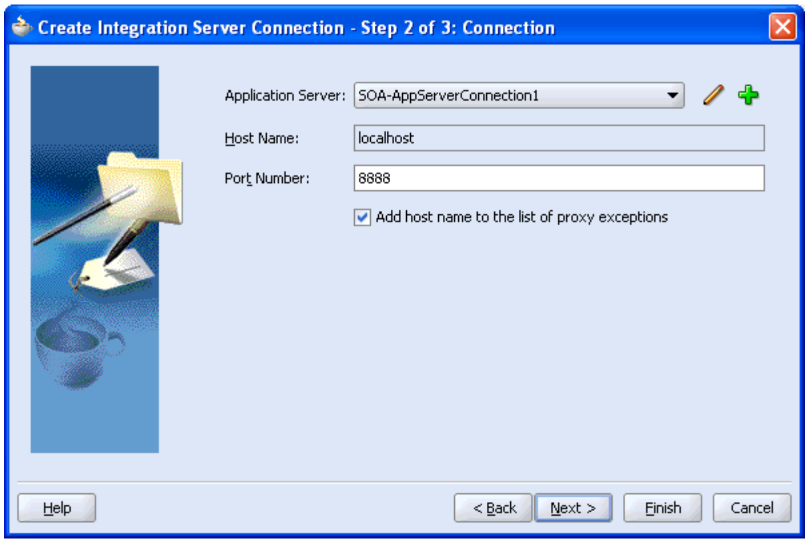
The Create Integration Server Connection - Step 1 of 3: Name dialog box is displayed.


Create Integration Server Connection - Step 1 of 3: Name

Connection Name: SOA-ServerConnection1

4. Specify a unique name and click **Next**.

   The Create Integration Server Connection - Step 2 of 3: Connection dialog box is displayed.


Create Integration Server Connection - Step 2 of 3: Connection

Application Server: SOA-AppServerConnection1

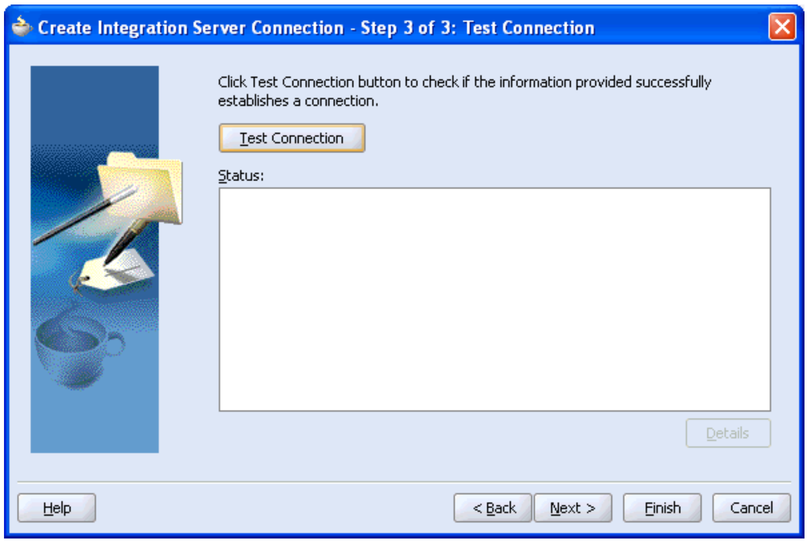
Host Name: localhost

Port Number: 8888

[✓] Add host name to the list of proxy exceptions

5. Select an Application Server connection, which is already created.
6. Enter **localhost** as the host name and **8888** for the port number.
7. Select **Add host name to the list of proxy exceptions** and click **Next**.

   The Create Integration Server Connection - Step 3 of 3: Test Connection dialog box is displayed.

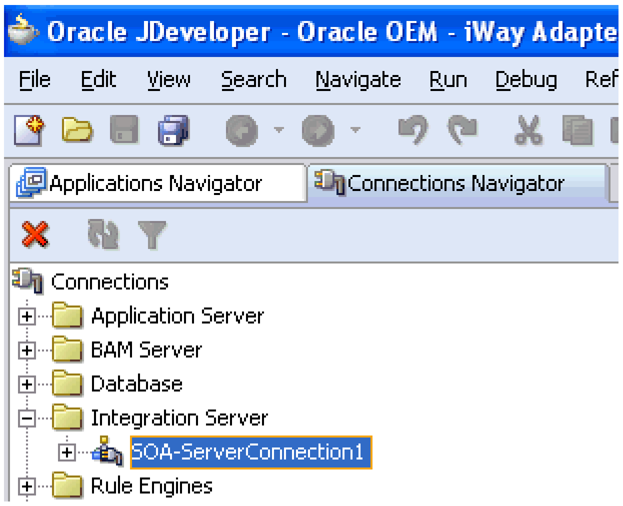

Create Integration Server Connection - Step 3 of 3: Test Connection
Click Test Connection button to check if the information provided successfully establishes a connection.
Test Connection
Status:
Details
Help
< Back
Next >
Finish
Cancel

8. Click **Test Connection**.

   When the test is complete and the connection is successful, a **Success!** message appears in the status area.

9. Click **Finish**.

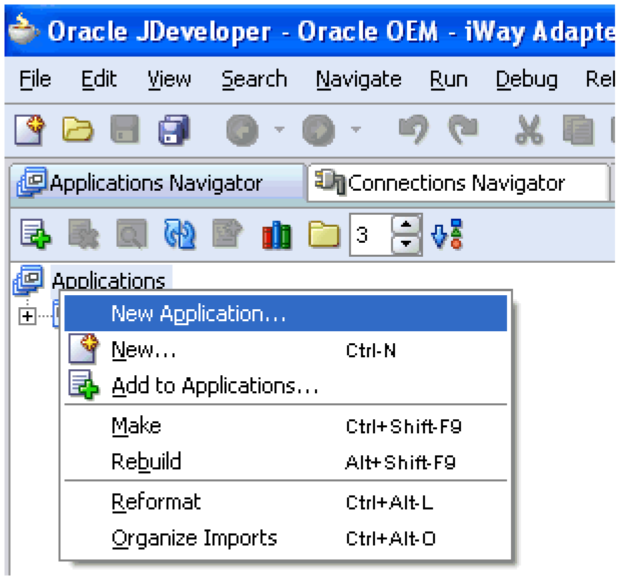
   Your newly created Integration Server connection is displayed in the Connections Navigator tab under the Integration Server node.



Create a New BPEL Project for Outbound Interaction (Synchronous Process)

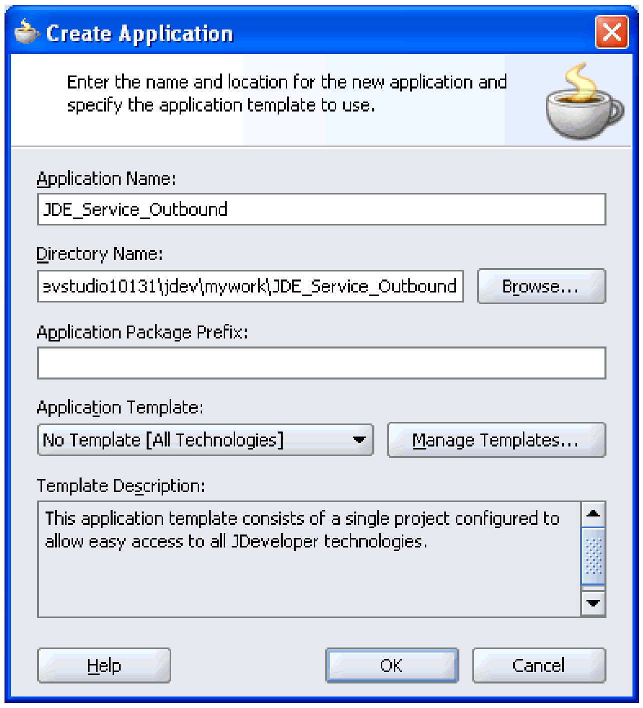
To create a new BPEL project for a synchronous process:

1. At the top of the upper left pane, click the **Applications Navigator** tab.



2. Right-click the Applications node and select **New Application**.

   The Create Application dialog box is displayed.


Create Application

Enter the name and location for the new application and specify the application template to use.

Application Name: JDE_Service_Outbound

Directory Name: evstudio10131\jdev\mywork\JDE_Service_Outbound

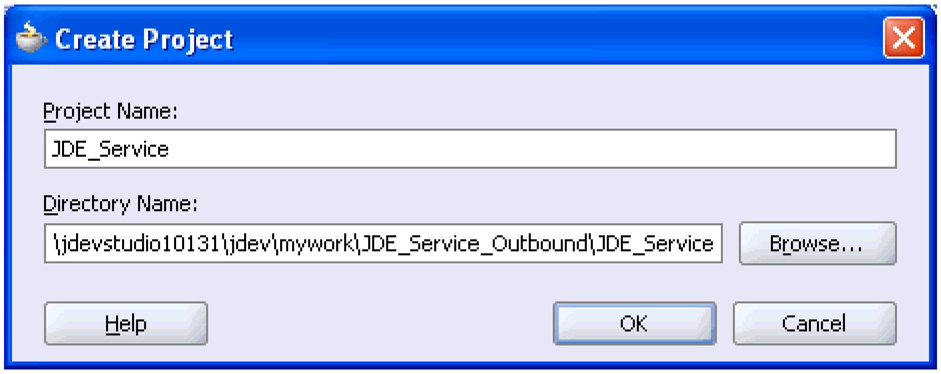
Application Package Prefix:

Application Template: No Template [All Technologies]

Template Description: This application template consists of a single project configured to allow easy access to all JDeveloper technologies.

3. Enter a unique name for your application and click **OK**.

   The Create Project dialog box is displayed.

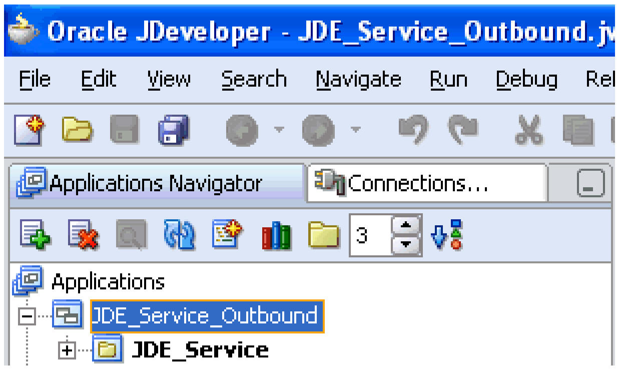


Project Name: JDE_Service

Directory Name: \jdevstudio10131\jdev\mywork\JDE_Service_Outbound\JDE_Service

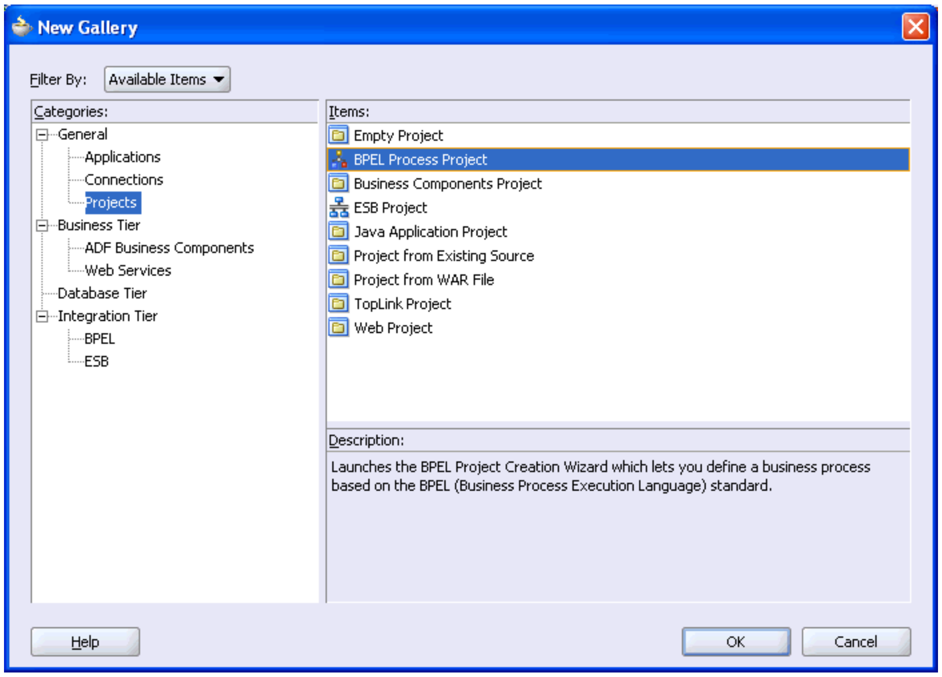
4. Enter a unique name for your project and click **OK**.

   Your new application is displayed in the Applications Navigator tab under the Applications node.



5. Right-click the application node you created and select **New Project**.

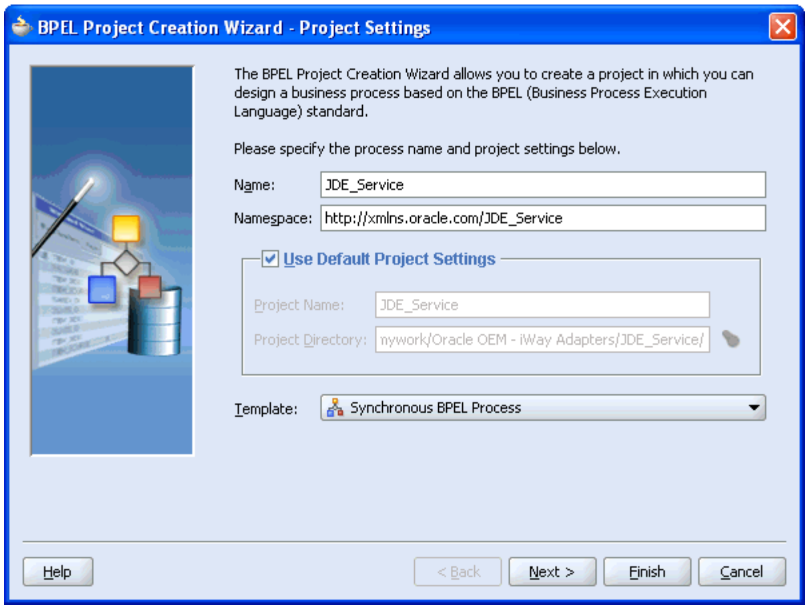
   The New Gallery window is displayed.



Description: Launches the BPEL Project Creation Wizard which lets you define a business process based on the BPEL (Business Process Execution Language) standard.

6. From the Items list, select **BPEL Process Project** and click **OK**.

The BPEL Project Creation Wizard - Project Settings dialog box is displayed.


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: JDE_Service

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

☑ Use Default Project Settings

Project Name: JDE_Service

Project Directory: nywork/Oracle OEM - iWay Adapters/JDE_Service/

Template: Synchronous BPEL Process

7. Perform the following steps:

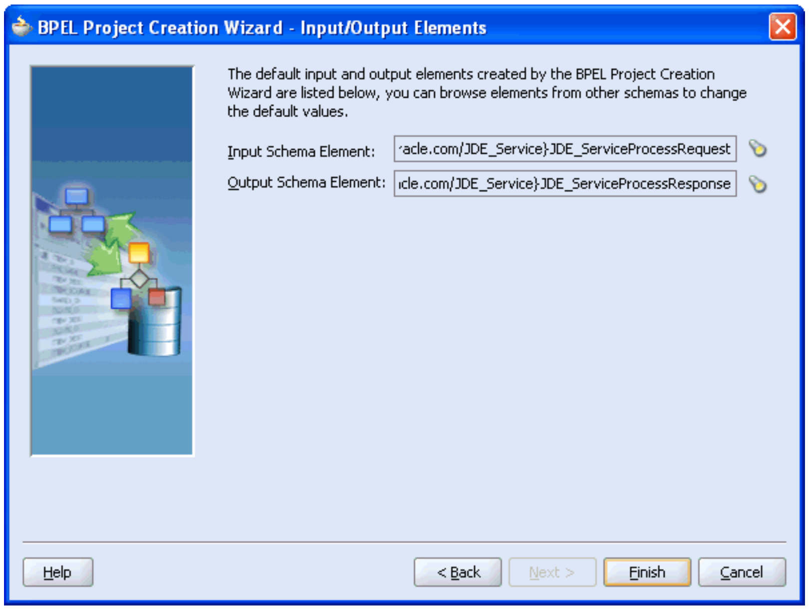
   a. Specify a name for the BPEL process.

      The Namespace field is updated automatically.

   b. From the Template list, select **Synchronous BPEL Process**.

8. Click **Next**.

   The BPEL Project Creation Wizard - Input/Output Elements dialog box is displayed.


BPEL Project Creation Wizard - Input/Output Elements

The default input and output elements created by the BPEL Project Creation Wizard are listed below, you can browse elements from other schemas to change the default values.

Input Schema Element: ·acle.com/JDE_Service}JDE_ServiceProcessRequest

Output Schema Element: ıcle.com/JDE_Service}JDE_ServiceProcessResponse

9. Review the input/output schema elements that are created by the BPEL Project Creation Wizard and click **Finish**.

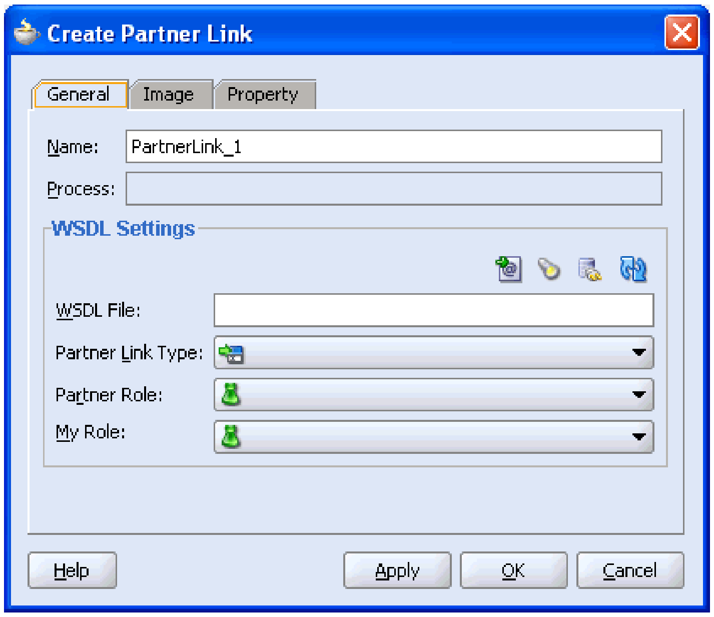
Create an Outbound PartnerLink Activity

When designing a BPEL process, a PartnerLink activity must be created to invoke the J.D. Edwards service. A PartnerLink describes a set of operations within a Web service. The WSDL document is the external contract to which the Web service conforms. Given a WSDL, any BPEL process can initiate a Web service through a PartnerLink.

To create an outbound PartnerLink using the WSDL file you generated in Application Explorer:

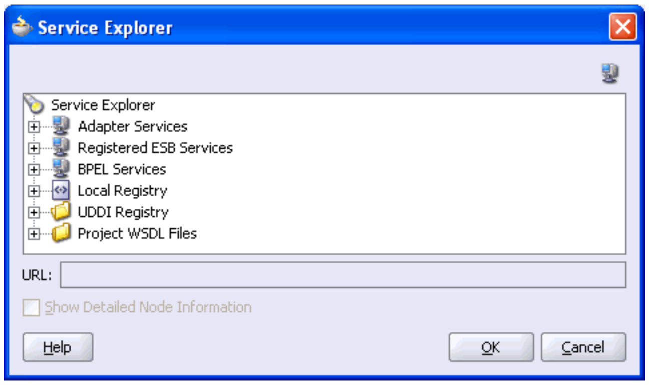
1. From the Services pane on the right, drag and drop a PartnerLink to the visual editor.

   The Create Partner Link dialog box is displayed.



2. Click the **Service Explorer** icon (second icon from the left preceding the **WSDL File** field).

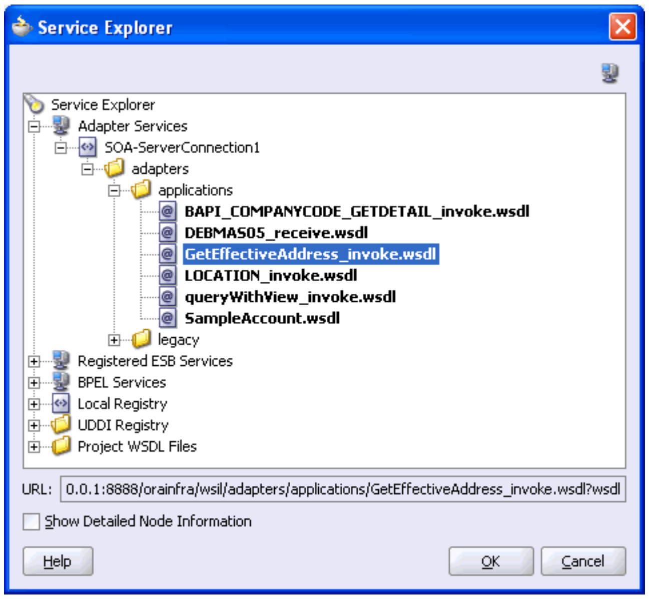
   The Service Explorer dialog box is displayed.



3. Expand your new connection under Adapter Services, followed by **adapters**, and then **applications**.

   The WSDL tree displayed in the Service Explorer dialog box lists any WSDL files you have created using Application Explorer. The WSDL tree is generated by a

WSDL servlet, which is automatically deployed as part of the BPEL Server installation.

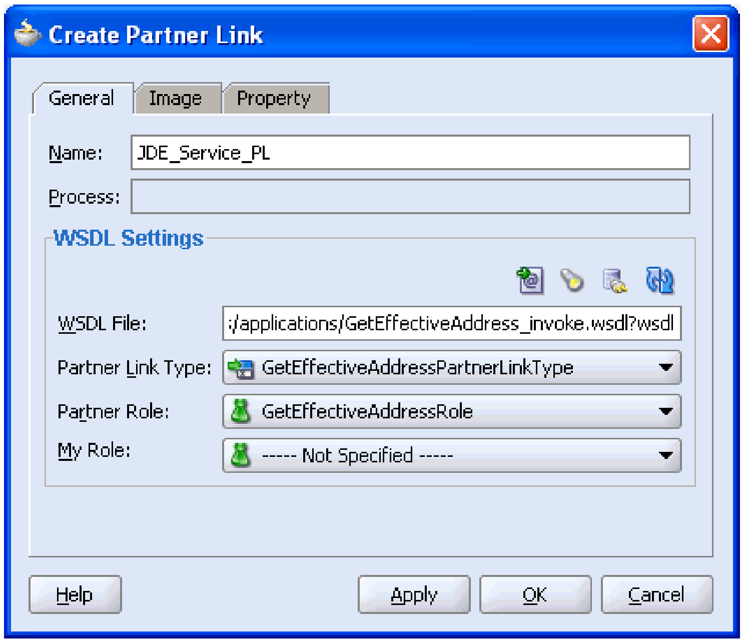


URL: 0.0.1:8888/orainfra/wsil/adapters/applications/GetEffectiveAddress_invoke.wsdl?wsdl

Note: If you have organized your WSDL files in subfolders, the WSIL browser will display the full tree structure of your WSDL hierarchy. By default, the names of all WSDL files generated for outbound adapter services end with `_invoke`.

4. Select **GetEffectiveAddress_invoke.wsdl** and click **OK**.

   The **WSDL File** field in the Create Partner Link dialog box displays the name and location of the selected WSDL file. The **Partner Link Type** field specifies the PartnerLink defined in the WSDL file.



Name: JDE_Service_PL
Process:
WSDL File: :/applications/GetEffectiveAddress_invoke.wsdl?wsdl
Partner Link Type: GetEffectiveAddressPartnerLinkType
Partner Role: GetEffectiveAddressRole
My Role: ----- Not Specified -----

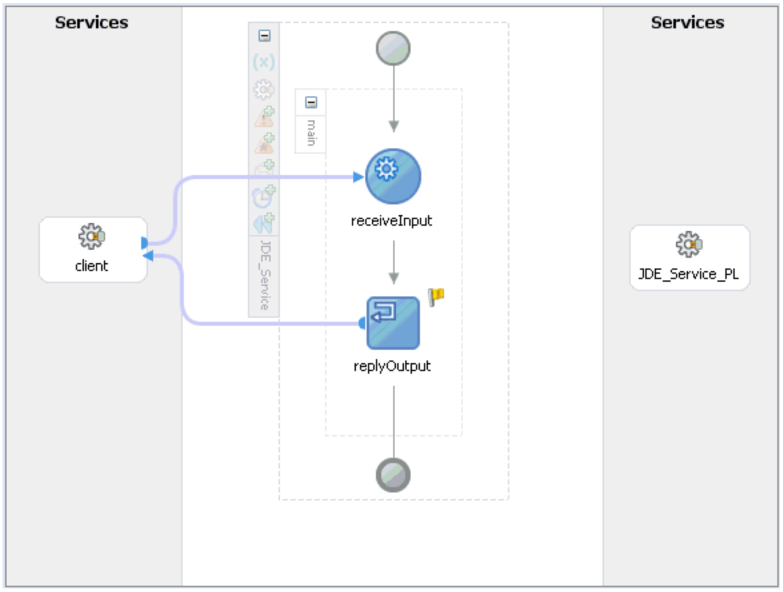
Perform the following steps:

a. Leave the **My Role** field unspecified. The role of the PartnerLink is null, as it will be synchronously invoked from the BPEL process.

b. From the **Partner Role** list, select the default value **GetEffectiveAddressRole**. This is the role of the BPEL process.

5. Click **OK**.

   The new PartnerLink appears in the visual editor.



6. Select **Save** from the **File** menu.

Create an Outbound Invoke Activity

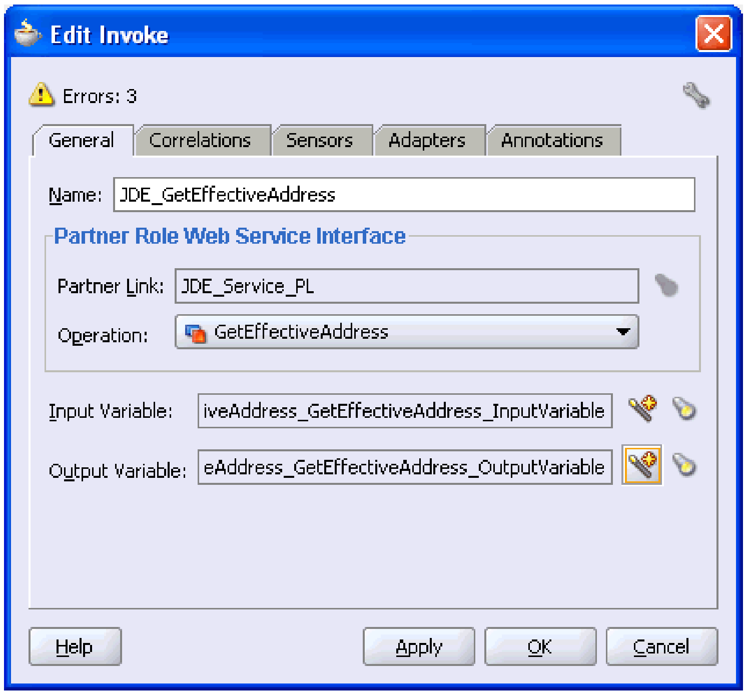
This activity enables you to specify an operation you want to invoke for the service identified by its PartnerLink. The Invoke activity opens a port in the process that is used to send and receive data. It uses this port to submit required data and receive a response. For synchronous callbacks, only one port is needed for both the send and the receive functions.

To create an outbound Invoke activity:

1. From the **Process Activities** pane on the right, drag an **Invoke** activity to the visual editor and place it between the Receive activity (`receiveInput`) and the Reply activity (`replyOutput`).

2. Extend a connection between the Invoke activity and your newly-created PartnerLink.

The Edit Invoke dialog box is displayed. Note that the Partner Link and Operation fields are automatically populated.



Name: JDE_GetEffectiveAddress

Partner Role Web Service Interface

Partner Link: JDE_Service_PL

Operation: GetEffectiveAddress

Input Variable: iveAddress_GetEffectiveAddress_InputVariable

Output Variable: eAddress_GetEffectiveAddress_OutputVariable

Note: Ignore any invalid settings and error warnings.

Perform the following steps:

a. In the **Name** field, provide a meaningful name for the Invoke activity.

b. Click the first icon to the right of the **Input Variable** field, then click **OK** in the Create Variable window that is displayed.

A global variable is automatically created in the Input Variable field.

c. Click the first icon to the right of the **Output Variable** field, then click **OK** in the Create Variable window that is displayed.

A global variable is automatically created in the Output Variable field.

d. Click **Apply**.

The Edit Invoke window should no longer display any warnings or errors.

3. Click **OK**.
4. Select **Save** from the **File** menu.

Create an Assign Activity

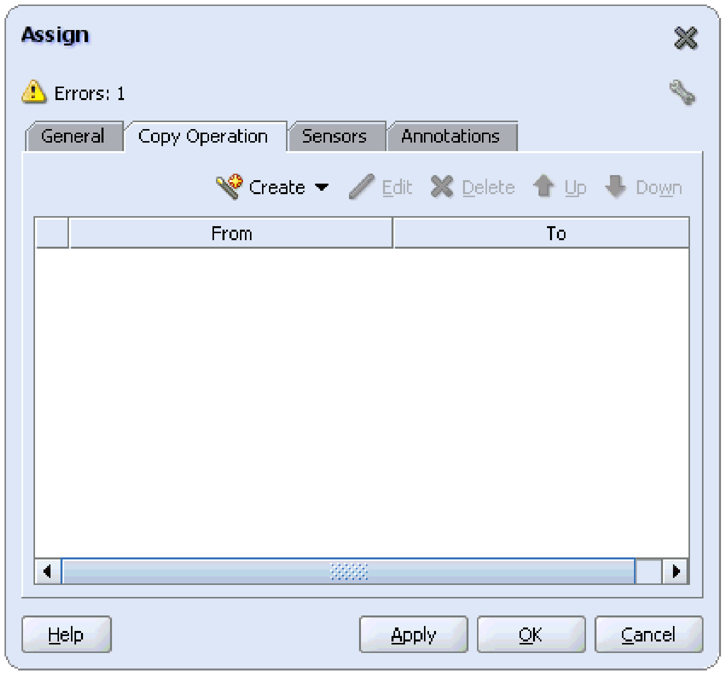
An Assign activity provides a method for simple data manipulation, such as copying the contents of one variable to another. This Assign activity maps the input variable of the J.D. Edwards process to the J.D. Edwards PartnerLink input.

To create an Assign activity:

1. From the **Process Activities** pane on the right, drag an **Assign** activity to the visual editor and place it between `receiveInput` and `JDE_GetEffectiveAddress`.

2. Double-click the **Assign** activity icon.

   The Assign dialog box is displayed.



Note: Ignore any invalid settings and error warnings.

---

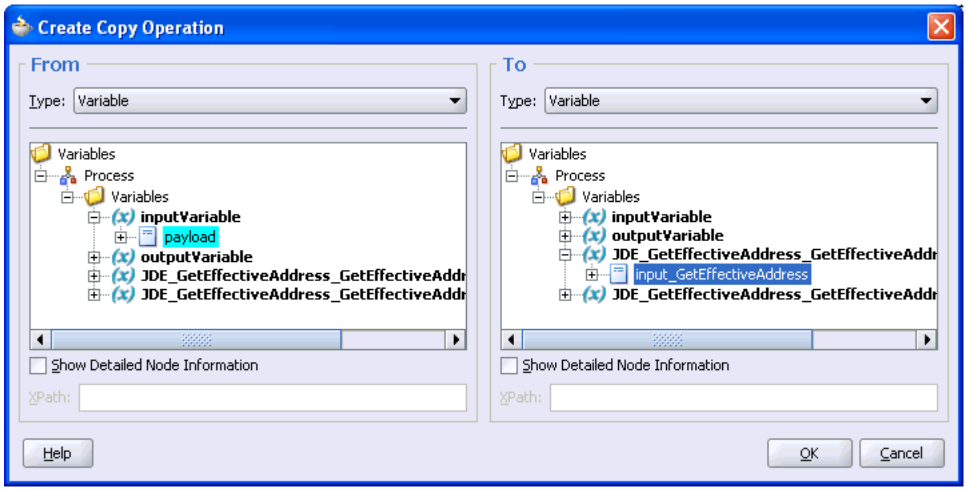
3. In the Copy Operation tab, click **Create** and select **Copy Operation**.

   The Create Copy Operation dialog box is displayed.

   a. In the **From** pane, expand **Variables**, then **inputVariable**, and then highlight **payload**.

   b. In the **To** pane, expand **Variables**, then **JDE_GetEffectiveAddress_GetEffectiveAddress_InputVariable**, and then highlight **input_GetEffectiveAddress**.

   Your Create Copy Operation dialog box should look as follows:


Create Copy Operation
From
Type: Variable
Variables
Process
Variables
inputVariable
payload
outputVariable
JDE_GetEffectiveAddress_GetEffectiveAddr
JDE_GetEffectiveAddress_GetEffectiveAddr
Show Detailed Node Information
XPath:
To
Type: Variable
Variables
Process
Variables
inputVariable
outputVariable
JDE_GetEffectiveAddress_GetEffectiveAddr
input_GetEffectiveAddress
JDE_GetEffectiveAddress_GetEffectiveAddr
Show Detailed Node Information
XPath:
Help
OK
Cancel

4. To close the Create Copy Operation dialog box and the Assign dialog box, click **OK**.

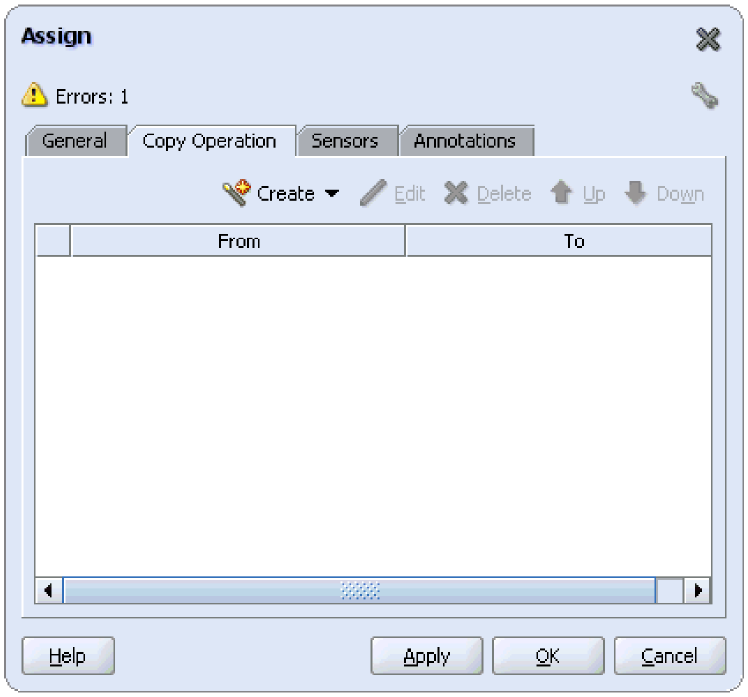
Create a Second Assign Activity

This Assign activity maps the output variable of the J.D. Edwards process to the J.D. Edwards PartnerLink output.

To create a second Assign activity:

1. From the **Process Activities** pane on the right, drag another **Assign** activity to the visual editor and place it between the Invoke activity (`JDE_GetEffectiveAddress`) and the Reply activity (`replyOutput`).
2. Double-click the **Assign** activity icon.

   The Assign settings dialog box is displayed.



Note: Ignore any invalid settings and error warnings.

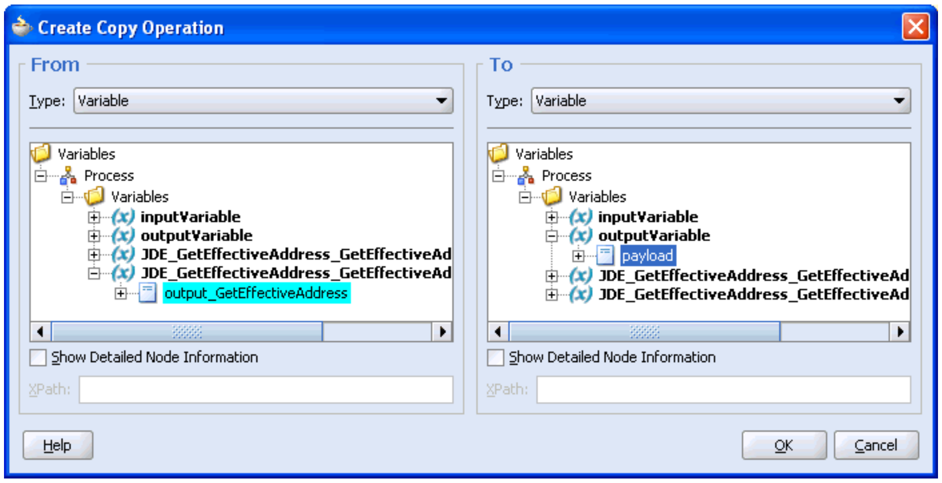
3. In the Copy Operation tab, click **Create** and select **Copy Operation**.

   The Create Copy Operation dialog box is displayed. Perform the following steps:

   a. In the **From** pane, expand **Variables**, then **JDE_GetEffectiveAddress_GetEffectiveAddress_OutputVariable**, and then highlight **output_GetEffectiveAddress**.

   b. In the **To** pane, expand **Variables**, then **outputVariable**, and then highlight **payload**.

   Your Create Copy Operation dialog box should look as follows:

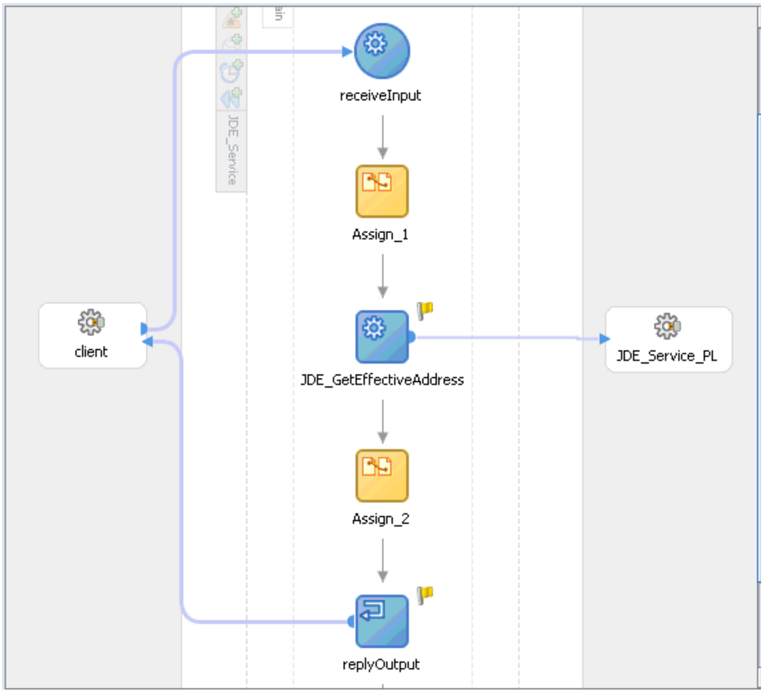

Create Copy Operation

From
Type: Variable
Variables
Process
Variables
(x) inputVariable
(x) outputVariable
(x) JDE_GetEffectiveAddress_GetEffectiveAd
(x) JDE_GetEffectiveAddress_GetEffectiveAd
output_GetEffectiveAddress
Show Detailed Node Information
XPath:

To
Type: Variable
Variables
Process
Variables
(x) inputVariable
(x) outputVariable
payload
(x) JDE_GetEffectiveAddress_GetEffectiveAd
(x) JDE_GetEffectiveAddress_GetEffectiveAd
Show Detailed Node Information
XPath:

4. To close the Create Copy Operation dialog box and the Assign dialog box, click **OK**.
5. Select **Save** from the **File** menu.

The following image shows the diagram view of your completed BPEL process.



See "Invoking Adapter Request-Response Service from Oracle BPEL Process Manager" on page 4-24 for information on how to deploy and manage your outbound process.

See Also:

- *Oracle BPEL Process Manager Developer's Guide*
- *Oracle Application Server Adapter Concepts*

Testing Outbound BPEL and ESB Processes

The BPEL console allows you to test deployed BPEL processes. Once a process is deployed, you can manage, monitor, and run an end-to-end scenario using the Initiate tab in the console. The OracleAS Adapter for J.D. Edwards OneWorld is certified for testing using the **XML Payload** option and the option of running using **Through Java Delivery API**. It is recommended that developers use this method for testing the OracleAS Adapter for J.D. Edwards OneWorld.

When testing an outbound BPEL process from the BPEL console or an outbound ESB process from the Enterprise Manager (EM) console, do not use the XML envelopes that are generated by these consoles. Instead, remove them and use the XML payloads that are generated from the schemas, which conform to the WSDLs for namespace qualifications.

The ESB data flows can be tested using the EM console. When creating an ESB data flow and interactions, the Web services are created and registered with the Oracle Application Server. For more information on creating an ESB outbound process, see Chapter 6, "ESB Integration Examples".

Design a BPEL Process for Event Handling (Inbound)

An inbound BPEL process consists of a PartnerLink and a Receive process activity. You must first create a channel and a new BPEL Process Manager Server connection. See Chapter 5, "BPEL Process Manager Integration Examples" for instructions on how to perform these procedures.

Note: You must create a separate channel for every event and select that channel when you generate WSDL for inbound interaction using Application Explorer. Do not start the channel in Application Explorer, as Oracle BPEL Process Manager manages endpoint activation independently. See "J.D. Edwards OneWorld Event Integration" on page 5-13 for more information.

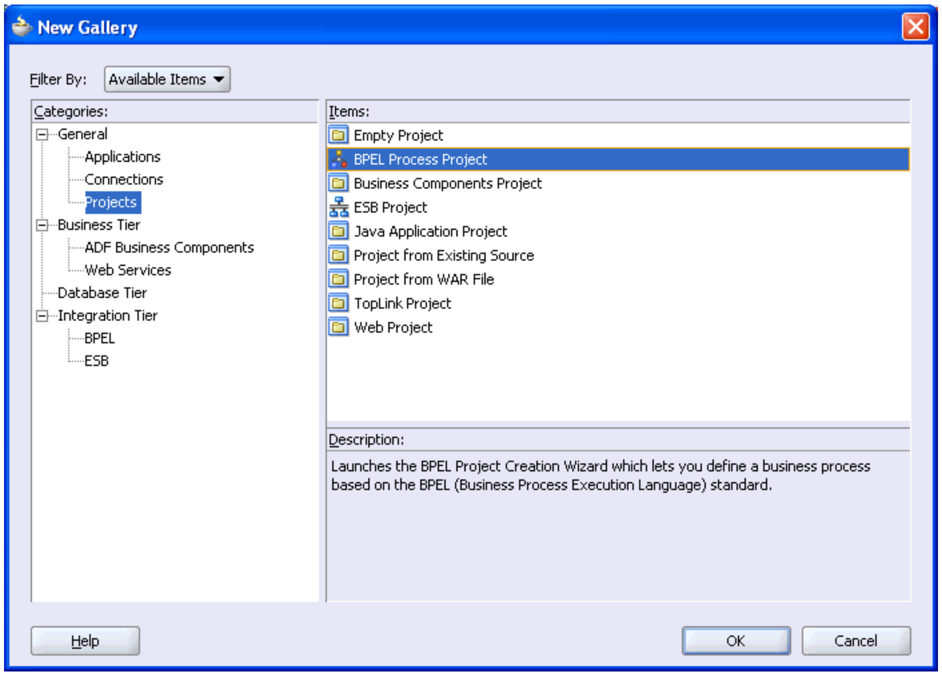
Create a New BPEL Project for Inbound Interaction (Empty Process)

Before you create a BPEL project, verify that your BPEL Server is running. After you have created a new server connection, you are ready to design an empty process template for your BPEL project.

To create a new BPEL project for inbound interaction:

1. Click the **Applications Navigator** tab and select an application for your project.
2. Right-click the application and select **New Project**.

The New Gallery window is displayed.

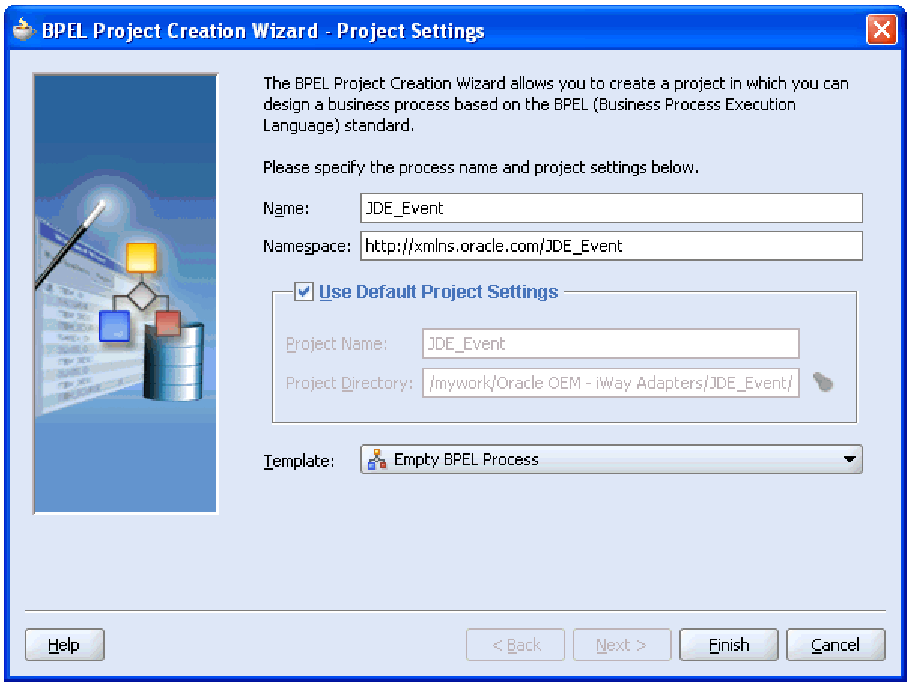


Description:

Launches the BPEL Project Creation Wizard which lets you define a business process based on the BPEL (Business Process Execution Language) standard.

3. From the Items list, select **BPEL Process Project** and click **OK**.

   The BPEL Project Creation Wizard - Project Settings dialog box is displayed.



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: JDE_Event

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

☑ Use Default Project Settings

Project Name: JDE_Event

Project Directory: /mywork/Oracle OEM - iWay Adapters/JDE_Event/

Template: Empty BPEL Process

4. Perform the following steps:

   a. Specify a name for the process.

      The Namespace field is updated automatically.

   b. From the Template list, select **Empty BPEL Process**.

c. Click **Finish**.

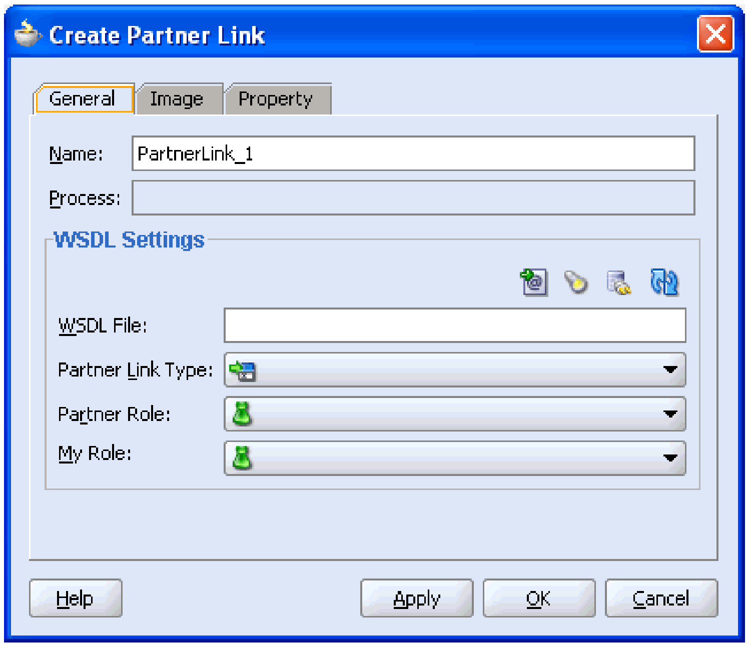
Create an Inbound PartnerLink Activity

When designing a BPEL process, a PartnerLink activity must be created to invoke the J.D. Edwards service. A PartnerLink describes a set of operations within a Web service. The WSDL document is the external contract to which the Web service conforms. Given a WSDL, any BPEL process can initiate a Web service through a PartnerLink.

To create an inbound PartnerLink using the WSDL file you generated in Application Explorer:

1. From the Process Activities pane on the right, drag and drop a PartnerLink to the visual editor.

   The Create Partner Link dialog box is displayed.



Create Partner Link

General | Image | Property

Name: PartnerLink_1

Process:

WSDL Settings

WSDL File:

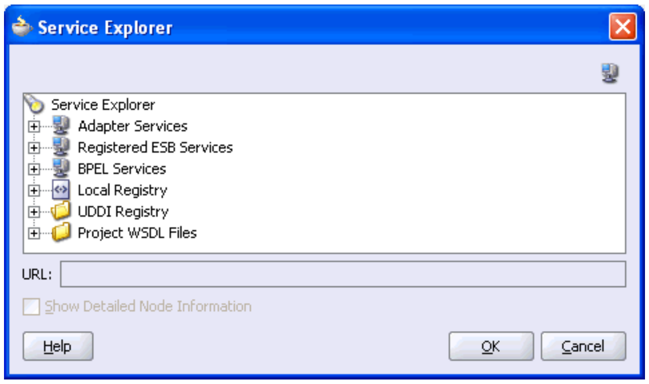
Partner Link Type:

Partner Role:

My Role:

2. Click the **Service Explorer** icon (second icon from the left preceding the **WSDL File** field).

   The Service Explorer dialog box is displayed.



Service Explorer

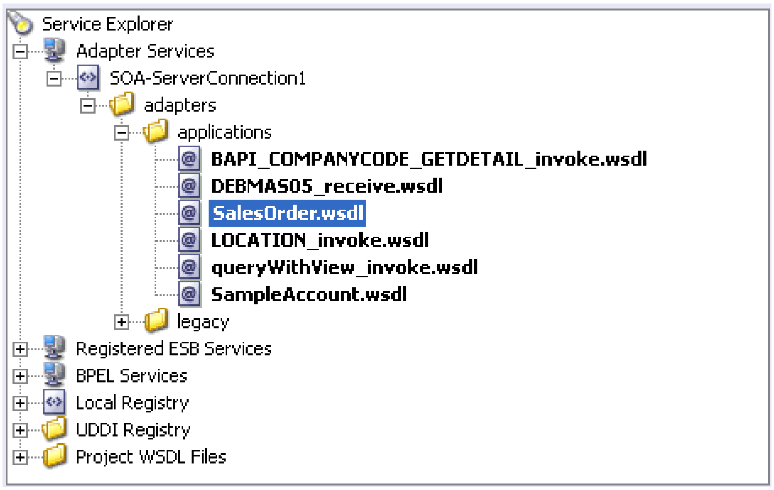
- Service Explorer
  - Adapter Services
  - Registered ESB Services
  - BPEL Services
  - Local Registry
  - UDDI Registry
  - Project WSDL Files

URL:

Show Detailed Node Information

3. Expand your new connection under Adapter Services, followed by **adapters**, and then **applications**.

The WSDL tree displays the WSDL files you created using Application Explorer. The WSDL tree is generated by a WSDL servlet, which is automatically deployed as part of the BPEL Server installation.



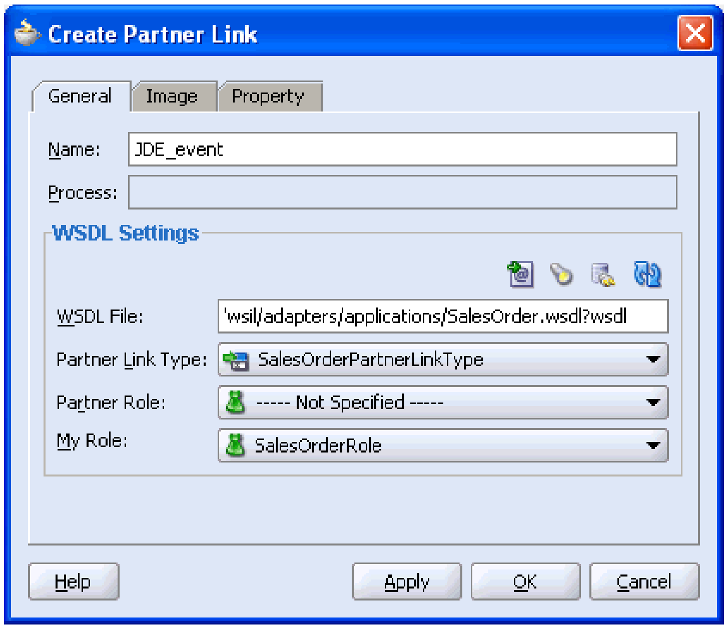
Service Explorer
- Adapter Services
  - SOA-ServerConnection1
    - adapters
      - applications
        - BAPI_COMPANYCODE_GETDETAIL_invoke.wsdl
        - DEBMAS05_receive.wsdl
        - SalesOrder.wsdl
        - LOCATION_invoke.wsdl
        - queryWithView_invoke.wsdl
        - SampleAccount.wsdl
      - legacy
- Registered ESB Services
- BPEL Services
- Local Registry
- UDDI Registry
- Project WSDL Files

Note: If you have organized your WSDL files in subfolders, the WSIL browser will display the full tree structure of your WSDL hierarchy. The names of WSDL files generated for inbound adapter services usually end with `_receive`.

---

4. Select **SalesOrder.wsdl** and click **OK**.

The Create Partner Link dialog box is displayed.



Name: JDE_event
Process:
WSDL File: 'wsil/adapters/applications/SalesOrder.wsdl?wsdl
Partner Link Type: SalesOrderPartnerLinkType
Partner Role: ----- Not Specified -----
My Role: SalesOrderRole

The **WSDL File** field displays the name and location of the selected WSDL file. The **Partner Link Type** field specifies the PartnerLink defined in the WSDL file.

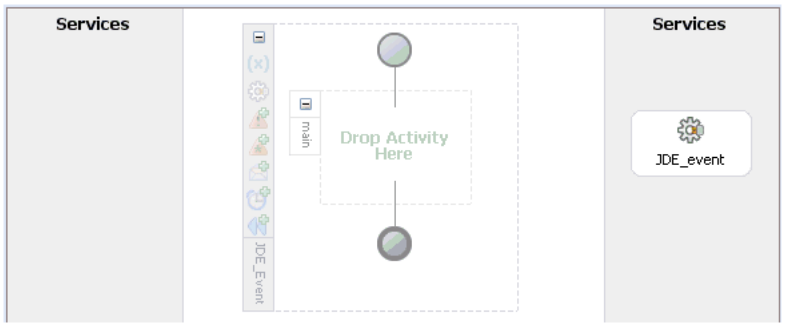
Perform the following steps:

a. From the **My Role** list, select the default value **SalesOrderRole**.

b. Leave the **Partner Role** field unspecified.

5. Click **Apply**, and then **OK**.

   The new JDE PartnerLink appears in the visual editor.



6. Select **Save** from the **File** menu.

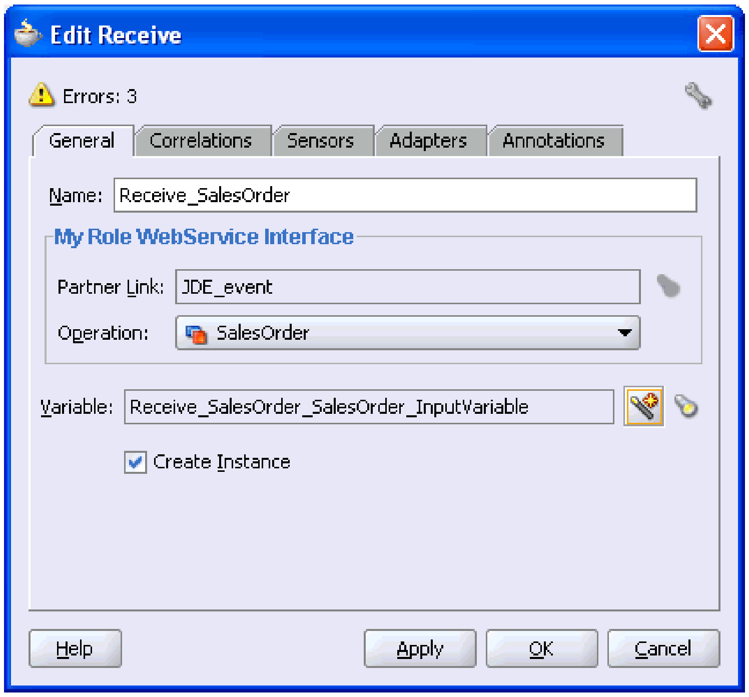
Create an Inbound Receive Activity

To create an inbound Receive Activity:

1. From the **Process Activities** pane on the right, drag a **Receive** activity to the visual editor and place it in the designated placeholder labeled **Drop Activity Here**.

2. Connect the Receive activity to the `JDE_event` PartnerLink.

   The Edit Receive dialog box is displayed.


Edit Receive

Errors: 3

General | Correlations | Sensors | Adapters | Annotations

Name: Receive_SalesOrder

My Role WebService Interface

Partner Link: JDE_event

Operation: SalesOrder

Variable: Receive_SalesOrder_SalesOrder_InputVariable

[✓] Create Instance

Note: Ignore any invalid settings and error warnings.

Perform the following steps:

a. Specify a name for the Receive Activity, for example, **Receive_SalesOrder**.

b. Click the first icon to the right of the **Variable** field, then click **OK** in the Create Variable dialog box that is displayed.

c. Verify that the **Create Instance** check box is selected.

3. Click **Apply**.

   The Receive dialog box should no longer display any warnings or errors.

4. Click **OK**.

   A connection is created between the PartnerLink and the Receive activity. You have completed the design of your inbound BPEL process.

See "Listening to Adapter Events Inside Oracle BPEL Process Manager" on page 4-27 for information on how to deploy and manage your inbound process.

See Also:

- *Oracle BPEL Process Manager Developer's Guide*
- *Oracle Application Server Adapter Concepts*

Invoking Adapter Request-Response Service from Oracle BPEL Process Manager

The OracleAS Adapter for J.D. Edwards OneWorld request-response service is used to create, delete, update, and query back-end data as well as to call back-end workflows and transactions. The following section describes how to invoke the adapter synchronous request-response service, also referred to as Outbound Interaction, as well as how to manage the process in Oracle BPEL Console.

Deploy the Outbound BPEL Process

The procedures for deploying an inbound and an outbound BPEL process using the JDeveloper interface are identical.

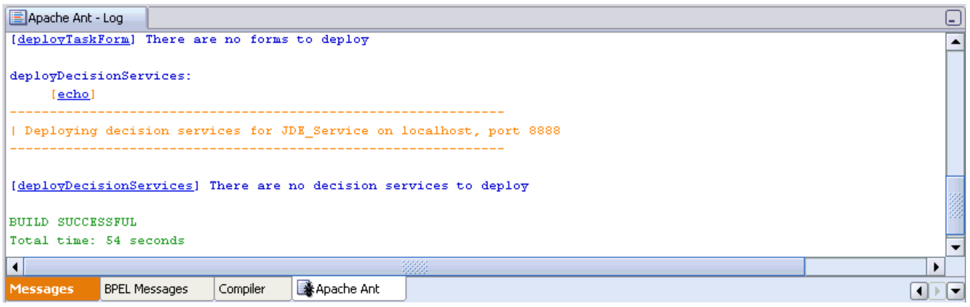
To deploy your BPEL process in JDeveloper:

1. Right-click your project in the Applications Navigator tab.
2. Select **Deploy**, then ***Your BPEL PM Server connection***, and then **Deploy to default domain**.

   The deployment process starts automatically.

3. Observe the **Messages** log at the bottom of the window.

   The Messages log displays the deployment status. In this example, it shows a successful deployment message for the outbound process.



```
[deployTaskForm] There are no forms to deploy

deployDecisionServices:
     [echo]
--------------------------------------------------------------
| Deploying decision services for JDE_Service on localhost, port 8888
--------------------------------------------------------------

[deployDecisionServices] There are no decision services to deploy

BUILD SUCCESSFUL
Total time: 54 seconds
```

If deployment was not successful, click the **Compiler** tab to view all error and warning messages generated during the deployment process.

Manage the Deployed Outbound Process in Oracle BPEL Console

JDeveloper deploys the developed process directly to the Oracle BPEL Console, which enables you to run, monitor, and administer BPEL processes.

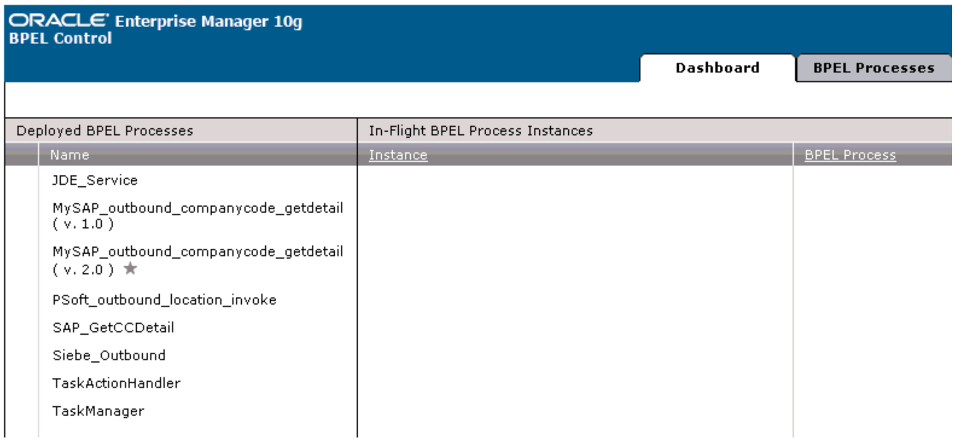
To invoke adapter request-response service:

1. Start the Oracle BPEL Console by entering the following URL in a browser:

   `http://`*`host`*`:`*`port`*`/BPELConsole`

2. Provide a valid user name and password.

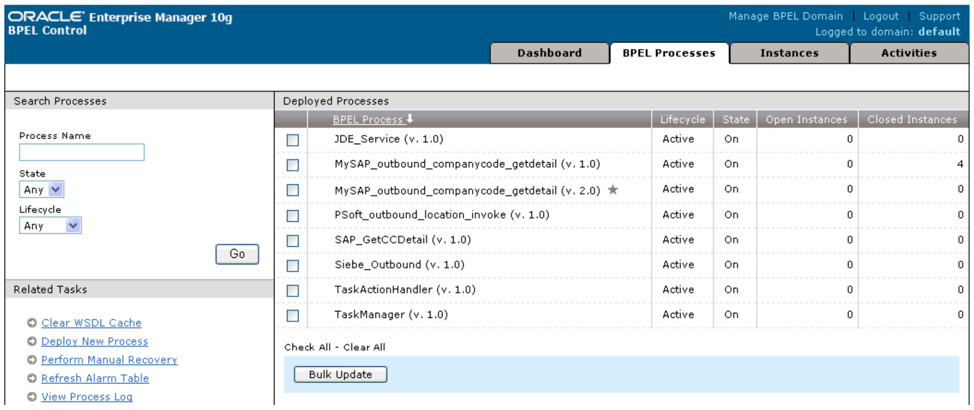
   The Oracle BPEL Console main page is displayed. All deployed BPEL processes are listed in the Dashboard tab.



Deployed BPEL Processes	In-Flight BPEL Process Instances	
Name	Instance	BPEL Process
JDE_Service		
MySAP_outbound_companycode_getdetail (v. 1.0)		
MySAP_outbound_companycode_getdetail (v. 2.0) ★		
PSoft_outbound_location_invoke		
SAP_GetCCDetail		
Siebe_Outbound		
TaskActionHandler		
TaskManager		

3. Click the **BPEL Processes** tab.

   This tab provides a more detailed view of each deployed process.

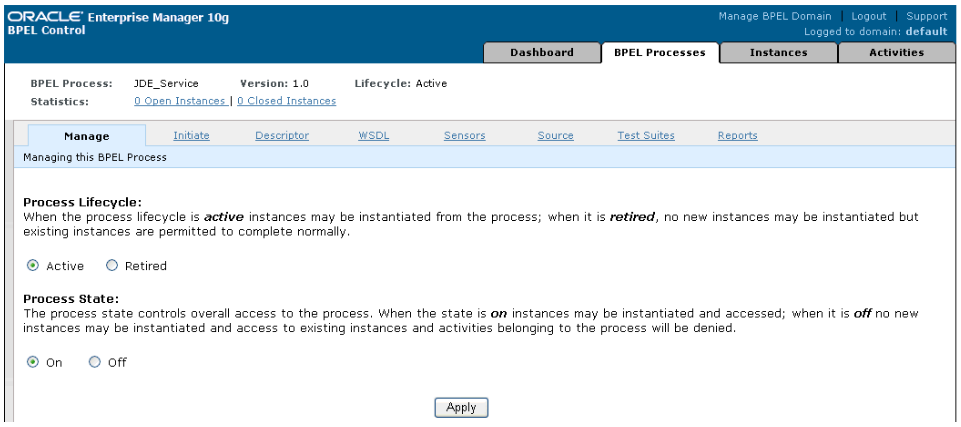


Deployed Processes

	BPEL Process	Lifecycle	State	Open Instances	Closed Instances
☐	JDE_Service (v. 1.0)	Active	On	0	0
☐	MySAP_outbound_companycode_getdetail (v. 1.0)	Active	On	0	4
☐	MySAP_outbound_companycode_getdetail (v. 2.0) ★	Active	On	0	0
☐	PSoft_outbound_location_invoke (v. 1.0)	Active	On	0	0
☐	SAP_GetCCDetail (v. 1.0)	Active	On	0	0
☐	Siebe_Outbound (v. 1.0)	Active	On	0	0
☐	TaskActionHandler (v. 1.0)	Active	On	0	0
☐	TaskManager (v. 1.0)	Active	On	0	0

4. Click the J.D. Edwards outbound process link.

The Manage window provides options for managing this BPEL process. Do not change any of the following default settings.


ORACLE Enterprise Manager 10g
BPEL Control
Manage BPEL Domain | Logout | Support
Logged to domain: default
Dashboard
BPEL Processes
Instances
Activities

BPEL Process: JDE_Service Version: 1.0 Lifecycle: Active
Statistics: 0 Open Instances | 0 Closed Instances

Managing this BPEL Process

Process Lifecycle:
When the process lifecycle is ***active*** instances may be instantiated from the process; when it is ***retired***, no new instances may be instantiated but existing instances are permitted to complete normally.

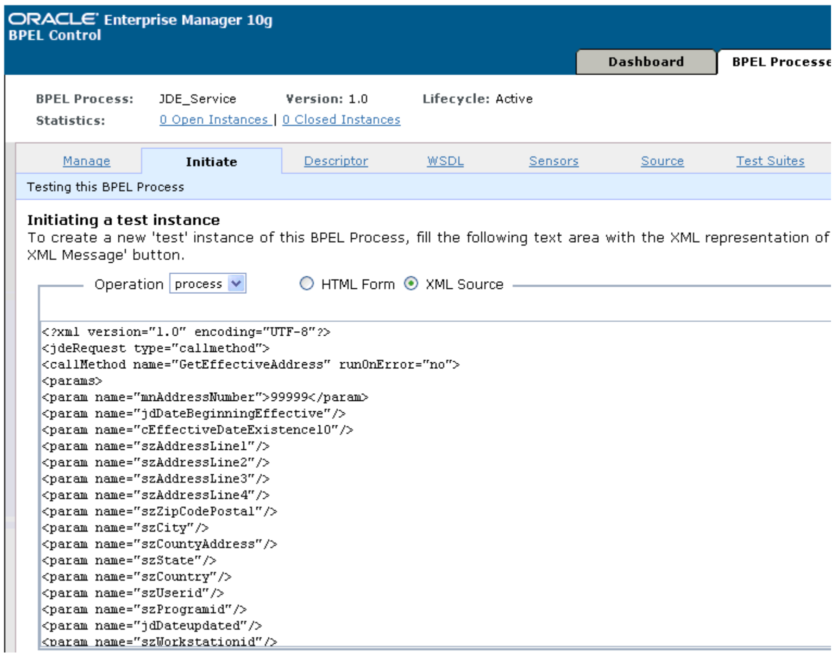
[✓ Active] Retired

Process State:
The process state controls overall access to the process. When the state is ***on*** instances may be instantiated and accessed; when it is ***off*** no new instances may be instantiated and access to existing instances and activities belonging to the process will be denied.

[✓ On] Off

5. Click the **Initiate** tab.

   The Initiate tab enables you to test your BPEL process.



BPEL Process: JDE_Service Version: 1.0 Lifecycle: Active
Statistics: 0 Open Instances | 0 Closed Instances

Testing this BPEL Process

Initiating a test instance
To create a new 'test' instance of this BPEL Process, fill the following text area with the XML representation of XML Message' button.

Operation process HTML Form [✓ XML Source]

```
<?xml version="1.0" encoding="UTF-8"?>
<jdeRequest type="callmethod">
<callMethod name="GetEffectiveAddress" runOnError="no">
<params>
<param name="mnAddressNumber">99999</param>
<param name="jdDateBeginningEffective"/>
<param name="cEffectiveDateExistence10"/>
<param name="szAddressLine1"/>
<param name="szAddressLine2"/>
<param name="szAddressLine3"/>
<param name="szAddressLine4"/>
<param name="szZipCodePostal"/>
<param name="szCity"/>
<param name="szCountyAddress"/>
<param name="szState"/>
<param name="szCountry"/>
<param name="szUserid"/>
<param name="szProgramid"/>
<param name="jdDateupdated"/>
<param name="szWorkstationid"/>
```

Perform the following steps:

a. From the **Initiating a test instance** list, select **XML Source**.

b. Enter the following code in the text area provided for XML input:

```
<?xml version="1.0" encoding="UTF-8"?>
<jdeRequest type="callmethod">
<callMethod name="GetEffectiveAddress" runOnError="no">
<params>
<param name="mnAddressNumber">99999</param>
<param name="jdDateBeginningEffective"/>
<param name="cEffectiveDateExistence10"/>
<param name="szAddressLine1"/>
<param name="szAddressLine2"/>
<param name="szAddressLine3"/>
```

```
<param name="szAddressLine4"/>
<param name="szZipCodePostal"/>
<param name="szCity"/>
<param name="szCountyAddress"/>
<param name="szState"/>
<param name="szCountry"/>
<param name="szUserid"/>
<param name="szProgramid"/>
<param name="jdDateupdated"/>
<param name="szWorkstationid"/>
<param name="mnTimelastupdated"/>
<param name="szNamealpha"/>
</params>
<onError abort="yes"/>
</callMethod>
</jdeRequest>
```

6. Click **Post XML Message**.
7. Click the **Instances** tab, and then click the **Audit** tab to view the response received from the J.D. Edwards OneWorld system.

   See Also: *Oracle Application Server Adapter Concepts*

Listening to Adapter Events Inside Oracle BPEL Process Manager

The OracleAS Adapter for J.D. Edwards OneWorld event notification service, also referred to as Inbound Interaction, is used to listen to events that occur in an EIS. The following section describes how to deploy your inbound BPEL process and listen to adapter events at runtime using Oracle BPEL Console.

Deploy the Inbound BPEL Process

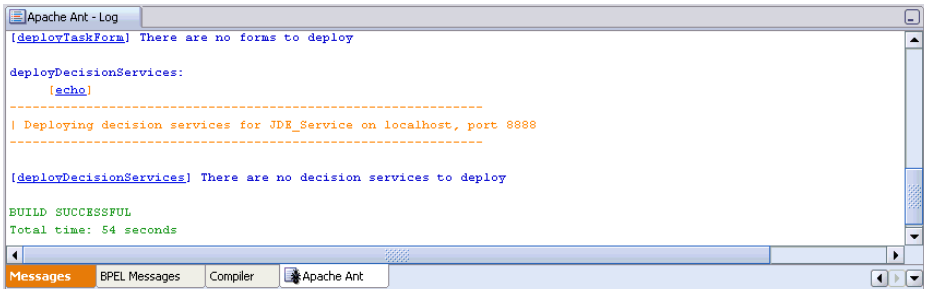
The procedures for deploying an inbound and an outbound BPEL process using the JDeveloper interface are identical.

To deploy your BPEL process in JDeveloper:

1. Right-click your project in the Applications Navigator tab.
2. Select **Deploy**, then ***Your BPEL PM Server connection***, and then **Deploy to default domain**.

   The deployment process starts automatically.
3. Observe the **Messages** log at the bottom of the window.

   The Messages log displays the deployment status. In this example, it shows a successful deployment message for the process.



```
[deployTaskForm] There are no forms to deploy

deployDecisionServices:
     [echo]
--------------------------------------------------------------
| Deploying decision services for JDE_Service on localhost, port 8888
--------------------------------------------------------------

[deployDecisionServices] There are no decision services to deploy

BUILD SUCCESSFUL
Total time: 54 seconds
```

If deployment was not successful, click the **Compiler** tab to view all error and warning messages generated during the deployment process.

Listen to Adapter Events in Oracle BPEL Console

JDeveloper deploys the developed process directly to Oracle BPEL Console, which enables you to run, monitor, and administer BPEL processes, as well as to listen to adapter events at runtime using Oracle BPEL Console.

To listen to adapter events:

1. Start the Oracle BPEL Console by entering the following URL in a browser:

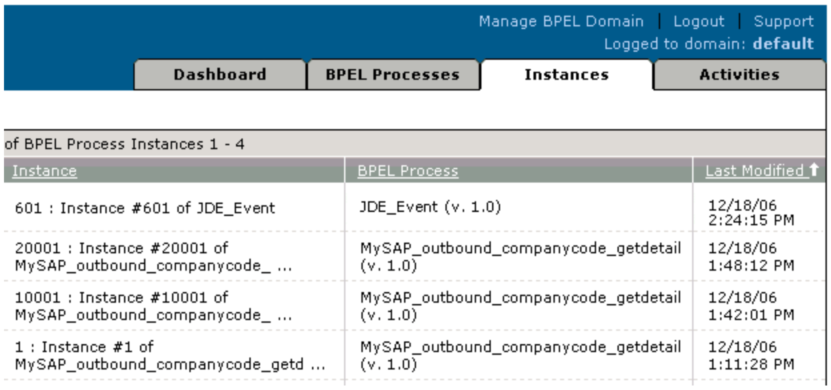
   `http://host:port/BPELConsole`

2. Select a domain and provide a valid password.

   The Oracle BPEL Console Dashboard tab is displayed.

3. Click the **Instances** tab.

   Upon receiving a runtime event, an instance of the event is displayed under the Instances tab.

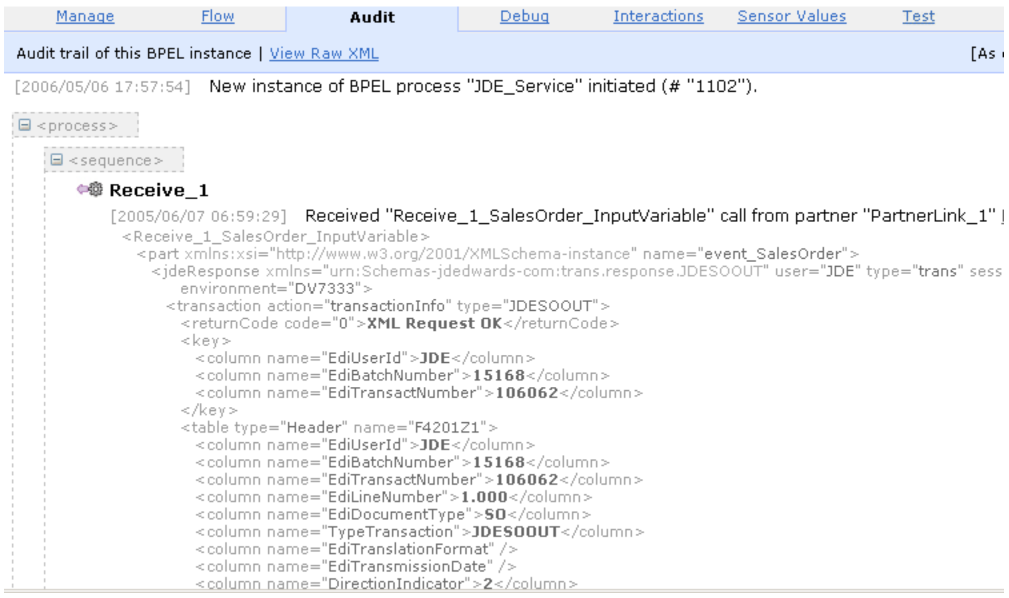

Manage BPEL Domain | Logout | Support
Logged to domain: default
Dashboard
BPEL Processes
Instances
Activities

of BPEL Process Instances 1 - 4

Instance	BPEL Process	Last Modified
601 : Instance #601 of JDE_Event	JDE_Event (v. 1.0)	12/18/06 2:24:15 PM
20001 : Instance #20001 of MySAP_outbound_companycode_ ...	MySAP_outbound_companycode_getdetail (v. 1.0)	12/18/06 1:48:12 PM
10001 : Instance #10001 of MySAP_outbound_companycode_ ...	MySAP_outbound_companycode_getdetail (v. 1.0)	12/18/06 1:42:01 PM
1 : Instance #1 of MySAP_outbound_companycode_getd ...	MySAP_outbound_companycode_getdetail (v. 1.0)	12/18/06 1:11:28 PM

4. To see the event message, click the instance, and then click the **Audit** tab.

   The event message is displayed.



```
Audit trail of this BPEL instance | View Raw XML                                  [As
[2006/05/06 17:57:54]  New instance of BPEL process "JDE_Service" initiated (# "1102").
<process>
  <sequence>
    Receive_1
      [2005/06/07 06:59:29]  Received "Receive_1_SalesOrder_InputVariable" call from partner "PartnerLink_1"
      <Receive_1_SalesOrder_InputVariable>
        <part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="event_SalesOrder">
          <jdeResponse xmlns="urn:Schemas-jdedwards-com:trans.response.JDESOOUT" user="JDE" type="trans" sess
            environment="DV7333">
            <transaction action="transactionInfo" type="JDESOOUT">
              <returnCode code="0">XML Request OK</returnCode>
              <key>
                <column name="EdiUserId">JDE</column>
                <column name="EdiBatchNumber">15168</column>
                <column name="EdiTransactNumber">106062</column>
              </key>
              <table type="Header" name="F4201Z1">
                <column name="EdiUserId">JDE</column>
                <column name="EdiBatchNumber">15168</column>
                <column name="EdiTransactNumber">106062</column>
                <column name="EdiLineNumber">1.000</column>
                <column name="EdiDocumentType">SO</column>
                <column name="TypeTransaction">JDESOOUT</column>
                <column name="EdiTranslationFormat" />
                <column name="EdiTransmissionDate" />
                <column name="DirectionIndicator">2</column>
```

To view the XML source code, click **View Raw XML**. See Chapter 5, "BPEL Process Manager Integration Examples" on page 5-1 for more information.

See Also: *Oracle Application Server Adapter Concepts*

5 BPEL Process Manager Integration Examples

This chapter contains the following examples:

- J.D. Edwards OneWorld Service Integration
- J.D. Edwards OneWorld Event Integration

The scenarios shown in this chapter require the following prerequisites.

Prerequisites

The following are installation and configuration requirements:

- OracleAS Adapter for J.D. Edwards OneWorld must be installed on Oracle Application Server.
- Oracle BPEL PM Server must be properly configured and running.
- Oracle JDeveloper must be properly installed.

> **See Also:** *Oracle Application Server Adapters Installation Guide*

The examples in this chapter present the configuration steps necessary for demonstrating service and event integration with J.D. Edwards OneWorld. Prior to using this material, you must be familiar with the following:

- How to create a J2CA configuration, as BPEL PM is only compatible with the J2CA Connector. See "Creating a Configuration for J2CA" on page 2-9 for more information.
- How to configure OracleAS Adapter for J.D. Edwards OneWorld for services and events using Application Explorer. See Chapter 2, "Configuring OracleAS Adapter for J.D. Edwards One World" for more information.

> **See Also:** *Oracle BPEL Process Manager Developer's Guide*

Adapter integration with Oracle BPEL Process Manager is a two-step process:

1. **Design Time:** OracleAS Adapter for J.D. Edwards OneWorld is configured in Application Explorer for services and events, as described in Chapter 2, "Configuring OracleAS Adapter for J.D. Edwards One World". Integration logic is modeled using JDeveloper.
2. **Runtime:** After you deploy the BPEL process you designed in JDeveloper, you can test your service configuration or see newly received events in the BPEL Console.

J.D. Edwards OneWorld Service Integration

This topic illustrates J.D. Edwards service integration. The design-time and runtime procedures are described below.

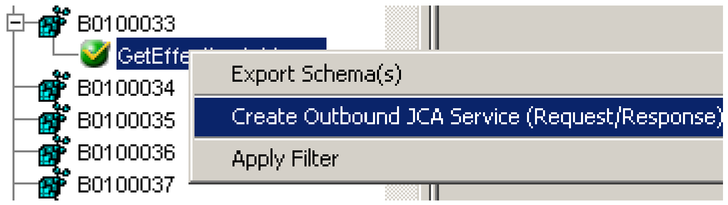
Design-Time Configuration

Generating WSDL for Request/Response Service

To generate WSDL for outbound interaction:

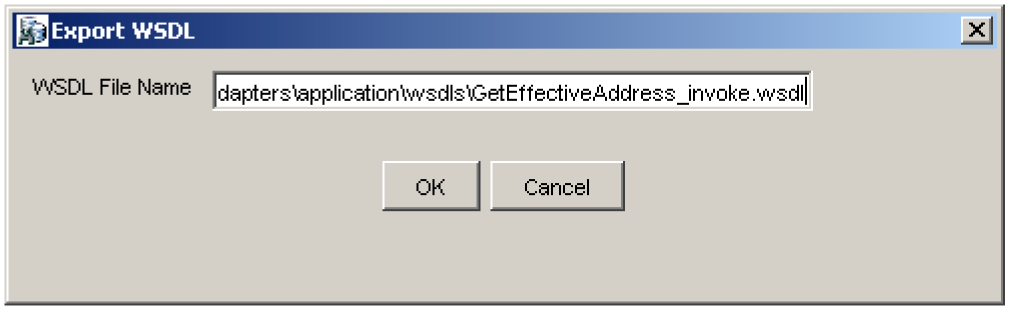
1. Start Application Explorer and connect to a defined J.D. Edwards target.
2. Expand **Services**, **JDEJAVA_CFIN**, **B0100033**, and then select **GetEffectiveAddress**.
3. Right-click **GetEffectiveAddress**.

   The following menu is displayed.



4. Select **Create Outbound JCA Service (Request/Response)**.

   The Export WSDL dialog box is displayed.



WSDL File Name: dapters\application\wsdls\GetEffectiveAddress_invoke.wsdl

5. Accept the default name and location for the file.

   The **.wsdl** file extension is added automatically.
6. Click **OK**.

   The WSDL file is created.

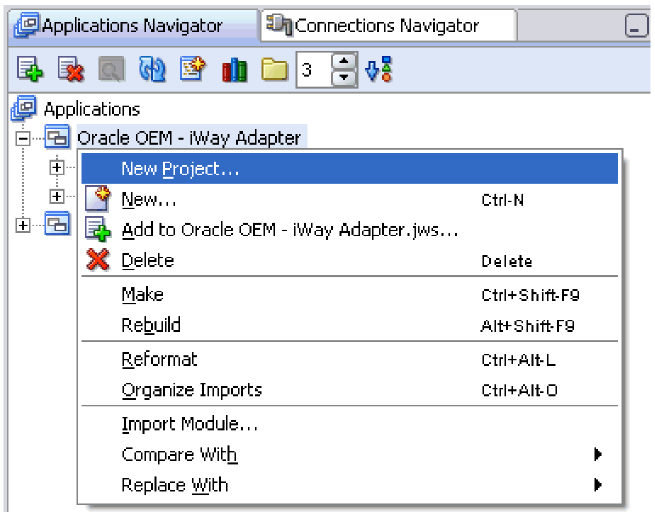
Creating a BPEL PM Server Connection

Before you design an outbound BPEL process, you must configure a new Application Server and Integration Server connection in Oracle JDeveloper. For more information, see Chapter 4, "Integration with Oracle BPEL Process Manager".

Creating a BPEL Project for a Synchronous BPEL Process

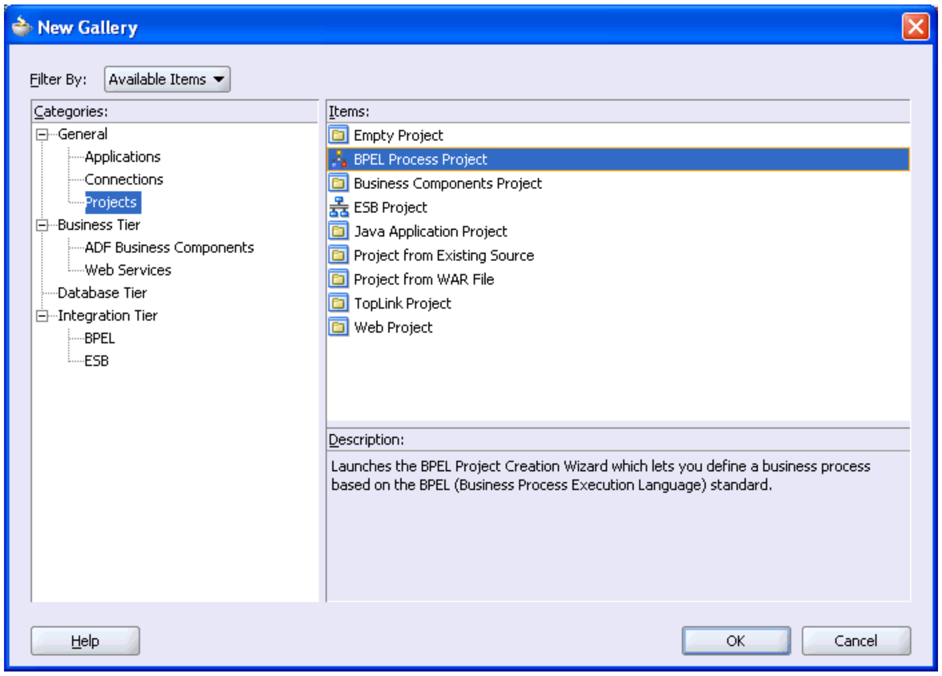
To create a BPEL Project for a synchronous BPEL process:

1. At the top of the upper left pane, click the **Applications Navigator** tab and select an application.



2. Right-click the application and select **New Project**.

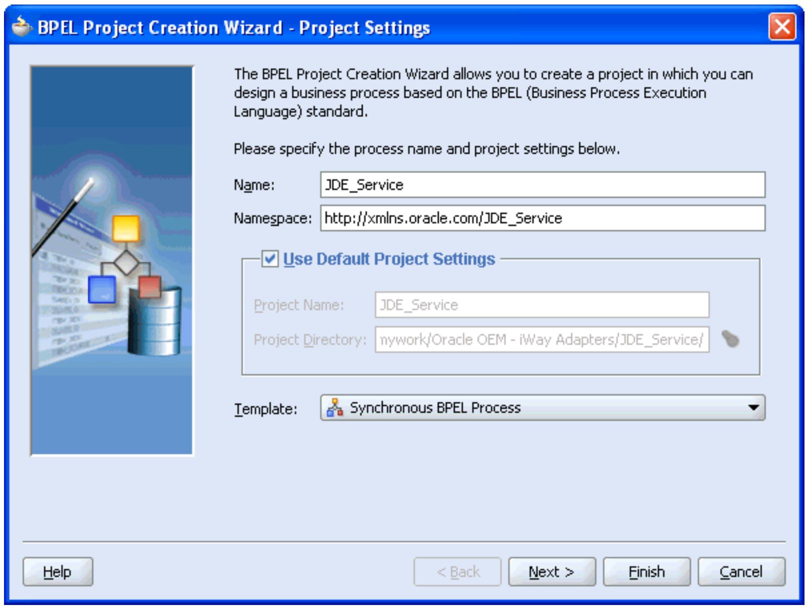
   The New Gallery window is displayed.



Launches the BPEL Project Creation Wizard which lets you define a business process based on the BPEL (Business Process Execution Language) standard.

3. From the Items list, select **BPEL Process Project** and click **OK**.

The BPEL Project Creation Wizard - Project Settings dialog box is displayed.



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: JDE_Service

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

[✓] Use Default Project Settings

Project Name: JDE_Service

Project Directory: nywork/Oracle OEM - iWay Adapters/JDE_Service/

Template: Synchronous BPEL Process

4. Perform the following steps:
   a. Specify a name for the BPEL project, for example, **JDE_Service**.

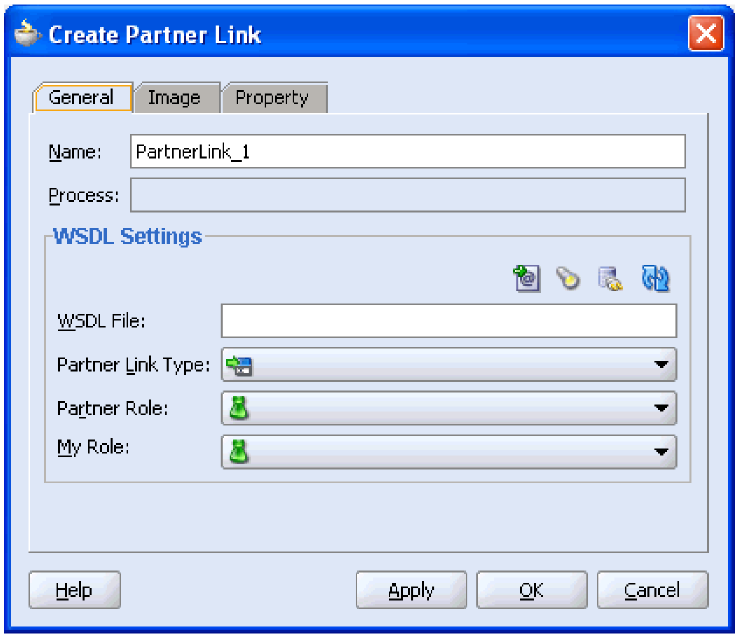
      The Namespace field is updated automatically.
   b. From the Template list, select **Synchronous BPEL Process**.
5. Click **OK**.

Designing the BPEL Process for GetEffectiveAddress (Outbound Service)

To design the BPEL Process:

1. From the Services pane on the right, drag and drop a PartnerLink to the visual editor.

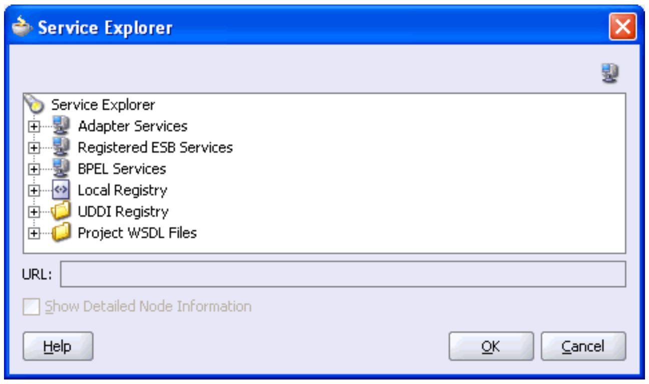
The Create Partner Link dialog box is displayed.



Name: PartnerLink_1

2. Click the **Service Explorer** icon (second icon from the left preceding the **WSDL File** field).

   The Service Explorer dialog box is displayed.



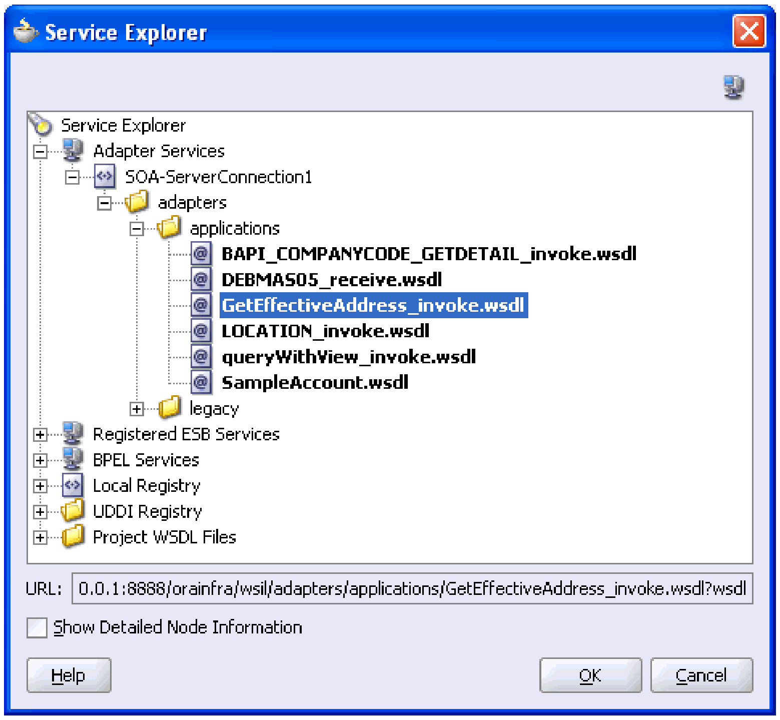
Service Explorer
- Adapter Services
- Registered ESB Services
- BPEL Services
- Local Registry
- UDDI Registry
- Project WSDL Files

URL:

Show Detailed Node Information

3. Expand your new connection under Adapter Services, followed by **adapters**, and then **applications**.

   The WSDL tree displayed in the Service Explorer dialog box lists any WSDL files you have created using Application Explorer. The WSDL tree is generated by a WSDL servlet, which is automatically deployed as part of the BPEL Server installation.

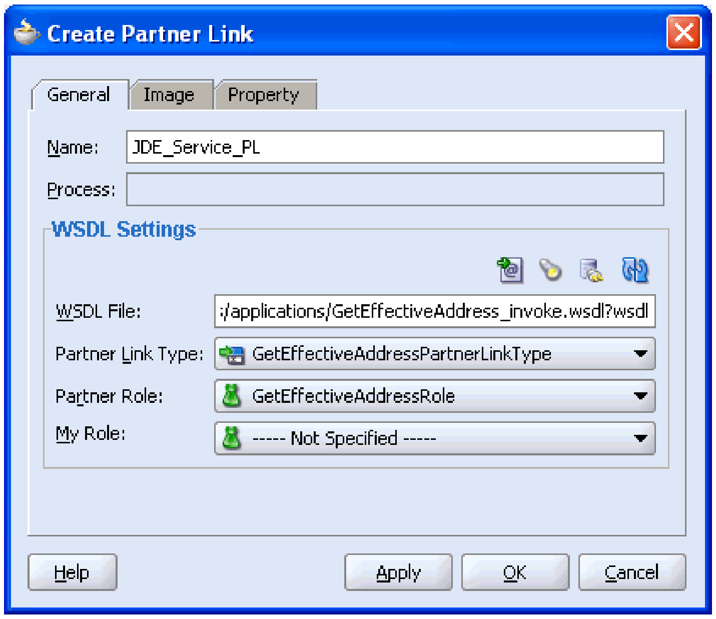

Service Explorer

- Service Explorer
  - Adapter Services
    - SOA-ServerConnection1
      - adapters
        - applications
          - BAPI_COMPANYCODE_GETDETAIL_invoke.wsdl
          - DEBMAS05_receive.wsdl
          - GetEffectiveAddress_invoke.wsdl
          - LOCATION_invoke.wsdl
          - queryWithView_invoke.wsdl
          - SampleAccount.wsdl
        - legacy
  - Registered ESB Services
  - BPEL Services
  - Local Registry
  - UDDI Registry
  - Project WSDL Files

URL: 0.0.1:8888/orainfra/wsil/adapters/applications/GetEffectiveAddress_invoke.wsdl?wsdl

4. Select **GetEffectiveAddress_invoke.wsdl** and click **OK**.

   The **WSDL File** field in the Create Partner Link dialog box displays the name and location of the selected WSDL file. The **Partner Link Type** field specifies the PartnerLink defined in the WSDL file.


Create Partner Link

Name: JDE_Service_PL

Process:

WSDL Settings

WSDL File: :/applications/GetEffectiveAddress_invoke.wsdl?wsdl

Partner Link Type: GetEffectiveAddressPartnerLinkType

Partner Role: GetEffectiveAddressRole

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

   Perform the following steps:

   a. Leave the **My Role** field unspecified. The role of the PartnerLink is null, as it will be synchronously invoked from the BPEL process.

   b. From the **Partner Role** list, select the default value **GetEffectiveAddressRole**. This is the role of the BPEL process.

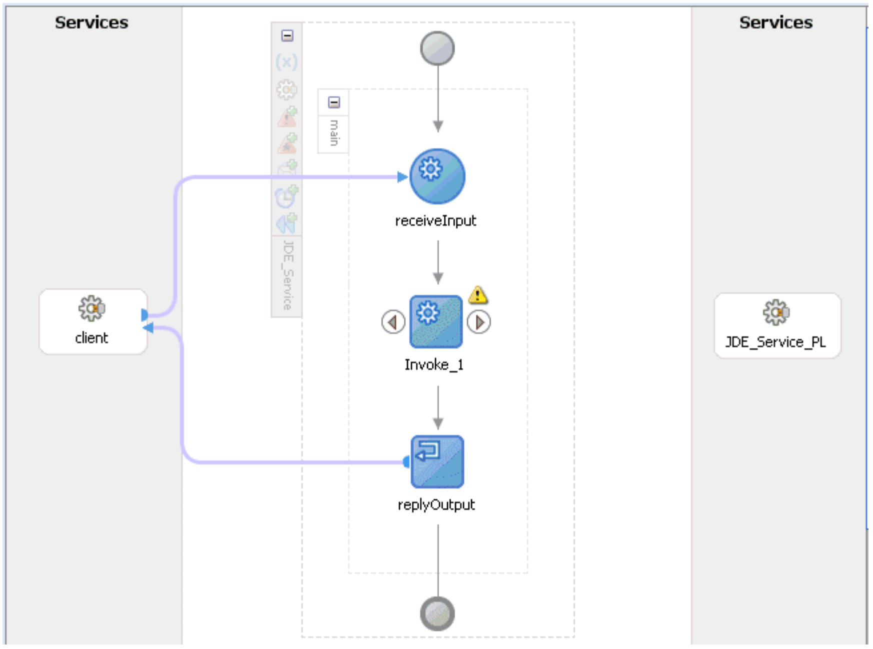
5. Click **OK**.

   The new PartnerLink appears in the visual editor.

6. Select **Save** from the File menu.

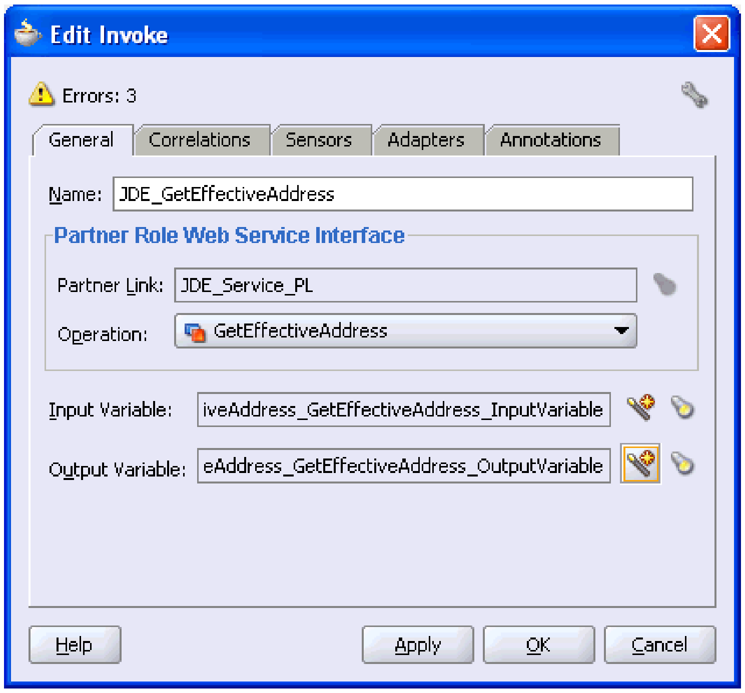
7. From the **Process Activities** pane on the right, drag an **Invoke** activity to the visual editor and place it between the Receive activity (`receiveInput`) and the Reply activity (`replyOutput`).

   The Invoke process activity is shown in the following diagram view.



8. Drag the right arrow from Invoke_1 and connect it to the `JDE_Service_PL` PartnerLink.

The Edit Invoke dialog box is displayed.


Edit Invoke

Errors: 3

Name: JDE_GetEffectiveAddress

Partner Role Web Service Interface

Partner Link: JDE_Service_PL

Operation: GetEffectiveAddress

Input Variable: iveAddress_GetEffectiveAddress_InputVariable

Output Variable: eAddress_GetEffectiveAddress_OutputVariable

Perform the following steps:

a. Provide a name for the Invoke activity, for example, **JDE_ GetEffectiveAddress**.

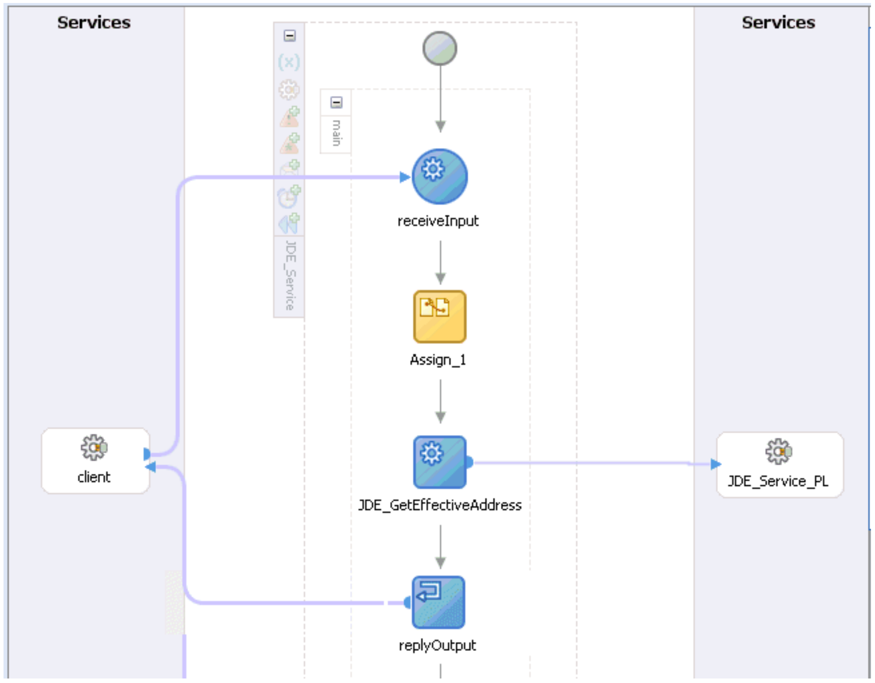
b. Click the first icon to the right of the **Input Variable** field, then click **OK** in the Create Variable window that is displayed.

c. Repeat the previous step to create a default variable for Output Variable.

9. Click **OK**.

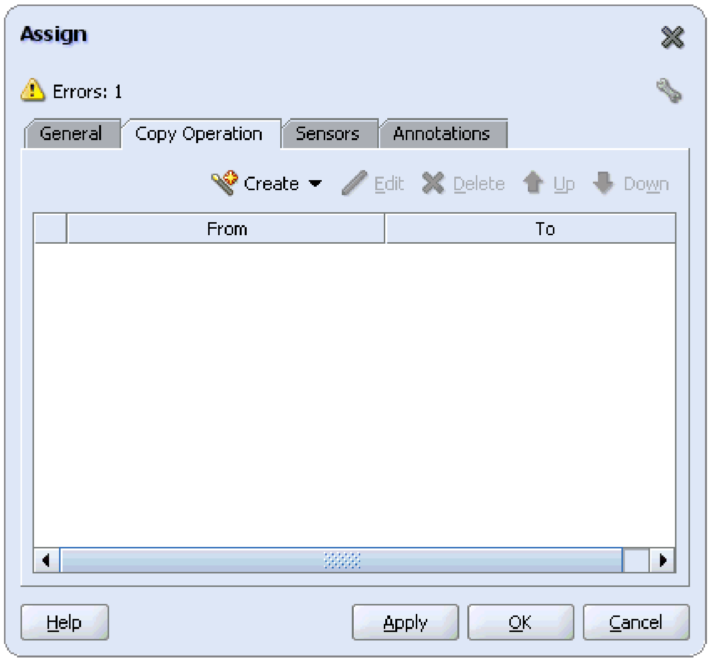
10. Drag an Assign process activity and drop it between `receiveInput` and `JDE_GetEffectiveAddress`.

The following image shows the new Assign activity in JDeveloper visual editor.



11. Double-click the **Assign** activity icon.

The Assign dialog box is displayed.



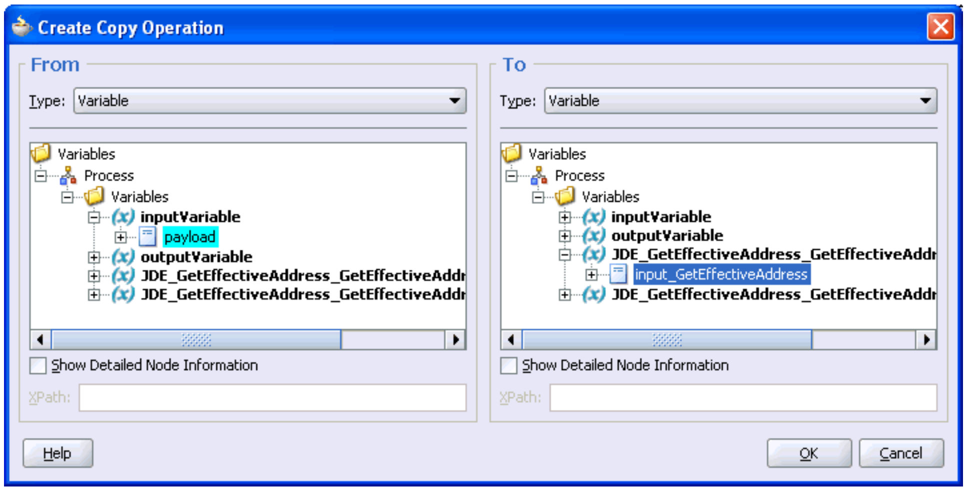
12. In the Copy Operation tab, click **Create** and select **Copy Operation**.

The Create Copy Operation dialog box is displayed.

a. In the **From** pane, expand **Variables**, then **inputVariable**, and then highlight **payload**.

b. In the **To** pane, expand **Variables**, then **JDE_GetEffectiveAddress_GetEffectiveAddress_InputVariable**, and then highlight **input_GetEffectiveAddress**.

Your Create Copy Operation dialog box should look as follows:


Create Copy Operation

From — Type: Variable

- Variables
  - Process
    - Variables
      - inputVariable
        - payload
      - outputVariable
      - JDE_GetEffectiveAddress_GetEffectiveAddr
      - JDE_GetEffectiveAddress_GetEffectiveAddr

Show Detailed Node Information

XPath:

To — Type: Variable

- Variables
  - Process
    - Variables
      - inputVariable
      - outputVariable
      - JDE_GetEffectiveAddress_GetEffectiveAddr
        - input_GetEffectiveAddress
      - JDE_GetEffectiveAddress_GetEffectiveAddr

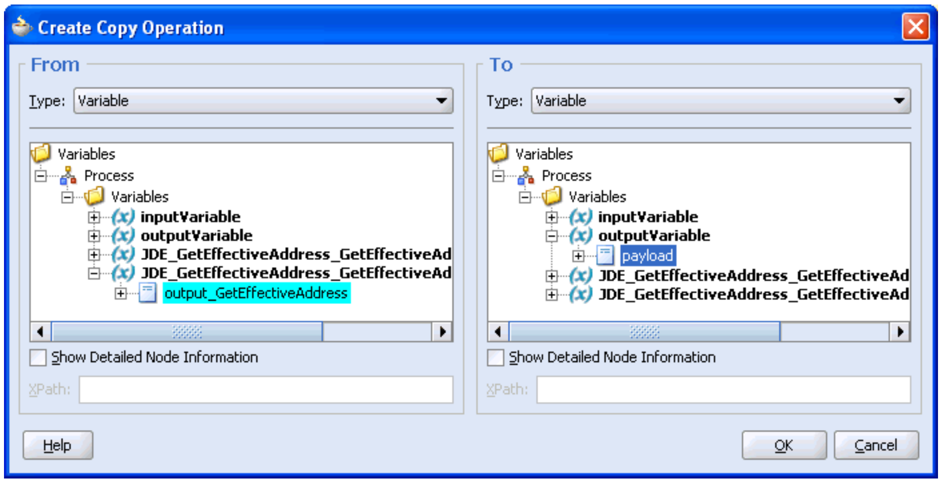
Show Detailed Node Information

XPath:

Help OK Cancel

13. To close the Create Copy Operation dialog box and the Assign dialog box, click **OK**.
14. From the **Process Activities** pane on the right, drag another **Assign** activity to the visual editor and place it between the Invoke activity (`JDE_GetEffectiveAddress`) and the Reply activity (`replyOutput`).
15. Double-click the **Assign** activity icon and click **Create**.
16. Map **JDE_GetEffectiveAddress_GetEffectiveAddress_OutputVariable**, **output_GetEffectiveAddress** to **outputVariable**, **payload**.

    Verify that you have mapped all variables as follows:


Create Copy Operation

From — Type: Variable

- Variables
  - Process
    - Variables
      - inputVariable
      - outputVariable
      - JDE_GetEffectiveAddress_GetEffectiveAd
      - JDE_GetEffectiveAddress_GetEffectiveAd
        - output_GetEffectiveAddress

Show Detailed Node Information

XPath:

To — Type: Variable

- Variables
  - Process
    - Variables
      - inputVariable
      - outputVariable
        - payload
      - JDE_GetEffectiveAddress_GetEffectiveAd
      - JDE_GetEffectiveAddress_GetEffectiveAd

Show Detailed Node Information

XPath:

Help OK Cancel

17. Click **OK**, and then click **OK** again.
18. Select **Save** from the File menu.

You have completed the design of your BPEL process.

Deploying the BPEL Process for GetEffectiveAddress (Outbound Service)

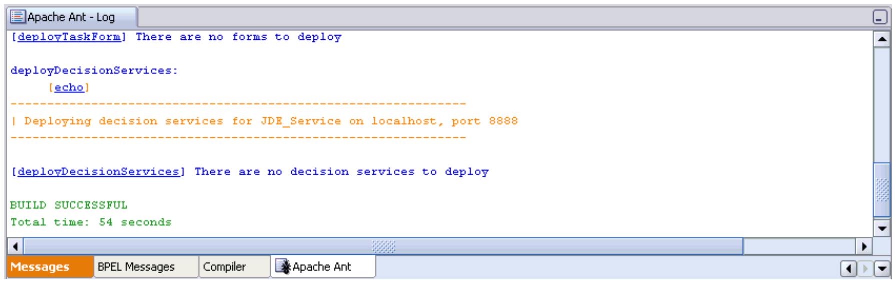
JDeveloper deploys BPEL processes directly to Oracle BPEL Console.

To deploy your BPEL process in JDeveloper:

1. Right-click your project in the Applications Navigator tab.
2. Select **Deploy**, then ***Your BPEL PM Server connection***, and then **Deploy to default domain**.

   The deployment process starts automatically.
3. Observe the **Messages** log at the bottom of the window.

   The Messages log displays the deployment status. In this example, it shows a successful deployment message for the process.



```
[deployTaskForm] There are no forms to deploy

deployDecisionServices:
     [echo]
--------------------------------------------------------------
| Deploying decision services for JDE_Service on localhost, port 8888
--------------------------------------------------------------

[deployDecisionServices] There are no decision services to deploy

BUILD SUCCESSFUL
Total time: 54 seconds
```

If deployment was not successful, click the **Compiler** tab to view all error and warning messages generated during the deployment process.

Runtime Configuration

To invoke the GetEffectiveAddress process from Oracle BPEL Console:

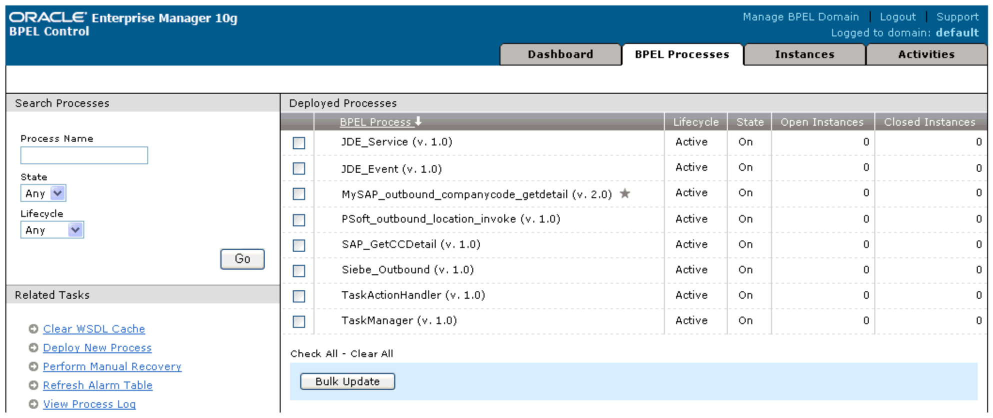
1. Start the Oracle BPEL Console by entering the following URL in a browser:

   http://*host*:*port*/BPELConsole

2. Provide a valid user name and password.

   The Oracle BPEL Console main page is displayed.
3. Click the **BPEL Processes** tab.

   Your deployed processes are displayed in this tab.

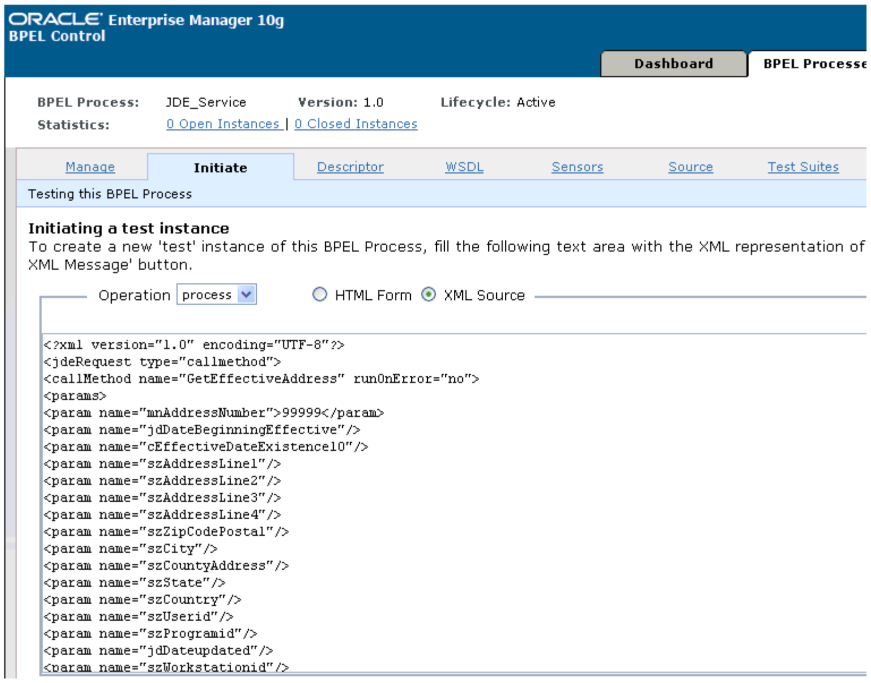


	BPEL Process	Lifecycle	State	Open Instances	Closed Instances
☐	JDE_Service (v. 1.0)	Active	On	0	0
☐	JDE_Event (v. 1.0)	Active	On	0	0
☐	MySAP_outbound_companycode_getdetail (v. 2.0) ★	Active	On	0	0
☐	PSoft_outbound_location_invoke (v. 1.0)	Active	On	0	0
☐	SAP_GetCCDetail (v. 1.0)	Active	On	0	0
☐	Siebe_Outbound (v. 1.0)	Active	On	0	0
☐	TaskActionHandler (v. 1.0)	Active	On	0	0
☐	TaskManager (v. 1.0)	Active	On	0	0

4. Click the JDE Service process link, **JDE_Service**.

5. Click the **Initiate** tab.

   The Initiate tab enables you to test your BPEL process.



BPEL Process: JDE_Service Version: 1.0 Lifecycle: Active
Statistics: 0 Open Instances | 0 Closed Instances

Testing this BPEL Process

Initiating a test instance
To create a new 'test' instance of this BPEL Process, fill the following text area with the XML representation of
XML Message' button.

Operation process

```
<?xml version="1.0" encoding="UTF-8"?>
<jdeRequest type="callmethod">
<callMethod name="GetEffectiveAddress" runOnError="no">
<params>
<param name="mnAddressNumber">99999</param>
<param name="jdDateBeginningEffective"/>
<param name="cEffectiveDateExistence10"/>
<param name="szAddressLine1"/>
<param name="szAddressLine2"/>
<param name="szAddressLine3"/>
<param name="szAddressLine4"/>
<param name="szZipCodePostal"/>
<param name="szCity"/>
<param name="szCountyAddress"/>
<param name="szState"/>
<param name="szCountry"/>
<param name="szUserid"/>
<param name="szProgramid"/>
<param name="jdDateupdated"/>
<param name="szWorkstationid"/>
```

Perform the following steps:

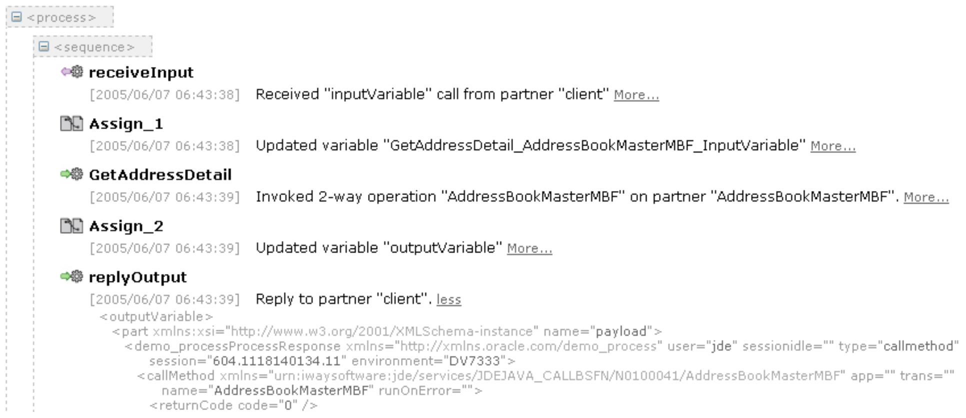
a. From the **Initiating a test instance** list, select **XML Source**.

b. Enter the following code in the text area provided for XML input:

```
<?xml version="1.0" encoding="UTF-8"?>
<jdeRequest type="callmethod">
<callMethod name="GetEffectiveAddress" runOnError="no">
<params>
<param name="mnAddressNumber">99999</param>
<param name="jdDateBeginningEffective"/>
<param name="cEffectiveDateExistence10"/>
<param name="szAddressLine1"/>
<param name="szAddressLine2"/>
<param name="szAddressLine3"/>
<param name="szAddressLine4"/>
<param name="szZipCodePostal"/>
<param name="szCity"/>
<param name="szCountyAddress"/>
<param name="szState"/>
<param name="szCountry"/>
<param name="szUserid"/>
<param name="szProgramid"/>
<param name="jdDateupdated"/>
<param name="szWorkstationid"/>
<param name="mnTimelastupdated"/>
<param name="szNamealpha"/>
</params>
<onError abort="yes"/>
</callMethod>
</jdeRequest>
```

6. Click **Post XML Message**.

Click the **Audit** tab to view the response received from the J.D. Edwards system.



```
<process>
  <sequence>
    receiveInput
      [2005/06/07 06:43:38]  Received "inputVariable" call from partner "client" More...
    Assign_1
      [2005/06/07 06:43:38]  Updated variable "GetAddressDetail_AddressBookMasterMBF_InputVariable" More...
    GetAddressDetail
      [2005/06/07 06:43:39]  Invoked 2-way operation "AddressBookMasterMBF" on partner "AddressBookMasterMBF". More...
    Assign_2
      [2005/06/07 06:43:39]  Updated variable "outputVariable" More...
    replyOutput
      [2005/06/07 06:43:39]  Reply to partner "client". less
       <outputVariable>
         <part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="payload">
           <demo_processProcessResponse xmlns="http://xmlns.oracle.com/demo_process" user="jde" sessionidle="" type="callmethod"
               session="604.1118140134.11" environment="DV7333">
             <callMethod xmlns="urn:iwaysoftware:jde/services/JDEJAVA_CALLBSFN/N0100041/AddressBookMasterMBF" app="" trans=""
                 name="AddressBookMasterMBF" runOnError="">
               <returnCode code="0" />
```

J.D. Edwards OneWorld Event Integration

This example illustrates how OracleAS Adapter for J.D. Edwards OneWorld integrates with J.D. Edwards OneWorld to receive event data. The design-time and runtime configuration procedures are outlined in the following sections.

Design-Time Configuration

Before you design a BPEL process using JDeveloper, you must create a separate channel for every J2CA event and select that channel when you generate WSDL for inbound interaction using Application Explorer.

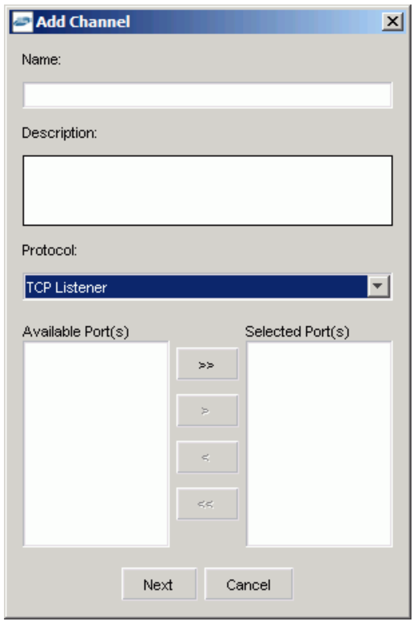
Note: If two or more events share the same channel, event messages may not be delivered to the right BPEL process.

Creating a Channel in Application Explorer

To create a channel:

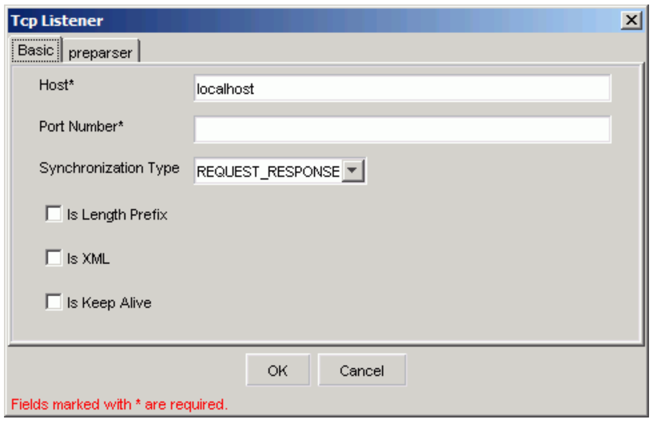
1. In Application Explorer, expand the **JDEdwards** node.
2. Right-click the **Channels** node, and select **Add Channels**.

The Add Channel dialog box is displayed.



3. In the **Name** field, enter a descriptive name for the channel.
4. In the **Description** field, enter a description (optional).
5. From the **Protocol** list, choose a protocol for your channel.
6. Click **Next**.

   The dialog box is displayed for the selected listener.



Host* localhost

Port Number*

Synchronization Type REQUEST_RESPONSE

Is Length Prefix

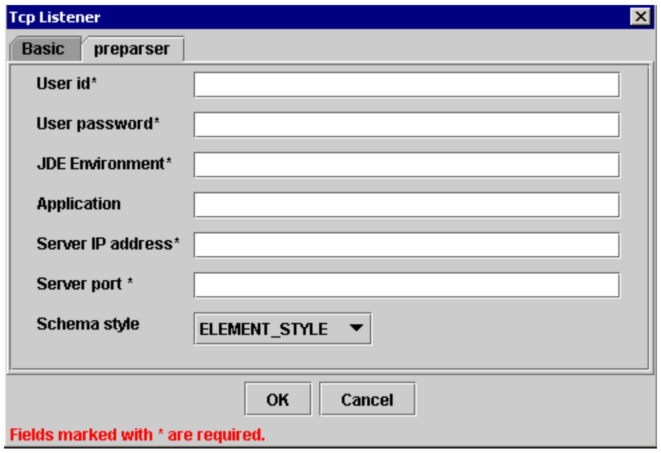
Is XML

Is Keep Alive

Fields marked with * are required.

7. Enter the location of the server in the **Host** field.
8. Enter the port number of the channel in the **Port Number** field.
9. Select the Synchronization type from the **Synchronization Type** list.
10. Select **Is Length Prefix** for events that send data which is not in XML format. The TCP/IP event application must prefix the data with a 4-byte binary length field when writing the data to the TCP/IP port.
11. Select **Is XML** for events that send data back in XML format. No preparser is required.

12. Select **Is Keep Alive** to maintain a continuous communication between the event transaction and the channel.
13. Click the **preparser** tab.



Enter values based on the table.

Parameter	Description
User id*	A valid user ID for J.D. Edwards OneWorld.
User password*	The password associated with the user ID.
JDE environment*	Your J.D. Edwards OneWorld environment. For more information about this parameter, see your J.D. Edwards OneWorld documentation or ask your OneWorld system administrator.
Application	XMLInterop or the application name in J.D. Edwards OneWorld. Optional.
Server IP address*	The name of the server on which J.D. Edwards OneWorld is running. This can be the name of the server, for example, JDEOW, or its IP address, for example, 123.45.67.89.
Server Port*	The port number on which the server is listening, for example, 6009.
Schema style	Choose a style from the list.

Click **OK**.

The channel is created and displayed under the Channels node. An X over the icon indicates that the channel is currently disconnected.

Note: The channel you created in Application Explorer is managed by BPEL PM Server. If you start the channel for testing and debugging purposes, stop it before runtime.

Generating WSDL for Event Notification (Command Prompt Only)

You cannot generate WSDL for J.D. Edwards OneWorld event notification using Application Explorer. To generate WSDL from the command prompt, you must perform the following steps.

Important: You can create inbound J2CA service only if the node you have selected supports events.

To generate a WSDL file for J.D. Edwards OneWorld event notification:

1. Create a channel in Application Explorer under the J.D. Edwards events node. See "Creating a Channel in Application Explorer" on page 5-13 .
2. Start the channel.

   Important: Do not restart the BPEL PM Server or Oracle Application Server after the channel is started.
3. Send an inbound message from J.D. Edwards OneWorld.
4. Capture the inbound message payload in the log file located under *`OracleAS_home`*`\adapters\application\config\jca_sample\log\iwaf_jca15.log`

   Alternatively, you can create a port with the File protocol under Event in Application Explorer to dispose the event message to the file system.
5. Use a third party tool such as XMLSpy to create the XSD schema using the XML payload captured in the previous step.
6. Verify that `orabpel-adapters.jar` is in your classpath.
7. Open a command prompt and change directory to: *`OracleAS_home`*`\adapters\application\tools`
8. Run `obadapter.bat` to set the environment.
9. Enter the following command to generate WSDL:

   ```
   java com.iwaysoftware.af.container.tools.wsdl.IWayWSILBrowser jcaHome adapter
   target channel schemaPrefix wsdlFileName
   ```

Where:

- `jcaHome` is the path to your J2CA configuration. For example, *`OracleAS_home`*`\adapters\application\`
- `adapter` is the name of the adapter. For example, JDEdwards.
- `target` is the name of the adapter target you created under Adapter in Application Explorer
- `channel` is the name of the channel you have created under Event in Application Explorer
- `schemaPrefix` is the prefix of the XSD schema. The schema file must be in the same directory where the Java command is executed, for example, *`OracleAS_home`*`\adapters\application\config\jca_sample\schemas\JDEdwards\bvision02`
- `wsdlFileName` is the name of the WSDL file being created in this procedure

For example:

```
C:\soaga\adapters\application\config\jca_
sample\schemas\JDEdwards\bvision02>java
com.iwaysoftware.af.container.tools.wsdl.IWayWSILBrowser
c:\soaga\adapters\application\  JDEdwards bvision02 JDE PO purchaseorder.wsdl
```

Once the command is successfully executed, the following output will be shown in the command window:

```
Running Inbound WSDL generation tool...
  -> user.dir = C:\soaga\adapters\application\config\jca_
sample\schemas\JDEdwards\bvision02
```

```
  -> Generating WSDL...
  -> Done.
  -> Writing WSDL 'c:\soaga\adapters\application\\wsdls\purchaseorder.wsdl' to
disk...
  -> Done.
```

Note: It is good practice to append `_receive` to the names of WSDL files generated for event notification services. This will allow you to easily differentiate between them and those generated for request-response services.

10. Stop the channel in Application Explorer.

Note: You can organize your WSDL files in subfolders, creating your own WSDL hierarchy structure. Create the folders under `*OracleAS_home*\adapters\application\wsdls\`. The WSIL browser in JDeveloper will display the full tree structure of your WSDL hierarchy.

The WSDL file is created. You can now create a BPEL process that uses the WSDL as Partner Link. As the BPEL process is deployed, the channel is activated from BPEL PM. Then, after you send an inbound message from J.D. Edwards OneWorld, the message will be received by the channel started by BPEL PM. These procedures are described in detail in the following sections.

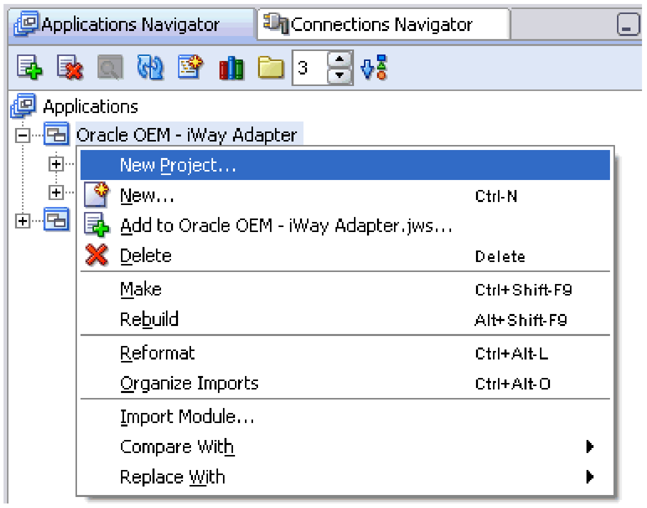
Creating a BPEL PM Server Connection

Before you design a BPEL process, you must configure a new Application Server and Integration Server connection in Oracle JDeveloper. For more information, see Chapter 4, "Integration with Oracle BPEL Process Manager".

Designing the BPEL Project for the SalesOrder Inbound Service

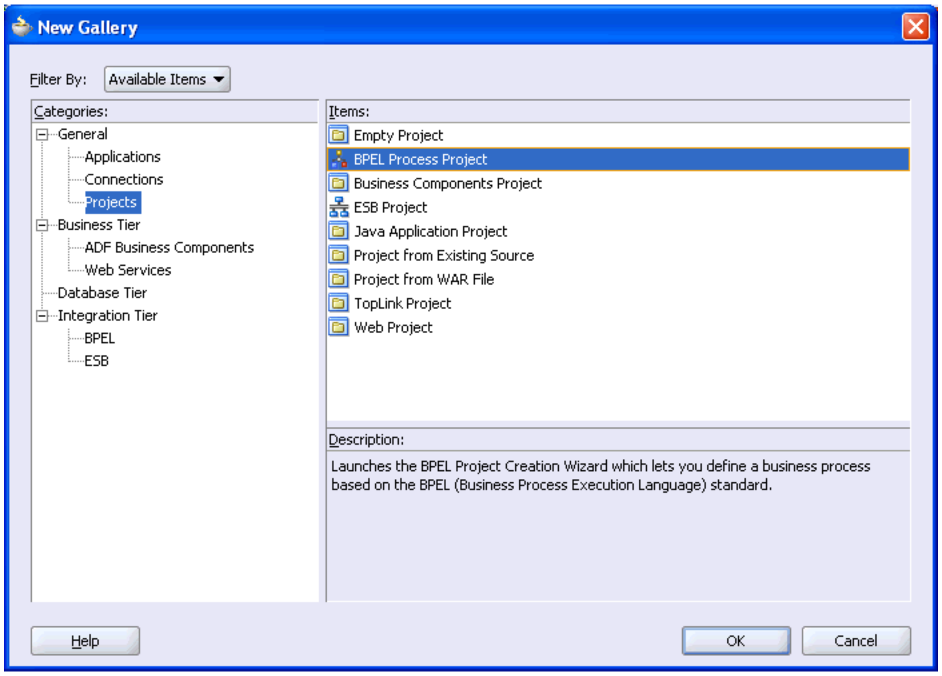
To design a BPEL project for inbound interaction:

1. At the top of the upper left pane, click the **Applications Navigator** tab and select an application.



2. Right-click the application and select **New Project**.

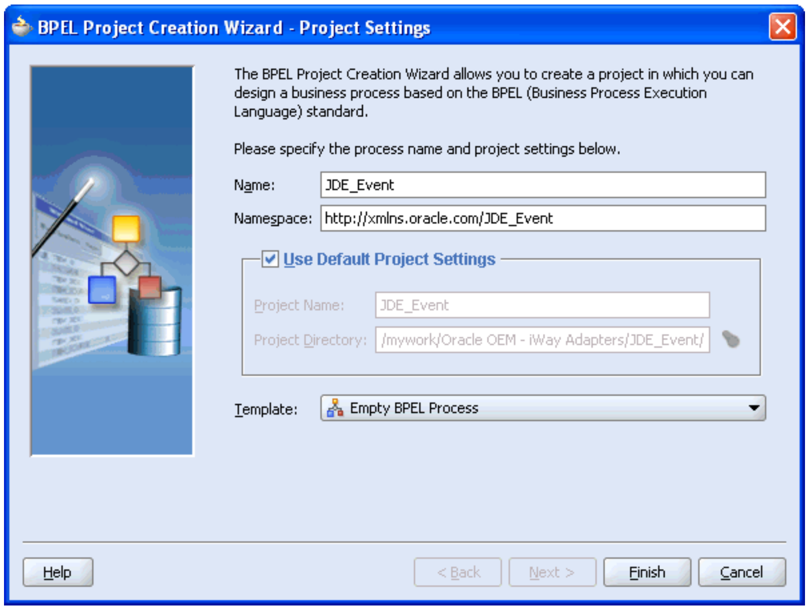
The New Gallery window is displayed.



Launches the BPEL Project Creation Wizard which lets you define a business process based on the BPEL (Business Process Execution Language) standard.

3. From the Items list, select **BPEL Process Project** and click **OK**.

   The BPEL Project Creation Wizard - Project Settings dialog box is displayed.


BPEL Project Creation Wizard - Project Settings
Use Default Project Settings
Help
< Back
Next >
Finish
Cancel

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: JDE_Event

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

Project Name: JDE_Event

Project Directory: /mywork/Oracle OEM - iWay Adapters/JDE_Event/

Template: Empty BPEL Process

4. Perform the following steps:

   a. Specify a name for the BPEL project.

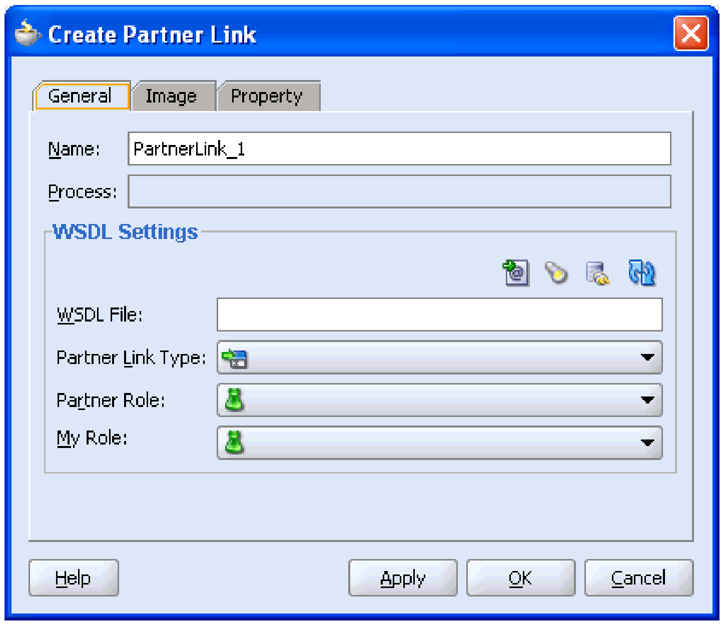
      The Namespace field is updated automatically.

   b. From the Template list, select **Empty BPEL Process**.

   c. Click **OK**.

5. From the Services pane on the right, drag and drop a **PartnerLink** to the visual editor.

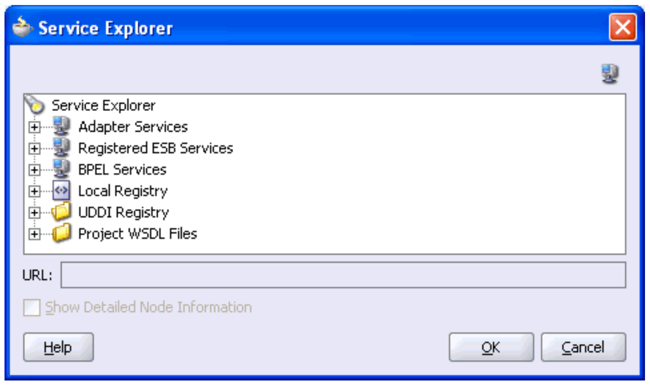
   The Create Partner Link dialog box is displayed.



Name: PartnerLink_1
Process:
WSDL File:
Partner Link Type:
Partner Role:
My Role:

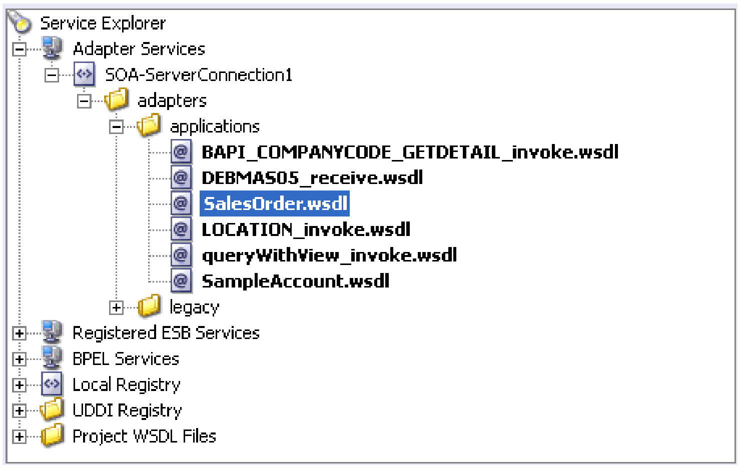
6. Click the **Service Explorer** icon (second icon from the left preceding the **WSDL File** field).

   The Service Explorer dialog box is displayed.



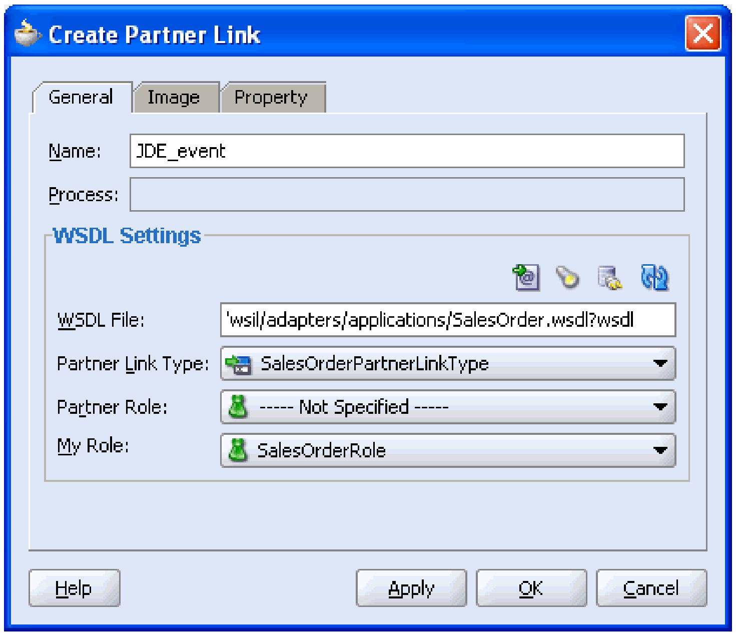
7. Expand your new connection under Adapter Services, followed by **adapters**, and then **applications**.

   The WSDL tree displayed in the Service Explorer dialog box lists any WSDL files you have created using Application Explorer. The WSDL tree is generated by a WSDL servlet, which is automatically deployed as part of the BPEL Server installation.


Service Explorer
Adapter Services
SOA-ServerConnection1
adapters
applications
BAPI_COMPANYCODE_GETDETAIL_invoke.wsdl
DEBMAS05_receive.wsdl
SalesOrder.wsdl
LOCATION_invoke.wsdl
queryWithView_invoke.wsdl
SampleAccount.wsdl
legacy
Registered ESB Services
BPEL Services
Local Registry
UDDI Registry
Project WSDL Files

8. Select **SalesOrder.wsdl** and click **OK**.

   The Create Partner Link dialog box is displayed.


Create Partner Link

General | Image | Property

Name: JDE_event

Process:

WSDL Settings

WSDL File: 'wsil/adapters/applications/SalesOrder.wsdl?wsdl

Partner Link Type: SalesOrderPartnerLinkType

Partner Role: ----- Not Specified -----

My Role: SalesOrderRole

The **WSDL File** field displays the name and location of the selected WSDL file. The **Partner Link Type** field specifies the PartnerLink defined in the WSDL file.

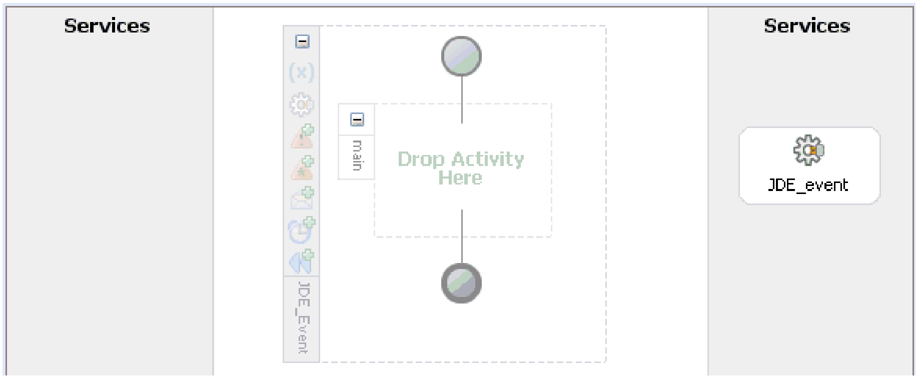
Perform the following steps:

a. From the **My Role** list, select the default value **SalesOrderRole**.

b. Leave the **Partner Role** field unspecified.

9. Click **Apply**, and then click **OK**.

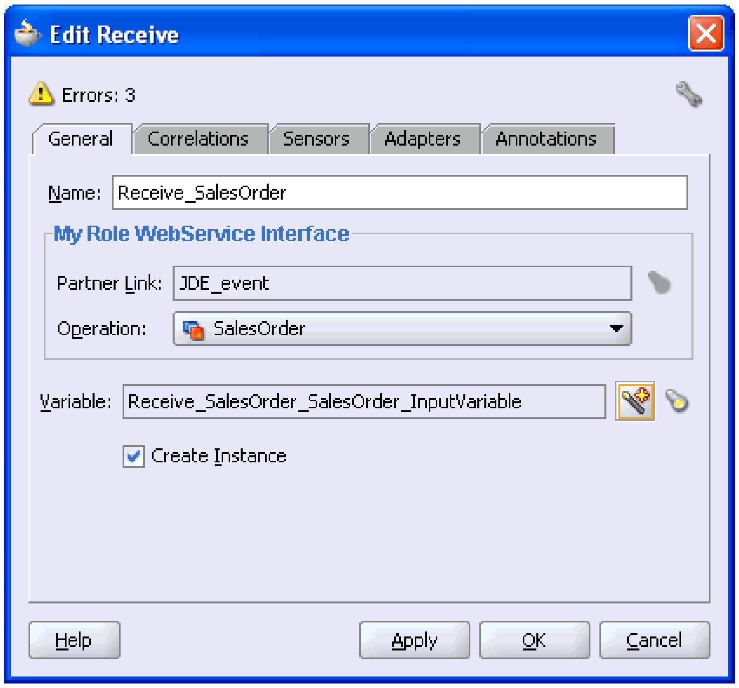
The new PartnerLink appears in the visual editor.



10. From the **Process Activities** pane on the right, drag a **Receive** activity to the visual editor and place it in the designated placeholder labeled **Drop Activity Here**.

11. Connect the Receive activity to the `JDE_event` PartnerLink.

    The Edit Receive dialog box is displayed.



Name: Receive_SalesOrder
Partner Link: JDE_event
Operation: SalesOrder
Variable: Receive_SalesOrder_SalesOrder_InputVariable
[✓] Create Instance

Perform the following steps:

a. Specify a name for the Receive Activity, for example, **Receive_SalesOrder**.

b. Click the first icon to the right of the **Variable** field, then click **OK** in the Create Variable dialog box that is displayed.

c. Verify that the **Create Instance** check box is selected.

12. Click **Apply**.

    The Receive dialog box should no longer display any warnings or errors.

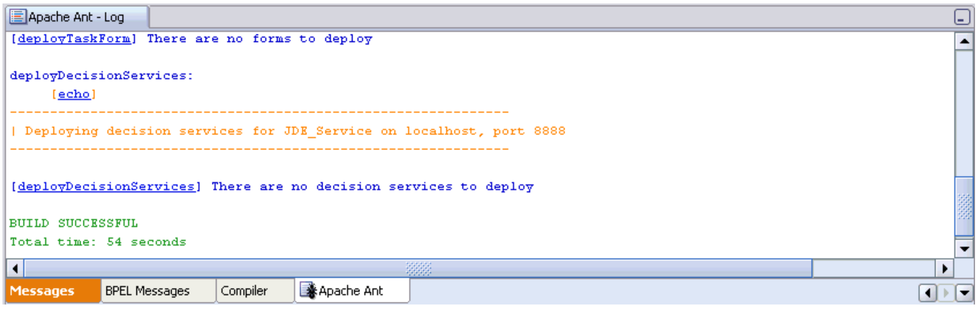
13. Click **OK**.

14. Select **Save** from the **File** menu.

Deploying the BPEL Project for the Inbound Service

1. Right-click your project in the Applications Navigator tab.
2. Select **Deploy**, then ***Your BPEL PM Server connection***, and then **Deploy to default domain**.

   The deployment process starts automatically.



```
[deployTaskForm] There are no forms to deploy

deployDecisionServices:
     [echo]
--------------------------------------------------------------
| Deploying decision services for JDE_Service on localhost, port 8888
--------------------------------------------------------------

[deployDecisionServices] There are no decision services to deploy

BUILD SUCCESSFUL
Total time: 54 seconds
```

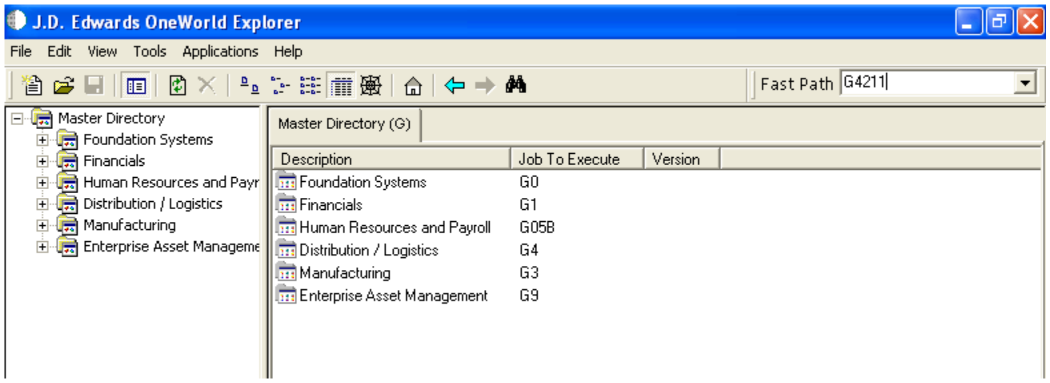
Runtime Configuration

Events are generated as a result of activity in a database or in an application system. You can use events to trigger an action in your application. See "Configuring an Event Adapter" on page 2-20 for more information on event configuration.

Triggering an Event in J.D. Edwards OneWorld

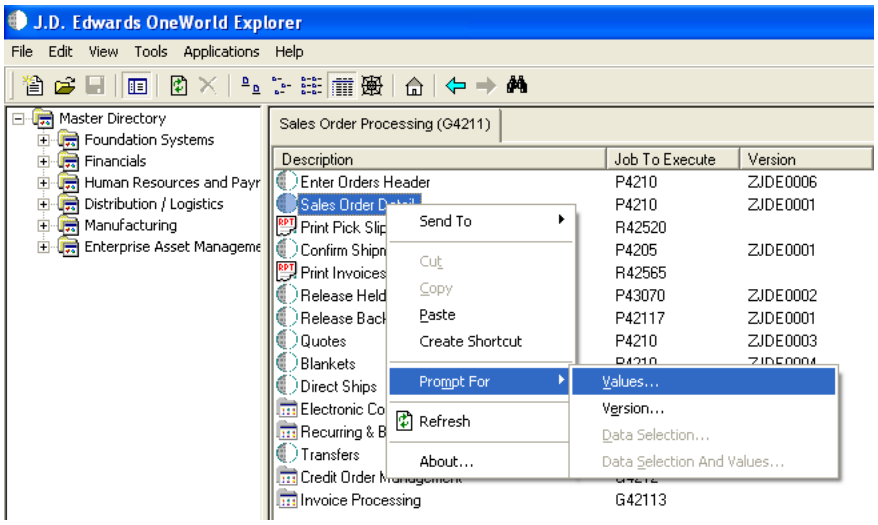
To trigger an event in J.D. Edwards OneWorld:

1. Log in to your J.D. Edwards OneWorld system.
2. In the **Fast Path** field of the J.D. Edwards OneWorld Explorer window, type **G4211** and press **Enter**.



Description	Job To Execute	Version
Foundation Systems	G0	
Financials	G1	
Human Resources and Payroll	G05B	
Distribution / Logistics	G4	
Manufacturing	G3	
Enterprise Asset Management	G9	

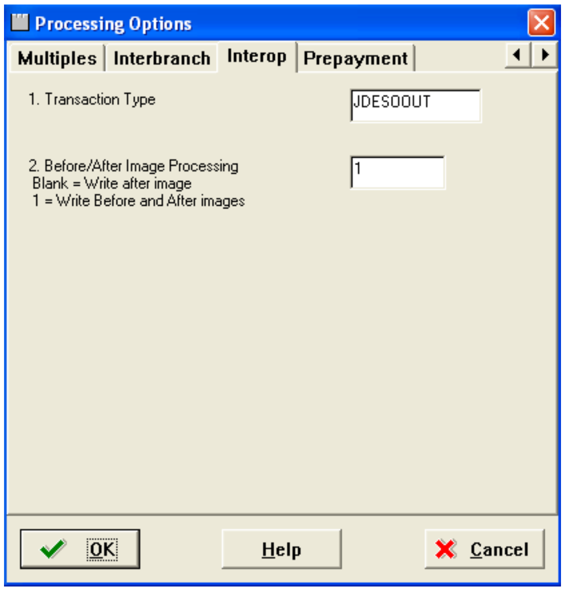
3. Right-click **Sales Order Detail** (P4210).



Description	Job To Execute	Version
Enter Orders Header	P4210	ZJDE0006
Sales Order D	P4210	ZJDE0001
Print Pick Slip	R42520	
Confirm Shipn	P4205	ZJDE0001
Print Invoices	R42565	
Release Held	P43070	ZJDE0002
Release Back	P42117	ZJDE0001
Quotes	P4210	ZJDE0003
Blankets	P4210	ZJDE0004
Direct Ships		
Electronic Co		
Recurring & B		
Transfers		
Credit Order Management	G4212	
Invoice Processing	G42113	

4. Select **Prompt for**, and then **Values**.

   The Processing Options dialog box is displayed.



1. Transaction Type: JDESOOUT

2. Before/After Image Processing
Blank = Write after image
1 = Write Before and After images: 1

   Perform the following steps:

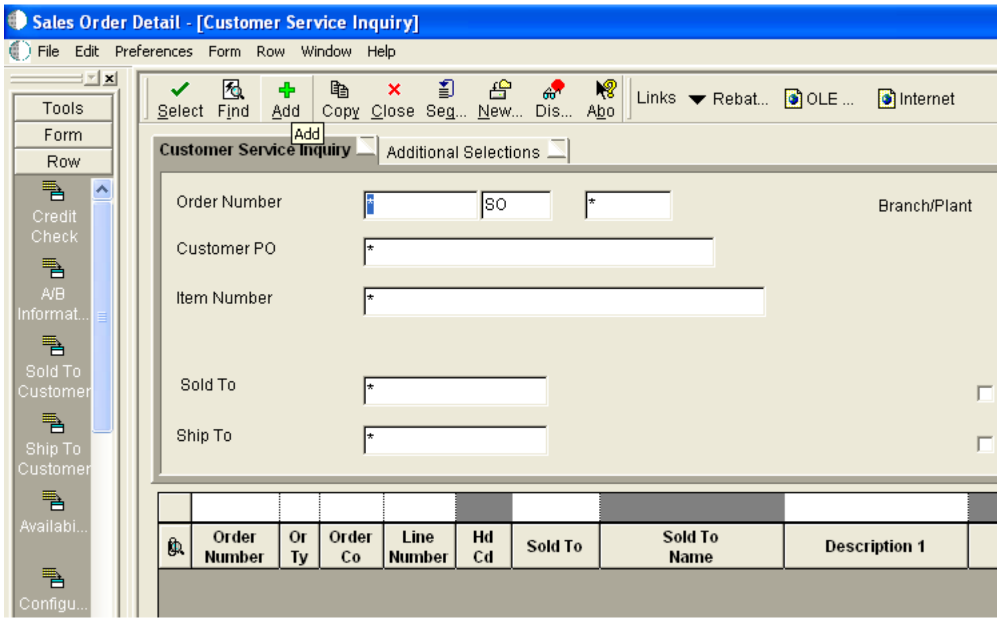
   a. Click the **Interop** tab.

   b. In the **Transaction Type** field, type **JDESOOUT**.

   c. Verify that the value in the **Before/After Image Processing Blank** field is 1.

5. Click **OK**.

   The **Sales Order Detail - (Customer Service Inquiry)** window is displayed.



Order Number: | SO | *

Customer PO: *

Item Number: *

Sold To: *

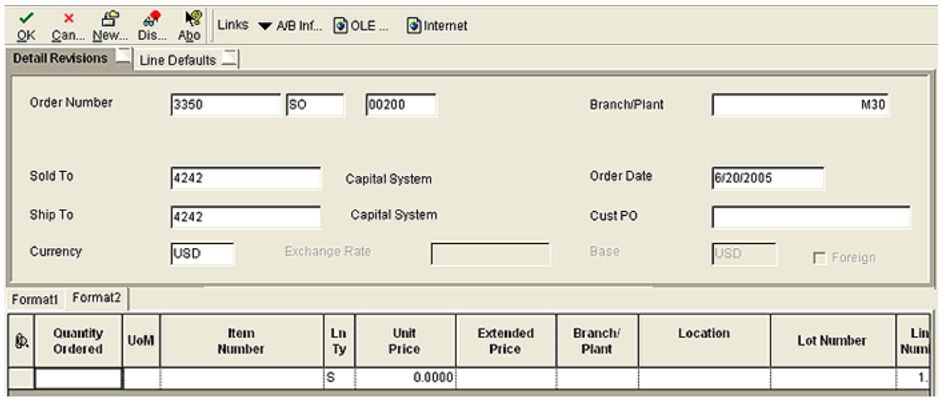
Ship To: *

	Order Number	Or Ty	Order Co	Line Number	Hd Cd	Sold To	Sold To Name	Description 1

6. Click the **Add** icon (third icon from left).

7. Enter the values as shown in the following screen.

To move to a different field, use the **Tab** key on your keyboard.



Order Number 3350 SO 00200 Branch/Plant M30

Sold To 4242 Capital System Order Date 6/20/2005

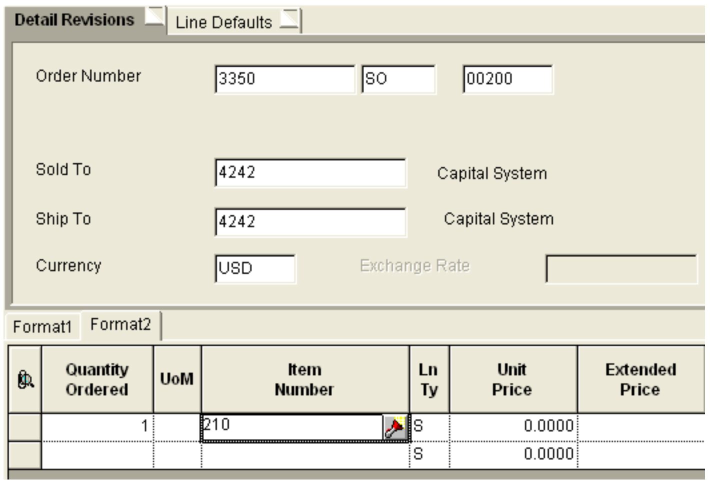
Ship To 4242 Capital System Cust PO

Currency USD Exchange Rate Base USD Foreign

Quantity Ordered	UoM	Item Number	Ln Ty	Unit Price	Extended Price	Branch/ Plant	Location	Lot Number	Lin Num
			S	0.0000					1.

8. Enter a value for **Quantity Ordered** and **Item Number**.

For example:



Order Number 3350 SO 00200

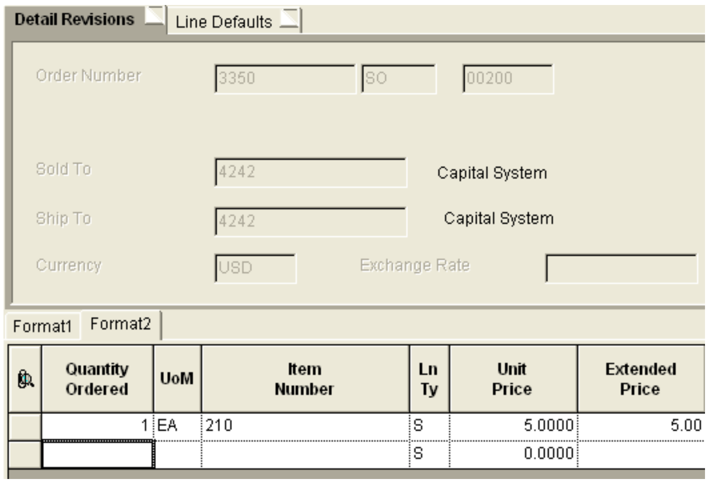
Sold To 4242 Capital System

Ship To 4242 Capital System

Currency USD Exchange Rate

Quantity Ordered	UoM	Item Number	Ln Ty	Unit Price	Extended Price
1		210	S	0.0000	
			S	0.0000	

9. Click the first field in the second row and allow a few seconds for processing.



Order Number 3350 SO 00200

Sold To 4242 Capital System

Ship To 4242 Capital System

Currency USD Exchange Rate

Quantity Ordered	UoM	Item Number	Ln Ty	Unit Price	Extended Price
1	EA	210	S	5.0000	5.00
			S	0.0000	

10. Click **OK**.

An event is triggered in the J.D. Edwards OneWorld system.

Verifying the Results

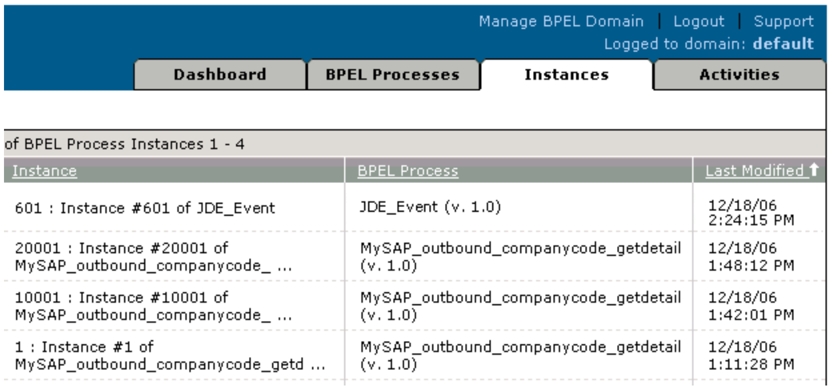
To verify your results:

1. Log in to Oracle BPEL Console at

   `http://`*`host`*`:`*`port`*`/BPELConsole`

2. Provide a valid user name and password.
3. Click the **Instances** tab.

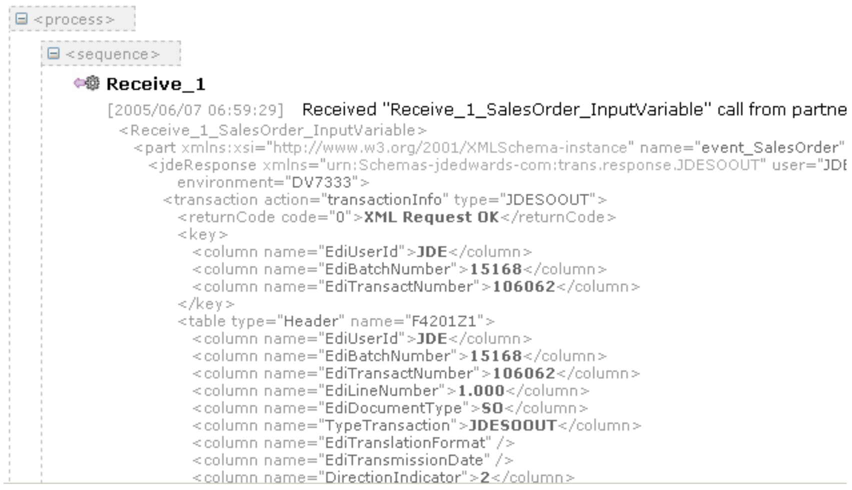
   Recently received runtime events are displayed in the Instances tab.


Manage BPEL Domain | Logout | Support
Logged to domain: default
Dashboard
BPEL Processes
Instances
Activities

of BPEL Process Instances 1 - 4

Instance	BPEL Process	Last Modified
601 : Instance #601 of JDE_Event	JDE_Event (v. 1.0)	12/18/06 2:24:15 PM
20001 : Instance #20001 of MySAP_outbound_companycode_ ...	MySAP_outbound_companycode_getdetail (v. 1.0)	12/18/06 1:48:12 PM
10001 : Instance #10001 of MySAP_outbound_companycode_ ...	MySAP_outbound_companycode_getdetail (v. 1.0)	12/18/06 1:42:01 PM
1 : Instance #1 of MySAP_outbound_companycode_getd ...	MySAP_outbound_companycode_getdetail (v. 1.0)	12/18/06 1:11:28 PM

4. Click your J.D. Edwards event instance, and then click **Audit** to see the received event message.



```
[2005/06/07 06:59:29]  Received "Receive_1_SalesOrder_InputVariable" call from partne
 <Receive_1_SalesOrder_InputVariable>
   <part xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="event_SalesOrder"
     <jdeResponse xmlns="urn:Schemas-jdedwards-com:trans.response.JDESOOUT" user="JDE
        environment="DV7333">
      <transaction action="transactionInfo" type="JDESOOUT">
        <returnCode code="0">XML Request OK</returnCode>
        <key>
          <column name="EdiUserId">JDE</column>
          <column name="EdiBatchNumber">15168</column>
          <column name="EdiTransactNumber">106062</column>
        </key>
        <table type="Header" name="F4201Z1">
          <column name="EdiUserId">JDE</column>
          <column name="EdiBatchNumber">15168</column>
          <column name="EdiTransactNumber">106062</column>
          <column name="EdiLineNumber">1.000</column>
          <column name="EdiDocumentType">SO</column>
          <column name="TypeTransaction">JDESOOUT</column>
          <column name="EdiTranslationFormat" />
          <column name="EdiTransmissionDate" />
          <column name="DirectionIndicator">2</column>
```

6

ESB Integration Examples

This chapter contains the following examples:

- Configuring an ESB Outbound Process
- Configuring an ESB Inbound Process

The scenarios shown in this chapter require the following prerequisites.

Prerequisites

The following are installation and configuration requirements:

- OracleAS Adapter for J.D. Edwards OneWorld must be installed on Oracle Application Server.
- J.D. Edwards OneWorld must be configured for inbound and outbound processing.
- OracleAS Technology adapters must be deployed and properly configured.

See Also: *Oracle Application Server Adapters Installation Guide*

The examples in this chapter present the configuration steps necessary for demonstrating service and event integration with J.D. Edwards OneWorld. Prior to using this material, you must be familiar with the following:

- How to configure OracleAS Adapter for J.D. Edwards OneWorld for services and events. For more information, see Chapter 2, "Configuring OracleAS Adapter for J.D. Edwards One World".
- How to configure a new Application Server and Integration Server connection in Oracle JDeveloper. For more information, see Chapter 4, "Integration with Oracle BPEL Process Manager".

Overview of ESB Integration

ESB provides a comprehensive application integration framework. OracleAS Adapter for J.D. Edwards OneWorld used in conjunction with ESB enables you to seamlessly integrate enterprise software, eliminating the need to write custom code. Functional modeling, as opposed to custom coding solutions, allows for software reuse and reduces the complexity and management challenges that arise over the software lifecycle. This integration model consists of two components--high-level integration logic and low-level platform services.

Adapter integration with OracleAS ESB is a two-step process:

1. **Design Time:** OracleAS Adapter for J.D. Edwards OneWorld is configured in Application Explorer for services and events, as described in Chapter 2,

"Configuring OracleAS Adapter for J.D. Edwards One World". Integration logic is modeled in iStudio. Metadata are stored in repositories.

2. **Runtime:** The underlying platform treats this metadata as runtime instructions to enable the communication between participating applications.

Namespace Requirements

The purpose of an XML namespace is to allow the deployment of XML vocabularies (where element and attribute names are defined) in a global environment and to reduce the risk of name collisions in a given document when vocabularies are combined. Qualified namespaces are used for stricter schema validation. In documents conforming to this specification, element and attribute names appear as qualified names. Syntactically, they are either prefixed names or unprefixed names. An attribute-based declaration syntax is provided to bind prefixes to namespace names and to bind a default namespace that applies to unprefixed element names. These declarations are scoped by the elements on which they appear so that different bindings may apply in different parts of a document. Processors conforming to this specification must recognize and act on these declarations and prefixes.

In the 10.1.3.1.0 SOA release, the recommendations for ESB integrations is to perform stricter name space validations. As a result, Application Explorer generates Web services for the backend with the namespace marked as "Qualified". This means that during testing or usage phases of this service by ESB, the request XML document that is used should adhere to the schema and WSDL document. Once again, it is important to remember that the namespaces are qualified. To further understand this point, the difference is illustrated with the following example:

1. Input XML for ESB based on unqualified namespaces:

```
<?xml version='1.0' encoding='utf-8' ?>
<jdeRequest type="callmethod" user="PSFT" pwd="PSFT" environment="DV810"
session="">
<callMethod name="GetEffectiveAddress">
  <params>
    <param name="mnAddressNumber">1001</param>
  </params>
</callMethod>
</jdeRequest>
```

2. Input XML for ESB based on qualified namespaces:

```
<jdeRequest
xmlns="urn:iwaysoftware:jde/services/CALLBSFN/AddressBook/GetEffectiveAddress"
type="callmethod" session="" sessionidle="">
<callMethod runOnError="" trans="String" name="GetEffectiveAddress"
app="string" returnNullData="yes">
  <params>
    <mnAddressNumber>1001</mnAddressNumber>
  </params>
  <onError abort="Yes"/>
 </callMethod>
</jdeRequest>
```

Note: If you are passing an unqualified input against a WSDL document that is expecting qualified namespaces, ESB will throw the exception as "Unable to process input xml...."

Configuring an ESB Outbound Process

The following example describes how to configure an ESB outbound process to your J.D. Edwards OneWorld system, using an ESB project in Oracle JDeveloper.

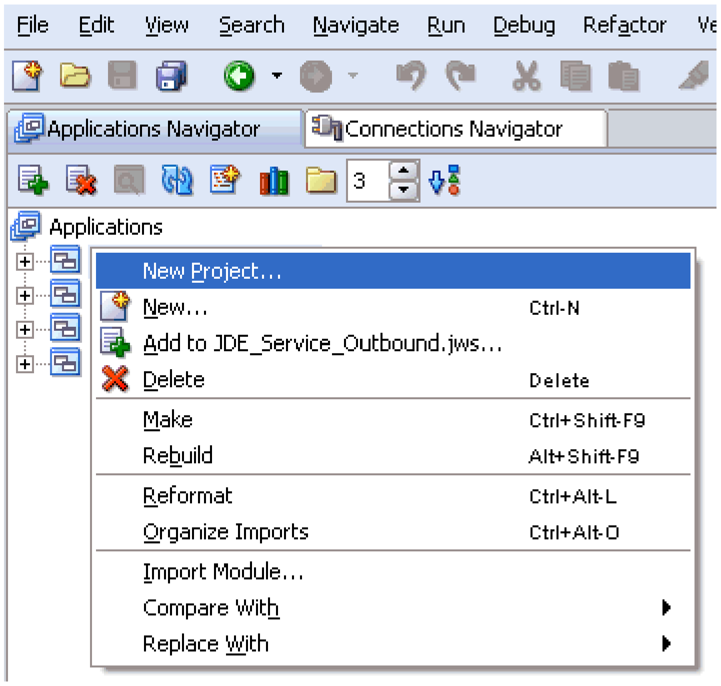
Prerequisites

Before you proceed, you must create an outbound WSDL file for the adapter by using the following steps:

1. Create a target using Application Explorer.
2. Connect to the target.
3. Create a WSDL file.
4. Restart the Oracle Application Server.

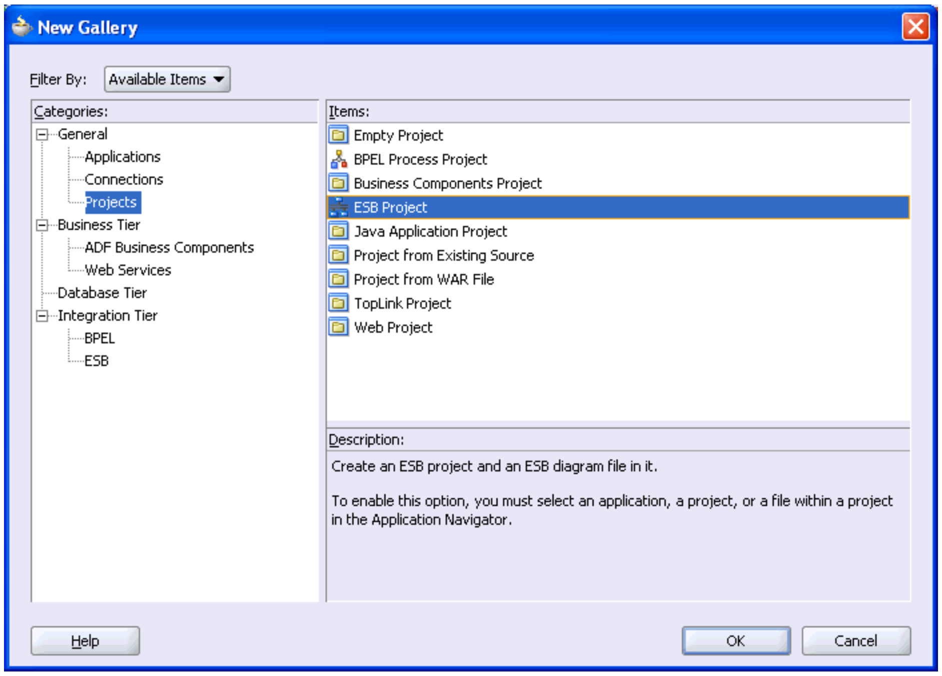
Creating an Outbound ESB Project and Assigning an Outbound WSDL File

1. At the top of the upper left pane, click the **Applications Navigator** tab.



2. Right-click an application node that you created and select **New Project**.

   The New Gallery window is displayed.



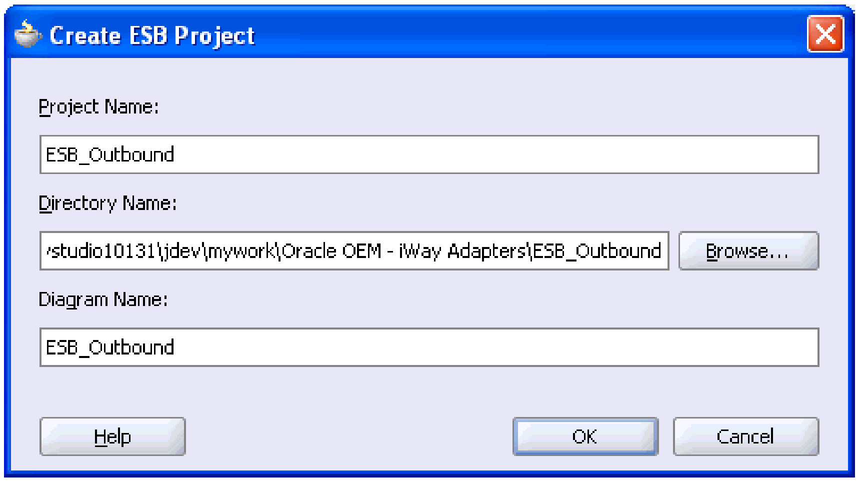
Description:

Create an ESB project and an ESB diagram file in it.

To enable this option, you must select an application, a project, or a file within a project in the Application Navigator.

3. From the Items list, select **ESB Project** and click **OK**.

   The Create ESB Project dialog box is displayed.



Project Name: ESB_Outbound

Directory Name: \studio10131\jdev\mywork\Oracle OEM - iWay Adapters\ESB_Outbound

Diagram Name: ESB_Outbound

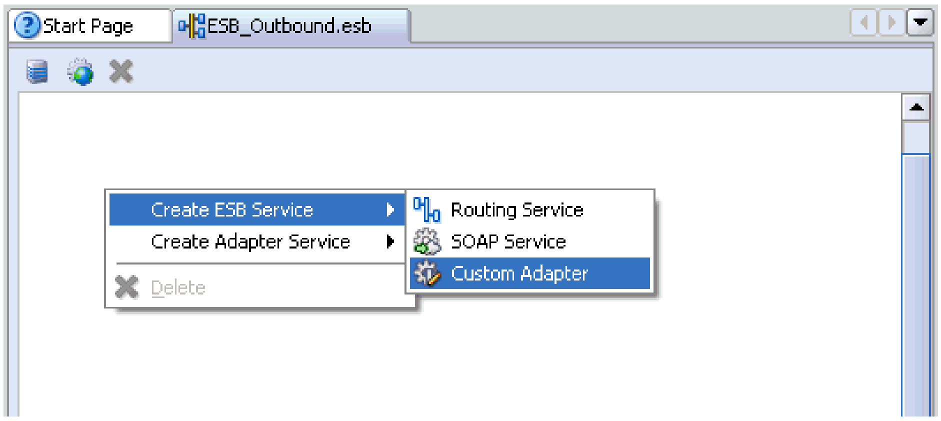
4. Perform the following steps:

   a. Specify a name for the ESB project.

      The Directory Name field and Diagram Name fields are updated automatically.

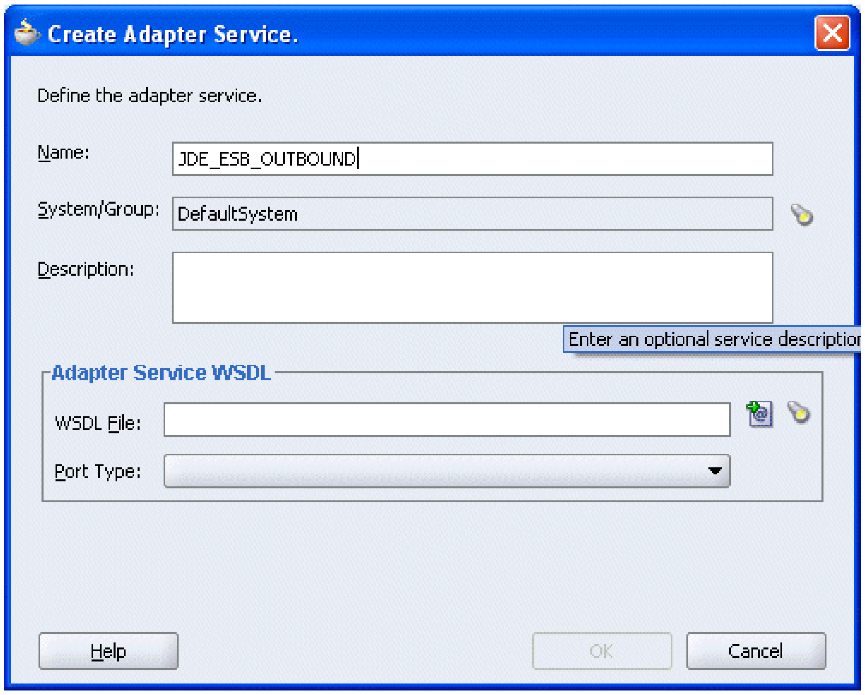
   b. Click **OK**.

   The ESB project is added at the top of the upper left pane.



5. Right-click the ESB project in the middle pane, select **Create ESB Service** followed by **Custom Adapter**.

   The Create Adapter Service dialog box is displayed.



Define the adapter service.

Name: JDE_ESB_OUTBOUND

System/Group: DefaultSystem

Description:

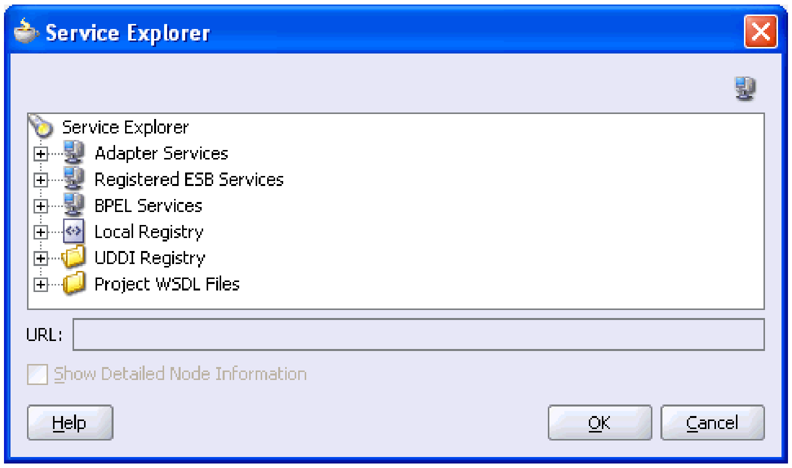
Enter an optional service descriptior

WSDL File:

Port Type:

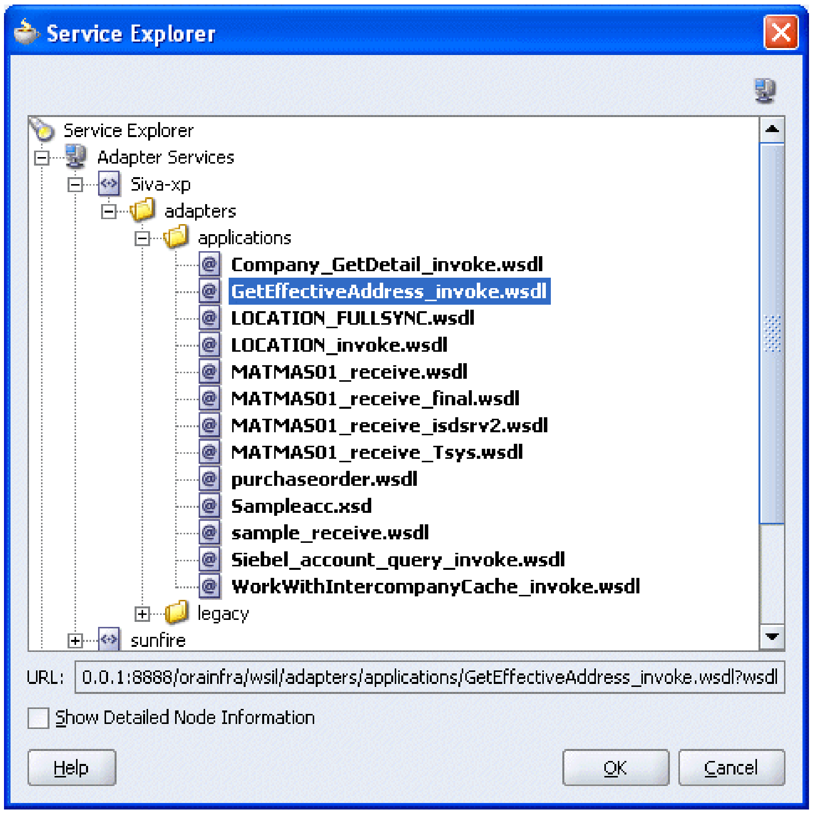
6. Enter a name for the adapter service and click the **Service Explorer** icon (second icon from the left preceding the **WSDL File** field).

The Service Explorer dialog box is displayed.



7. Expand your new connection under Adapter Services, followed by **adapters**, and then **applications**.

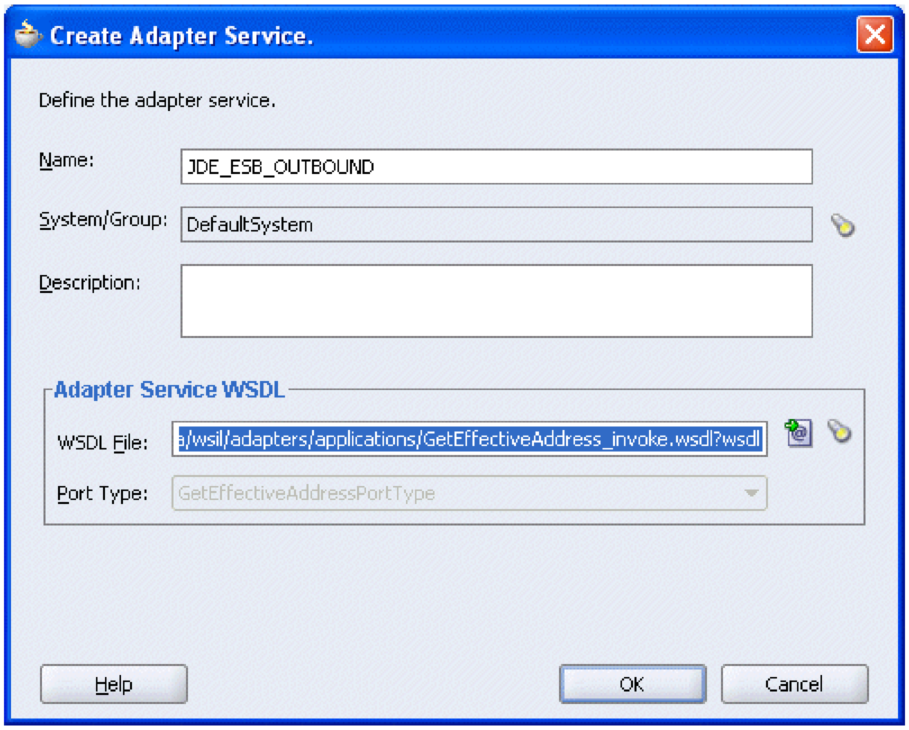
   The WSDL tree displayed in the Service Explorer dialog box lists any WSDL files you have created using Application Explorer. The WSDL tree is generated by a WSDL servlet, which is automatically deployed as part of the BPEL Server installation.


Service Explorer
Service Explorer
Adapter Services
Siva-xp
adapters
applications
Company_GetDetail_invoke.wsdl
GetEffectiveAddress_invoke.wsdl
LOCATION_FULLSYNC.wsdl
LOCATION_invoke.wsdl
MATMAS01_receive.wsdl
MATMAS01_receive_final.wsdl
MATMAS01_receive_isdsrv2.wsdl
MATMAS01_receive_Tsys.wsdl
purchaseorder.wsdl
Sampleacc.xsd
sample_receive.wsdl
Siebel_account_query_invoke.wsdl
WorkWithIntercompanyCache_invoke.wsdl
legacy
sunfire
Show Detailed Node Information
Help
OK
Cancel

URL: 0.0.1:8888/orainfra/wsil/adapters/applications/GetEffectiveAddress_invoke.wsdl?wsdl

8. Select an outbound WSDL file that has been created using Application Explorer and click **OK**.

   The **WSDL File** field in the Create Adapter Service dialog box displays the name and location of the selected WSDL file.


Create Adapter Service.

Define the adapter service.

Name: JDE_ESB_OUTBOUND

System/Group: DefaultSystem

Description:

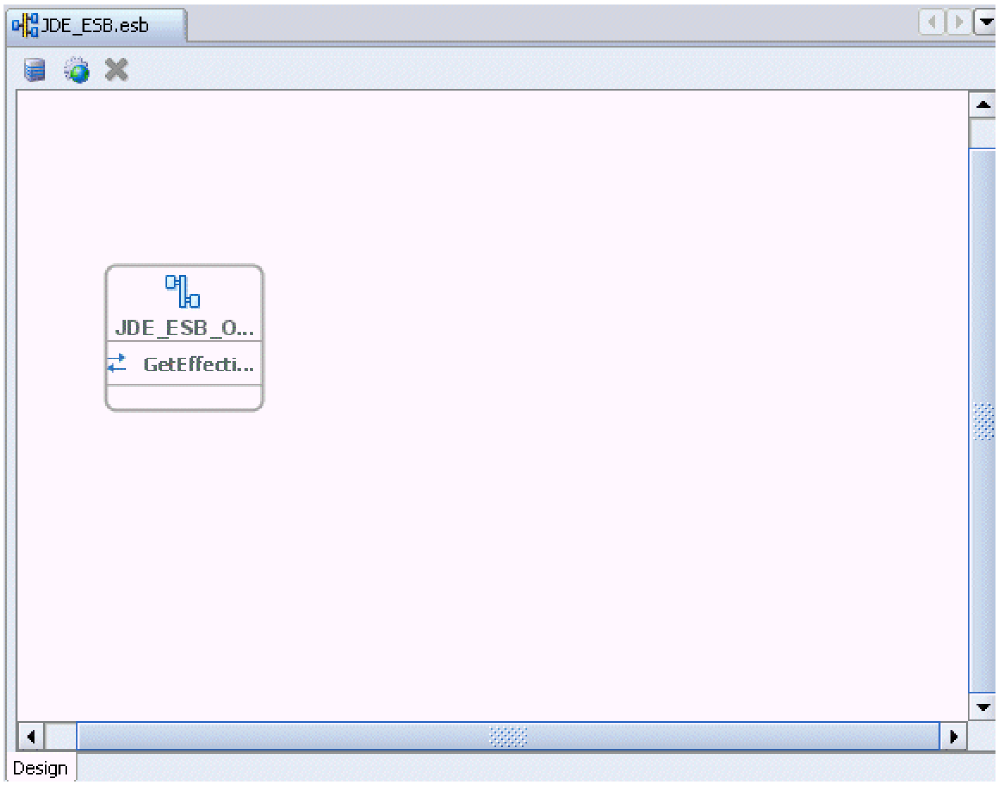
Adapter Service WSDL

WSDL File: a/wsil/adapters/applications/GetEffectiveAddress_invoke.wsdl?wsdl

Port Type: GetEffectiveAddressPortType

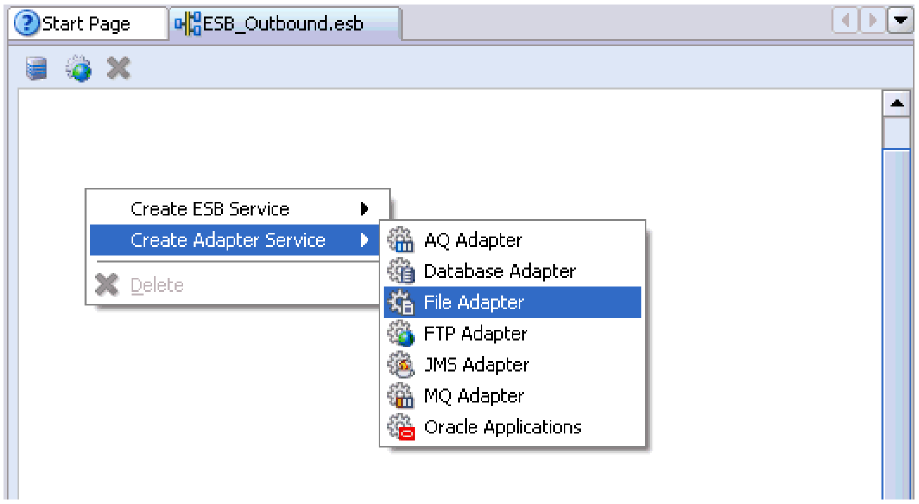
9. Click **OK**.

   The new ESB project appears in the visual editor.

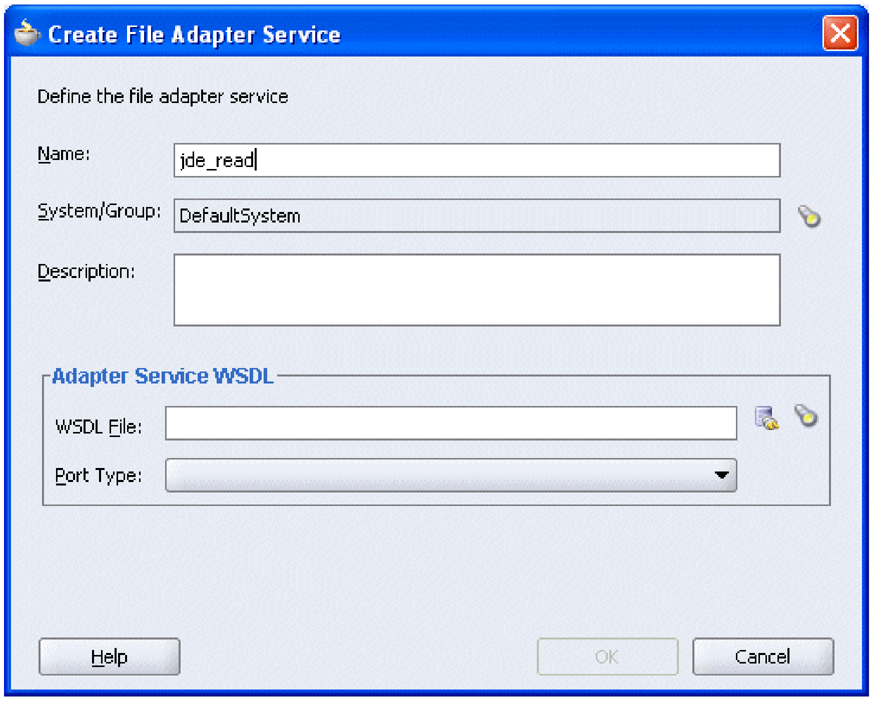


Creating a Read Process Operation Using the File Adapter

1. Right-click the ESB project in the middle pane, select **Create Adapter Service** followed by **File Adapter**.



The Create File Adapter Service dialog box is displayed.


Create File Adapter Service

Define the file adapter service

Name: jde_read

System/Group: DefaultSystem

Description:

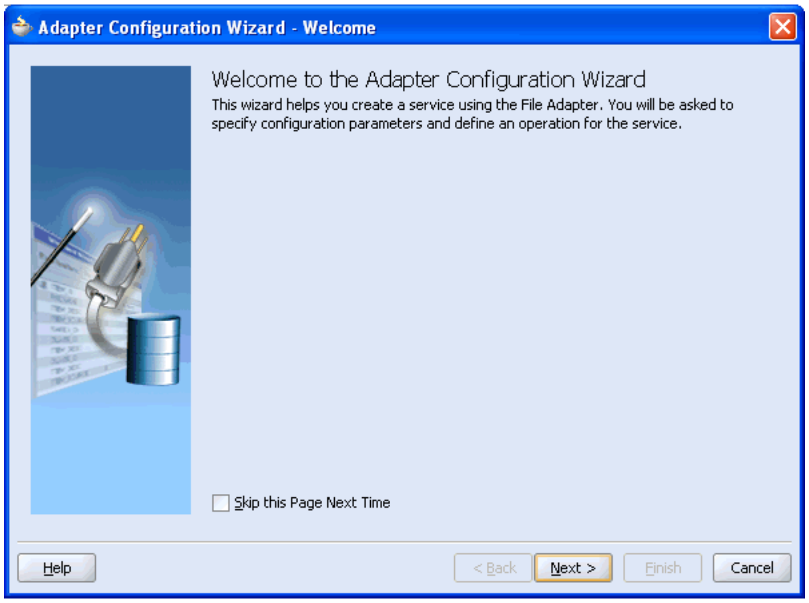
Adapter Service WSDL

WSDL File:

Port Type:

2. Enter a name for the File adapter and click the **Configure adapter service wsdl** icon next to the **WSDL File** field.

   The Adapter Configuration Wizard - Welcome window is displayed.



Welcome to the Adapter Configuration Wizard

This wizard helps you create a service using the File Adapter. You will be asked to specify configuration parameters and define an operation for the service.

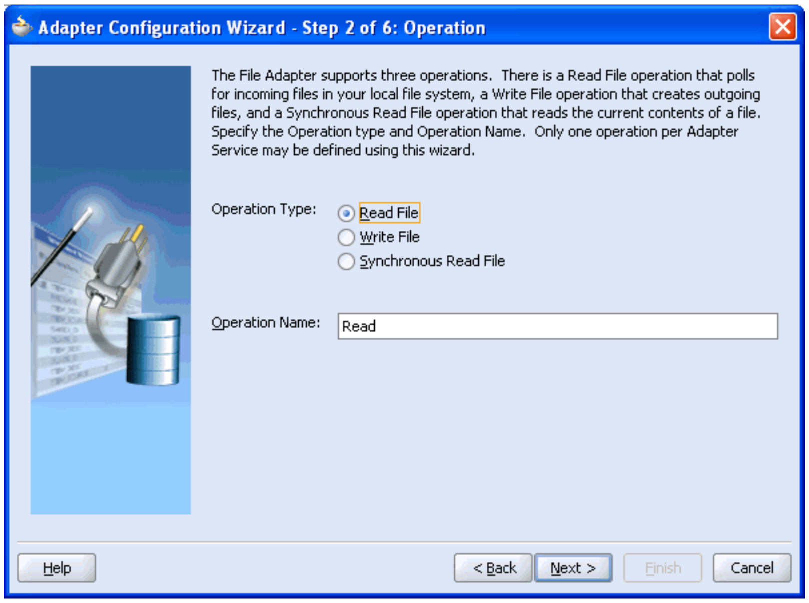
☐ Skip this Page Next Time

3. Click **Next**.

   The Adapter Configuration Wizard - Step 1 of 6: Service Name window is displayed.

4. Click **Next**.

   The Adapter Configuration Wizard - Step 2 of 6: Operation window is displayed.



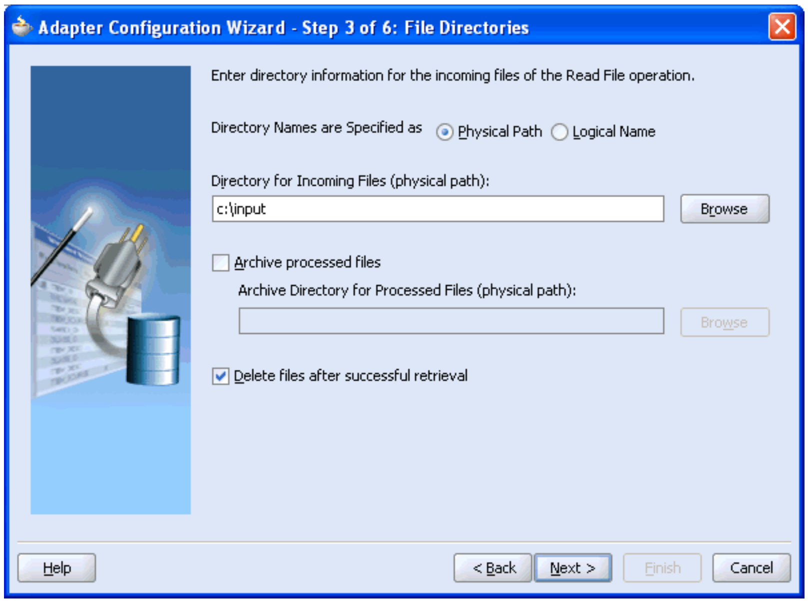
The File Adapter supports three operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, and a Synchronous Read File operation that reads the current contents of a file. Specify the Operation type and Operation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

Operation Type: [✓ Read File] Write File Synchronous Read File

Operation Name: Read

5. Click **Read File** as the Operation Type and click **Next**.

   The Adapter Configuration Wizard - Step 3 of 6: File Directories window is displayed.


Adapter Configuration Wizard - Step 3 of 6: File Directories
Browse
Help
< Back
Next >
Finish
Cancel

Enter directory information for the incoming files of the Read File operation.

Directory Names are Specified as [✓ Physical Path] Logical Name

Directory for Incoming Files (physical path): c:\input

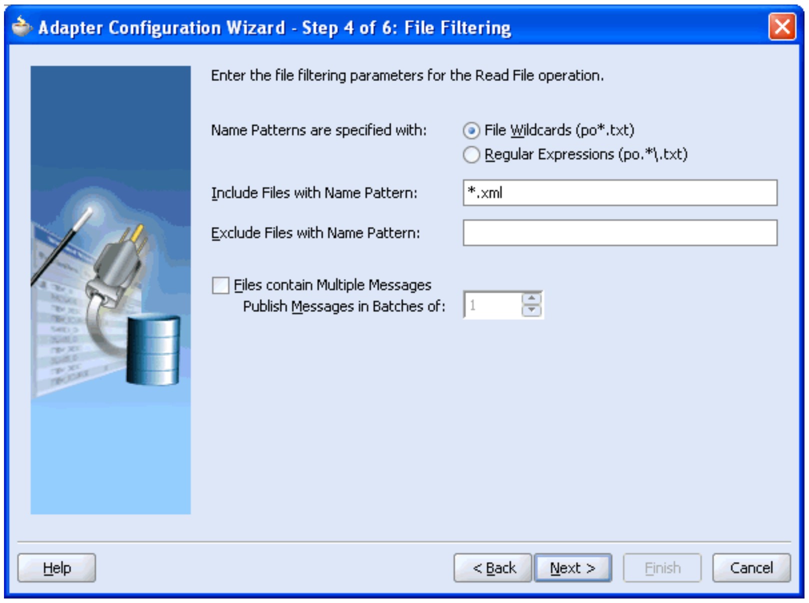
☐ Archive processed files

Archive Directory for Processed Files (physical path):

☑ Delete files after successful retrieval

6. Enter the path of the input directory where you are placing the incoming XML file and click **Next**.

   The Adapter Configuration Wizard - Step 4 of 6: File Filtering window is displayed.



Enter the file filtering parameters for the Read File operation.

Name Patterns are specified with: [✓ File Wildcards (po*.txt)] Regular Expressions (po.*\.txt)

Include Files with Name Pattern: *.xml

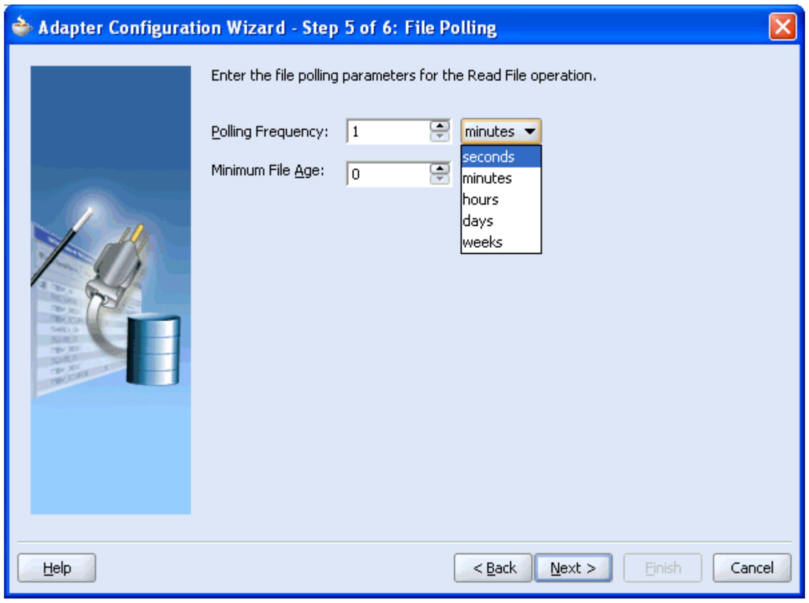
Exclude Files with Name Pattern:

☐ Files contain Multiple Messages

Publish Messages in Batches of: 1

7. Enter the input file extension, for example ***.xml**, and click **Next**.

   The Adapter Configuration Wizard - Step 5 of 6: File Polling window is displayed.


Adapter Configuration Wizard - Step 5 of 6: File Polling

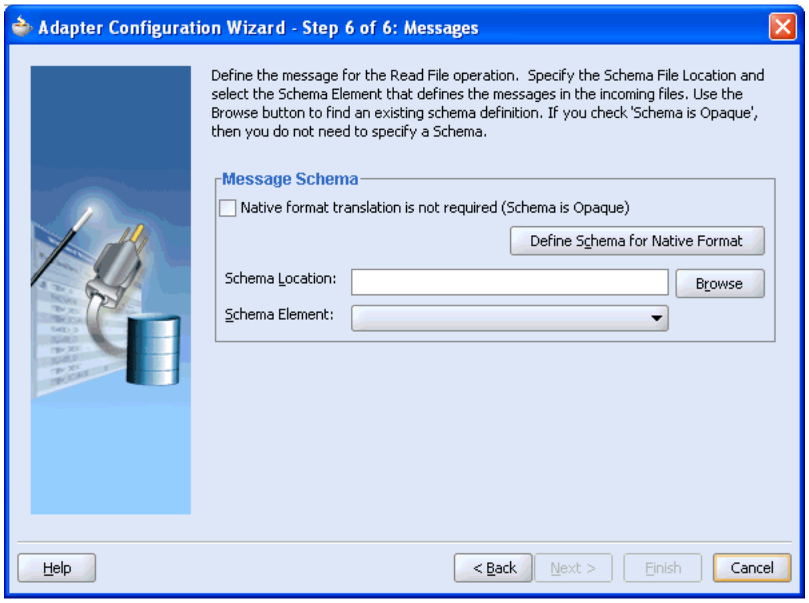
Enter the file polling parameters for the Read File operation.

Polling Frequency: 1 minutes

Minimum File Age: 0

8. Change the Polling Frequency to seconds and click **Next**.

   The Adapter Configuration Wizard - Step 6 of 6: Messages window is displayed.


Adapter Configuration Wizard - Step 6 of 6: Messages

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

Message Schema

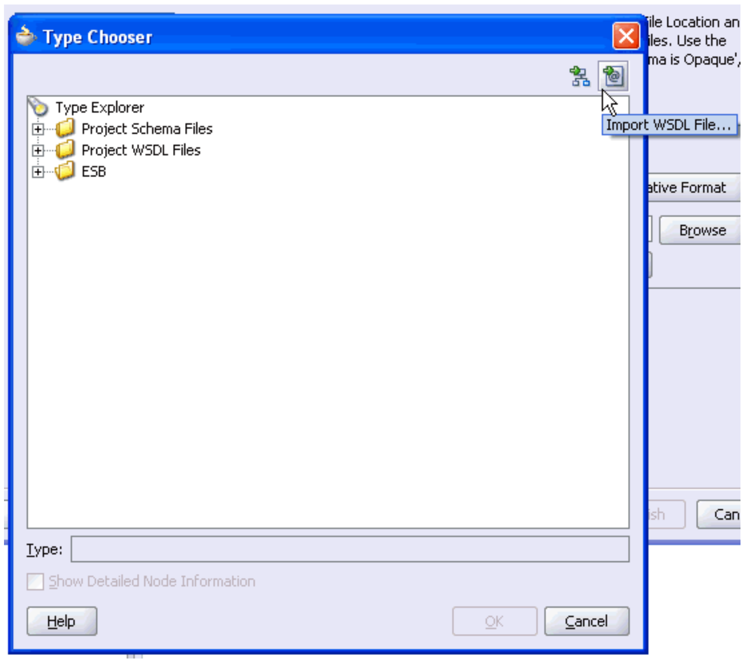
☐ Native format translation is not required (Schema is Opaque)

Schema Location:

Schema Element:

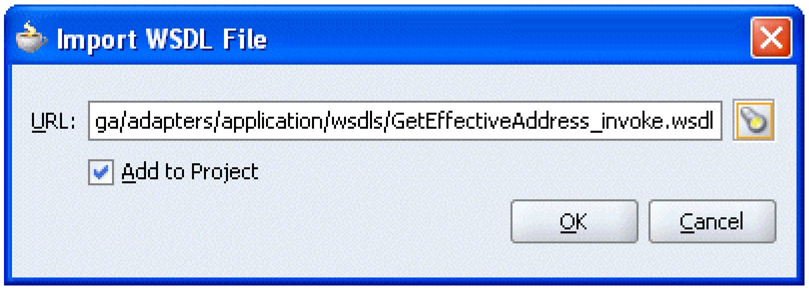
9. Click **Browse** to select the WSDL.

   The Type Chooser window is displayed.



10. Click the **Import WSDL File** icon on the upper right corner of the dialog box.

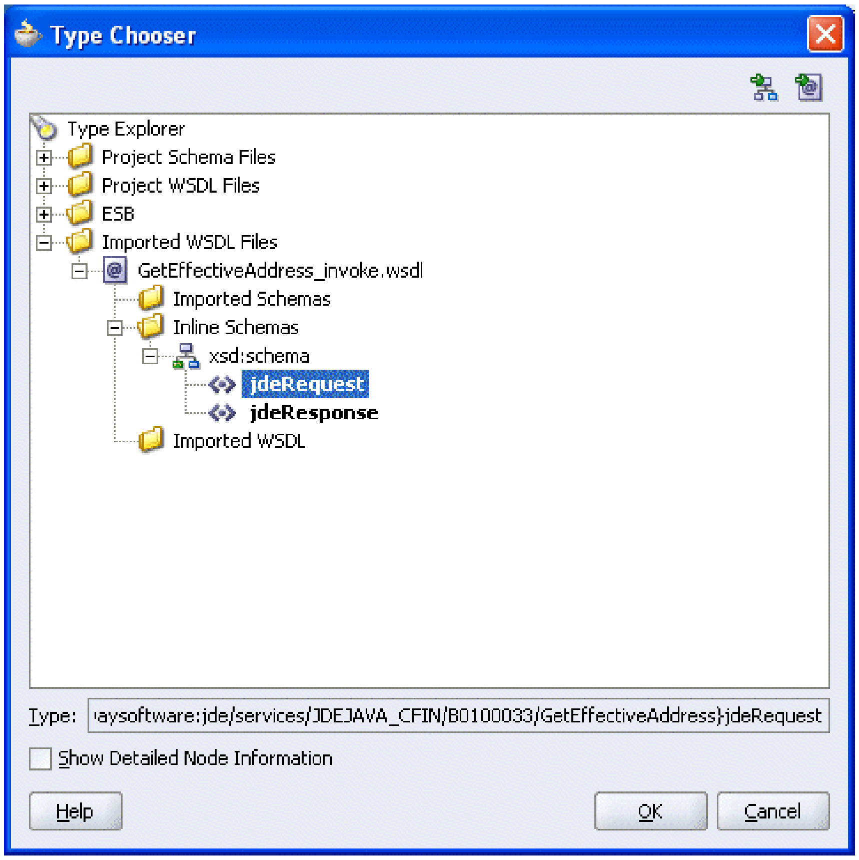
    The Import WSDL File dialog box is displayed.



URL: ga/adapters/application/wsdls/GetEffectiveAddress_invoke.wsdl

11. Select the WSDL file and click **OK**.

    The Imported WSDL Files folder is added.

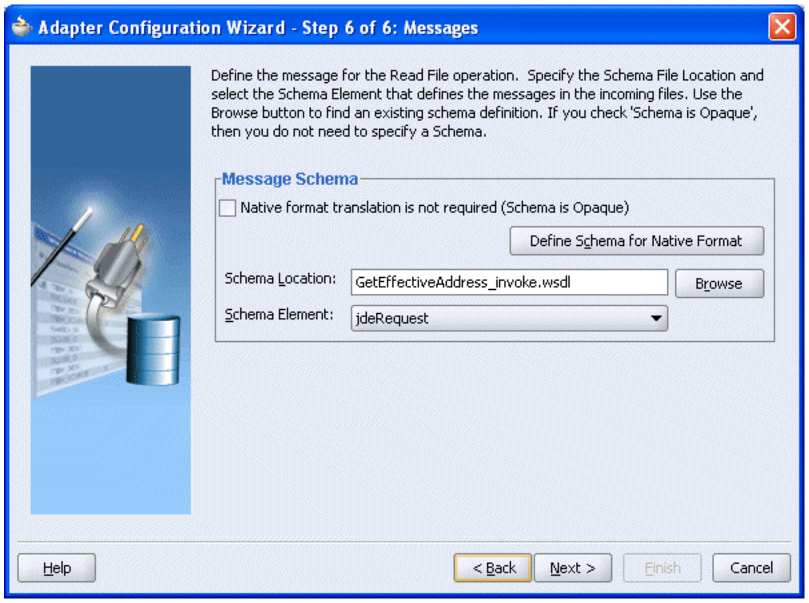


Type: ›aysoftware:jde/services/JDEJAVA_CFIN/B0100033/GetEffectiveAddress}jdeRequest

Show Detailed Node Information

12. Expand the Imported WSDL Files folder, select an Inline Schema, for example, **PS8**, and click **OK**.

    You are returned to the Adapter Configuration Wizard - Step 6 of 6: Messages window.


Adapter Configuration Wizard - Step 6 of 6: Messages
Message Schema
Define Schema for Native Format
Browse
Help
< Back
Next >
Finish
Cancel

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

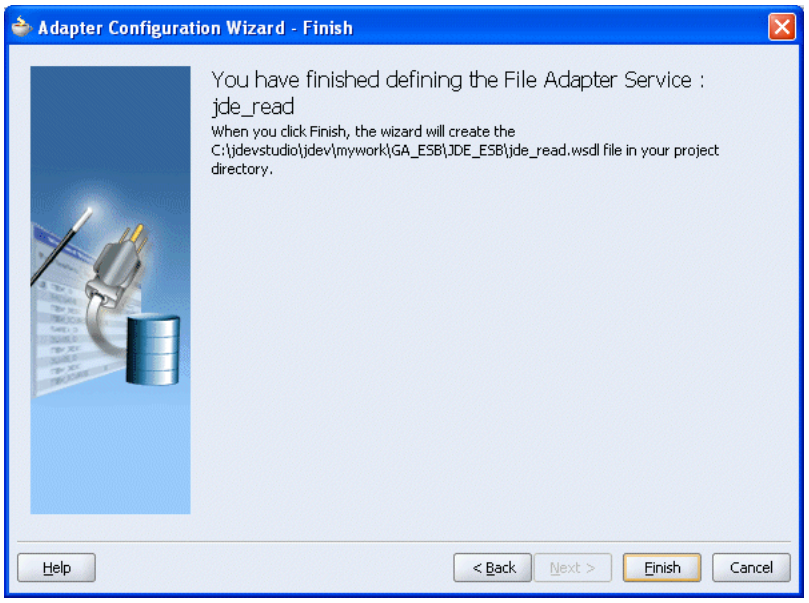
Native format translation is not required (Schema is Opaque)

Schema Location: GetEffectiveAddress_invoke.wsdl

Schema Element: jdeRequest

13. Click **Next**.

    The Adapter Configuration Wizard - Finish window is displayed.

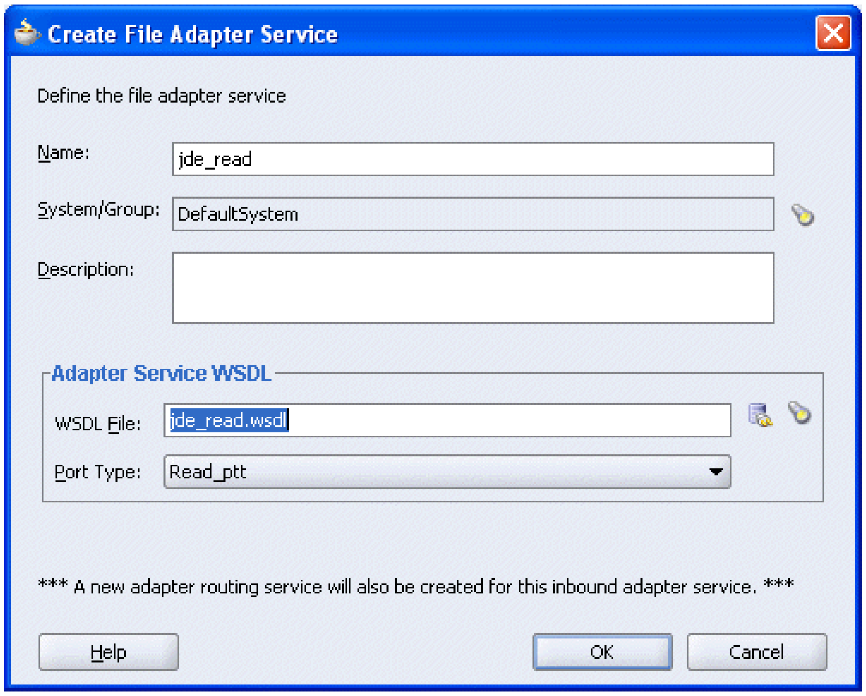

Adapter Configuration Wizard - Finish

You have finished defining the File Adapter Service : jde_read

When you click Finish, the wizard will create the C:\jdevstudio\jdev\mywork\GA_ESB\JDE_ESB\jde_read.wsdl file in your project directory.

14. Click **Finish**.

    You are returned to the Create File Adapter Service dialog box.


Create File Adapter Service

Define the file adapter service

Name: jde_read

System/Group: DefaultSystem

Description:

Adapter Service WSDL

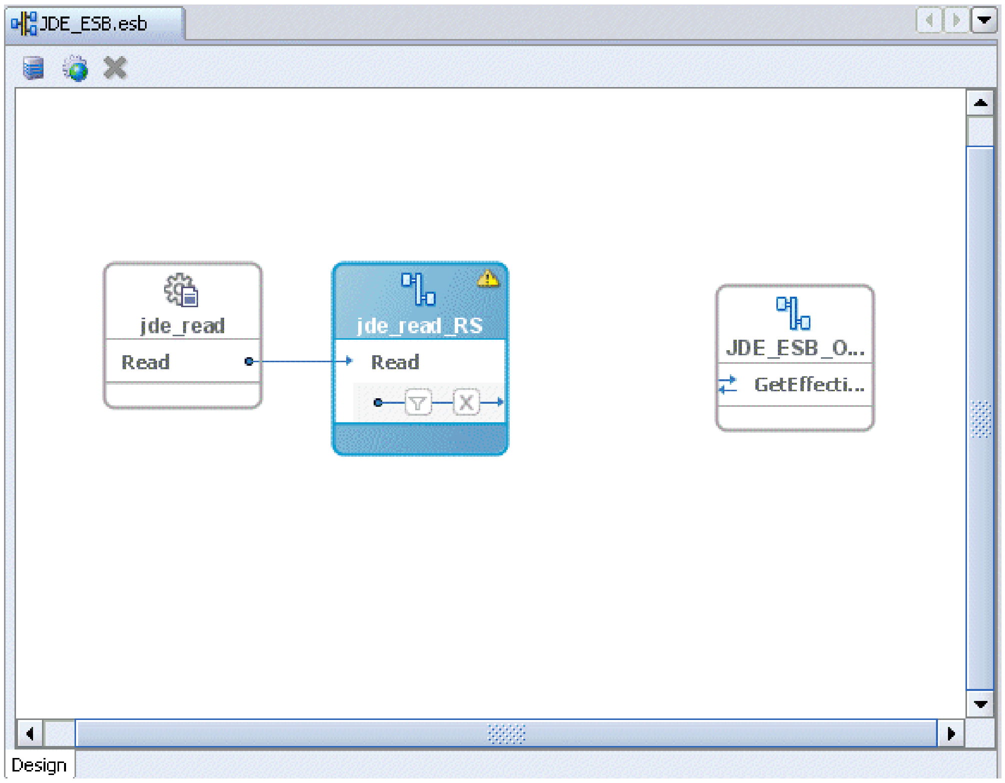
WSDL File: jde_read.wsdl

Port Type: Read_ptt

*** A new adapter routing service will also be created for this inbound adapter service. ***

15. Click **OK**.

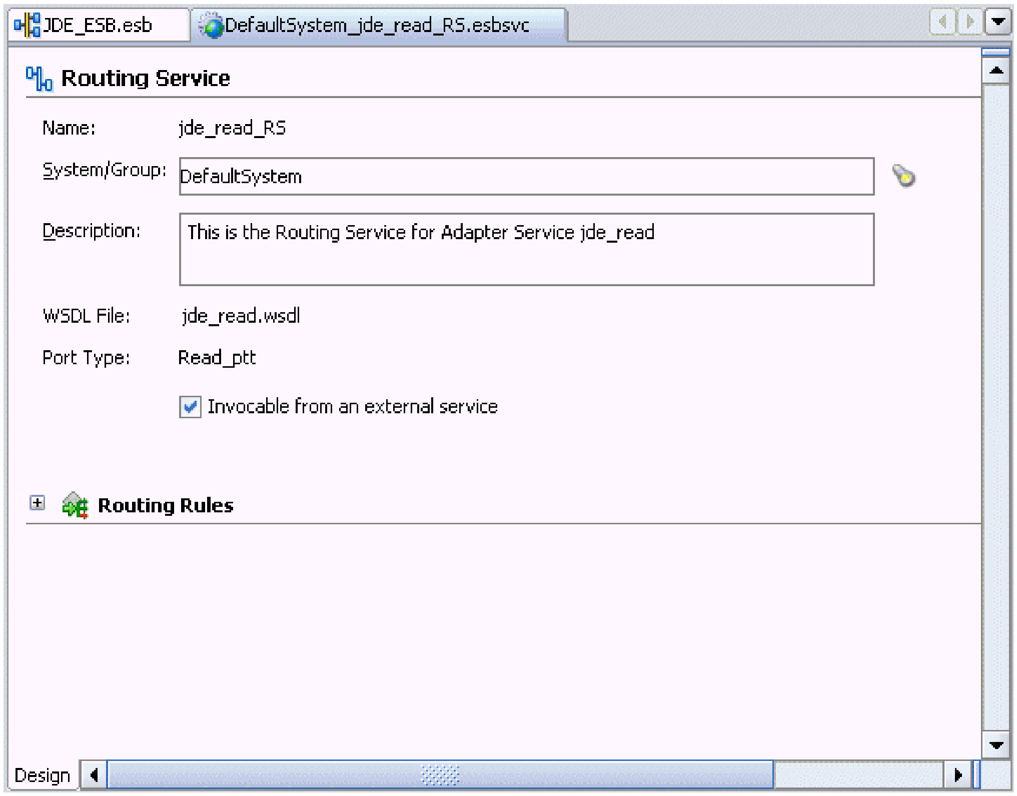
    The Read operation with a routing service is added to the ESB outbound project view.



Providing a Routing Service for the Read Operation

1. Double-click the routing service.

   The Routing Service window is displayed.



Name: jde_read_RS

System/Group: DefaultSystem

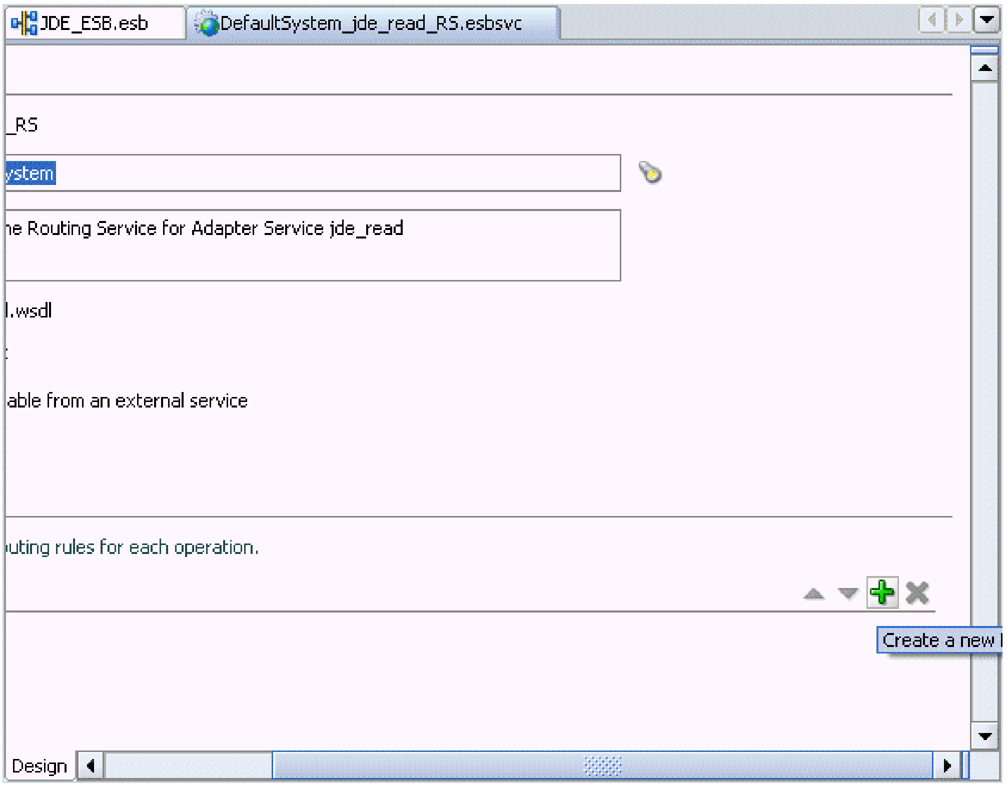
Description: This is the Routing Service for Adapter Service jde_read

WSDL File: jde_read.wsdl

Port Type: Read_ptt

[✓] Invocable from an external service

2. Expand the **Routing Rules**.



_RS

ystem

ne Routing Service for Adapter Service jde_read

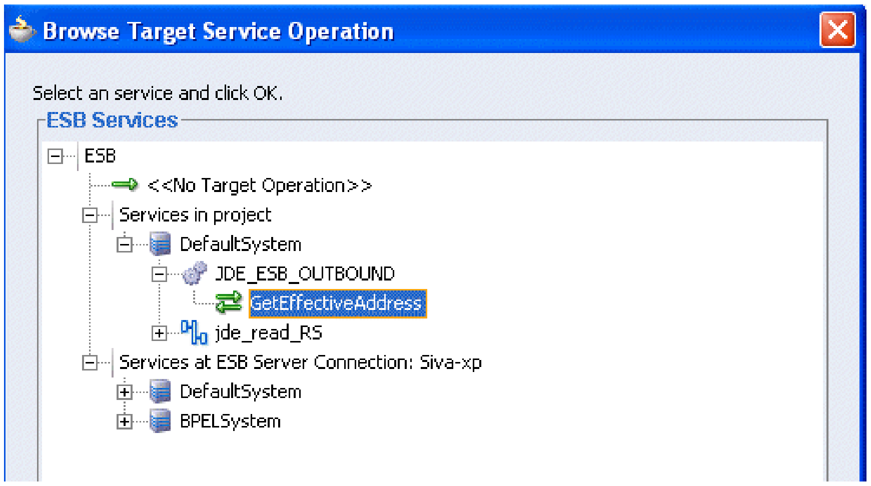
l.wsdl

able from an external service

uting rules for each operation.

3. Click the green plus sign icon, which represents the option to **Create a new Routing Rule**.

   The Browse Target Service Operation window is displayed.



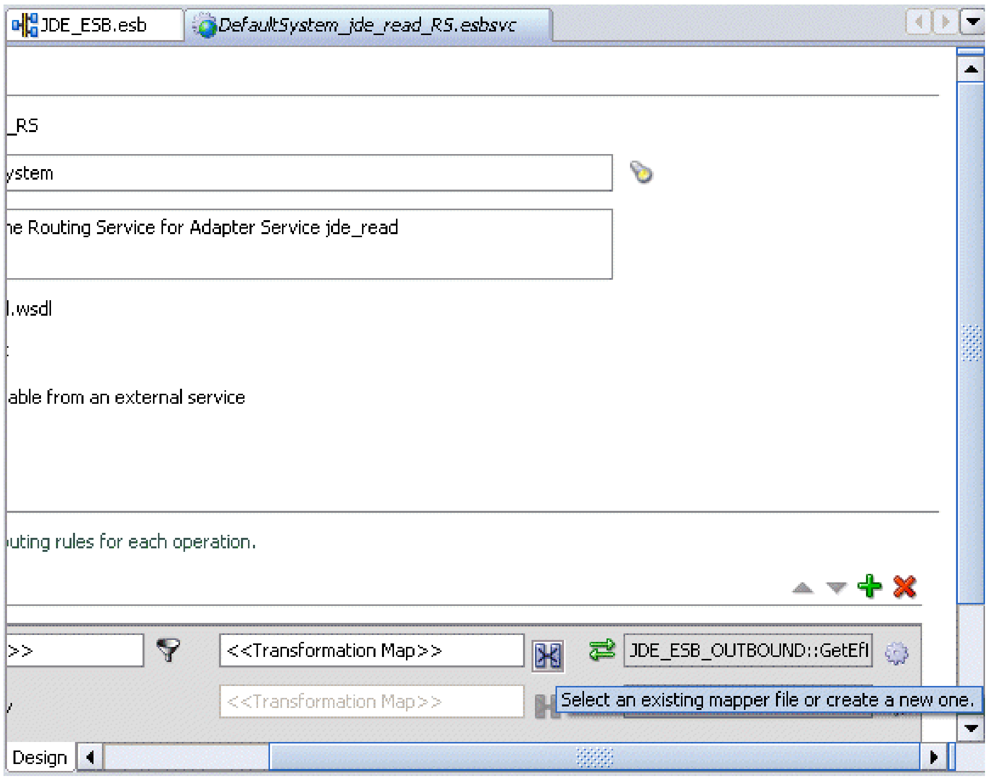
Select an service and click OK.

ESB Services

```
ESB
  <<No Target Operation>>
  Services in project
    DefaultSystem
      JDE_ESB_OUTBOUND
        GetEffectiveAddress
      jde_read_RS
  Services at ESB Server Connection: Siva-xp
    DefaultSystem
    BPELSystem
```

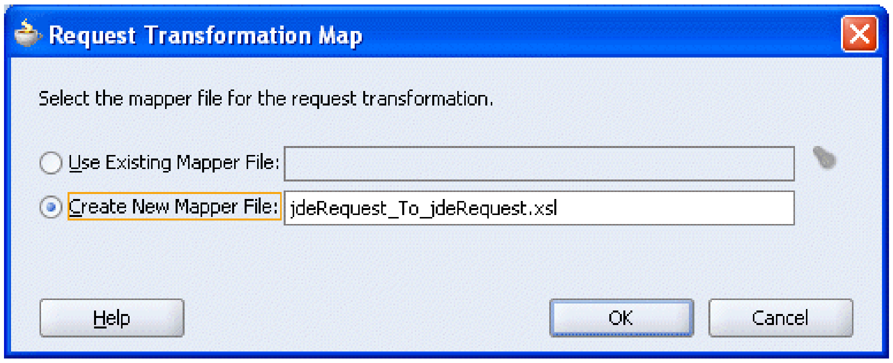
4. Expand **Services in project**, **Default System**, your adapter service node, for example, **JDE_ESB_Outbound**, and select the service name, for example, **GetDetail**.
5. Click **OK**.

   You are returned to the Routing Rules window.


JDE_ESB.esb
DefaultSystem_jde_read_RS.esbsvc
_RS
ystem
ne Routing Service for Adapter Service jde_read
l.wsdl
able from an external service
uting rules for each operation.
>>
<<Transformation Map>>
JDE_ESB_OUTBOUND::GetEfl
<<Transformation Map>>
Select an existing mapper file or create a new one.
Design

6. Click on the icon next to the <<Transformation Map>> field (**Select an existing mapper file or create a new one**).

   The Request Transformation Map dialog box is displayed.



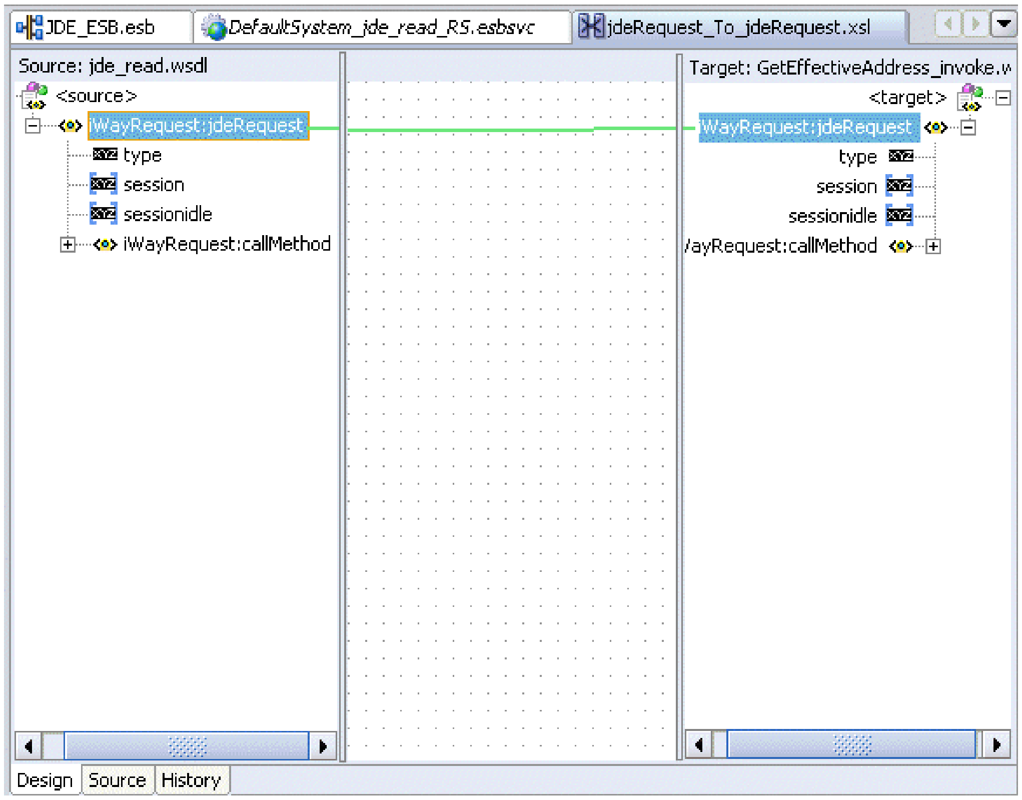
Select the mapper file for the request transformation.

Use Existing Mapper File:

[✓ Create New Mapper File:] jdeRequest_To_jdeRequest.xsl

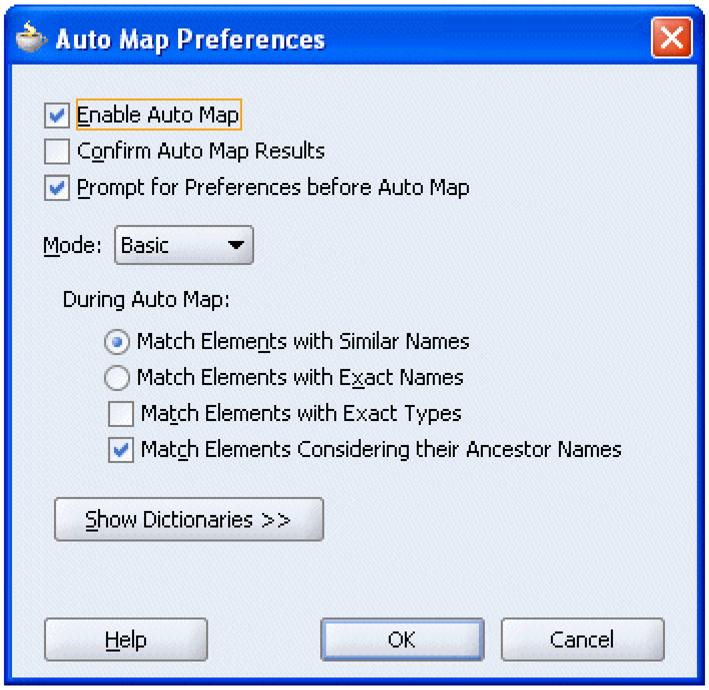
7. Select the **Create New Mapper File** option, specify the file name, and click **OK**.

   The following mapping window is displayed.



8. Select the WSDL file and map it to the Write operation.

   Once you map the WSDL file, the Auto Map Preferences dialog box is displayed.



[✓] Enable Auto Map
[ ] Confirm Auto Map Results
[✓] Prompt for Preferences before Auto Map

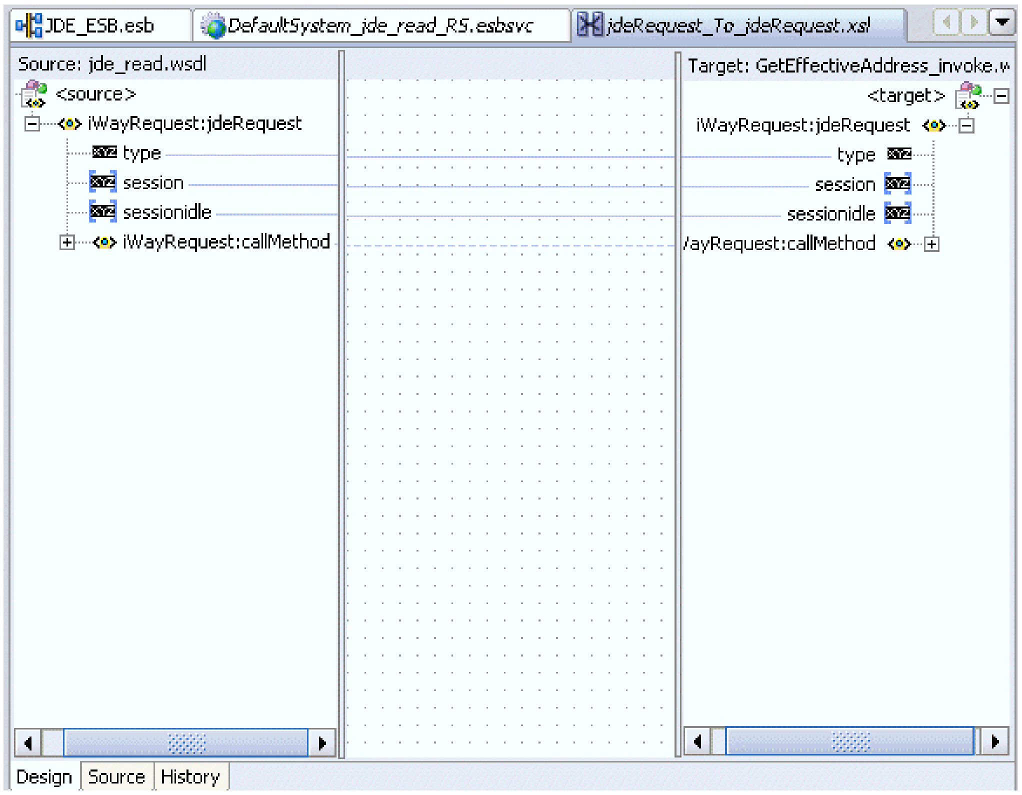
Mode: Basic

During Auto Map:

(●) Match Elements with Similar Names
( ) Match Elements with Exact Names
[ ] Match Elements with Exact Types
[✓] Match Elements Considering their Ancestor Names

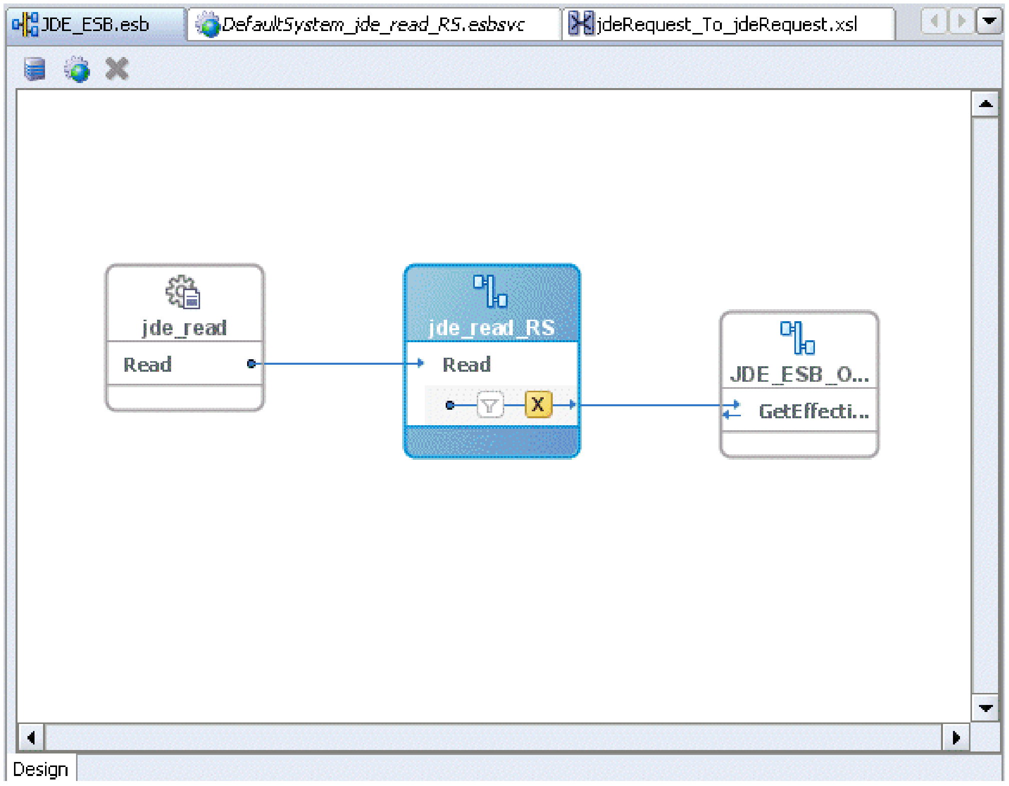
9. Click **OK**.

   The mapping is completed as shown in the following window.



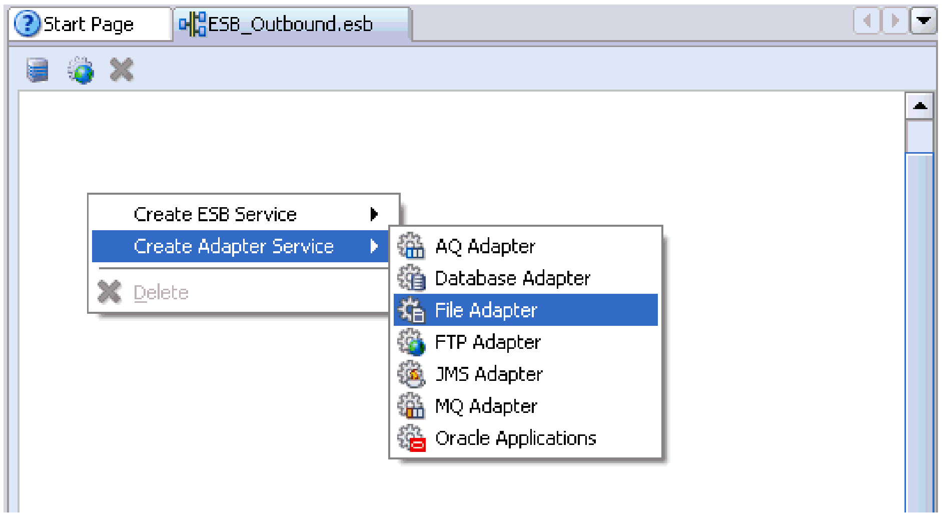
10. Double-click the ESB outbound project file in the left pane, for example, **ESB_ Outbound.esb**.

    Notice that the Routing service is now created for the Read operation.

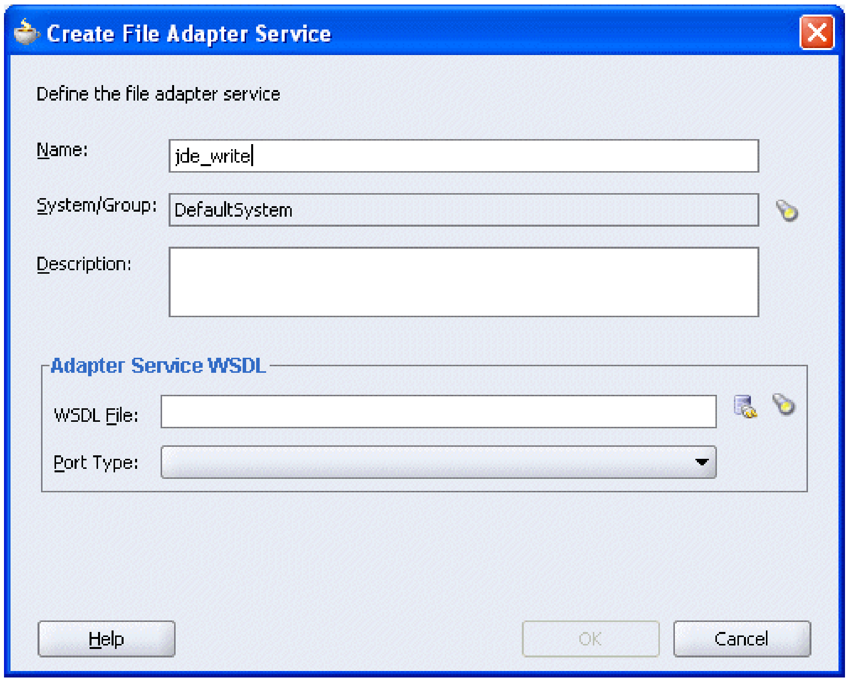


Creating a Write Process Operation Using the File Adapter

1. Right-click the ESB project in the middle pane, select **Create Adapter Service** followed by **File Adapter**.



   The Create File Adapter Service dialog box is displayed.


Create File Adapter Service

Define the file adapter service

Name: jde_write

System/Group: DefaultSystem

Description:

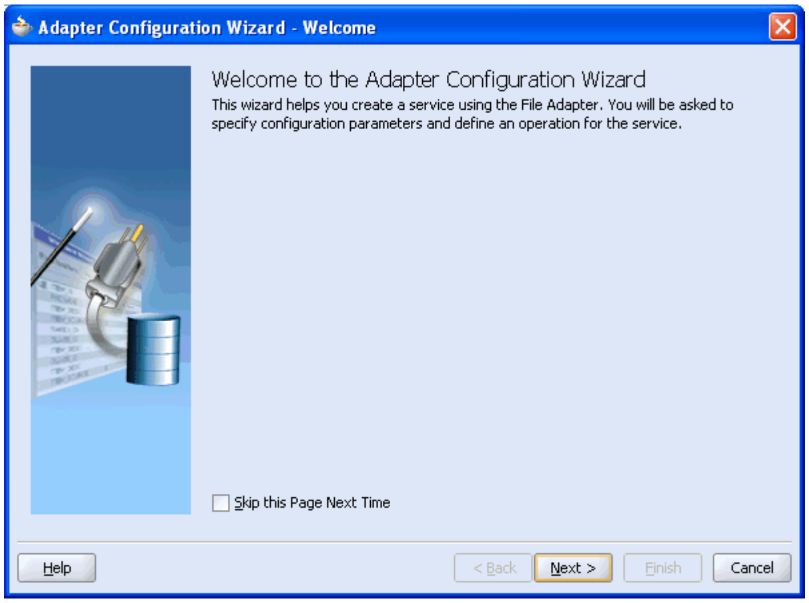
Adapter Service WSDL

WSDL File:

Port Type:

2. Enter a name for the File adapter and click the **Configure adapter service wsdl** icon next to the **WSDL File** field.

   The Adapter Configuration Wizard - Welcome window is displayed.



Welcome to the Adapter Configuration Wizard

This wizard helps you create a service using the File Adapter. You will be asked to specify configuration parameters and define an operation for the service.

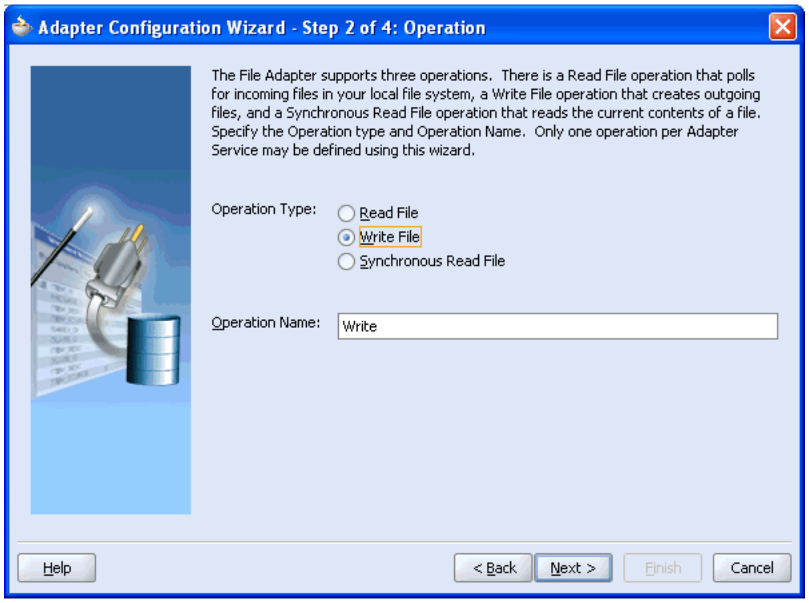
Skip this Page Next Time

3. Click **Next**.

   The Adapter Configuration Wizard - Step 1 of 4: Service Name window is displayed.

4. Click **Next**.

   The Adapter Configuration Wizard - Step 2 of 4: Operation window is displayed.



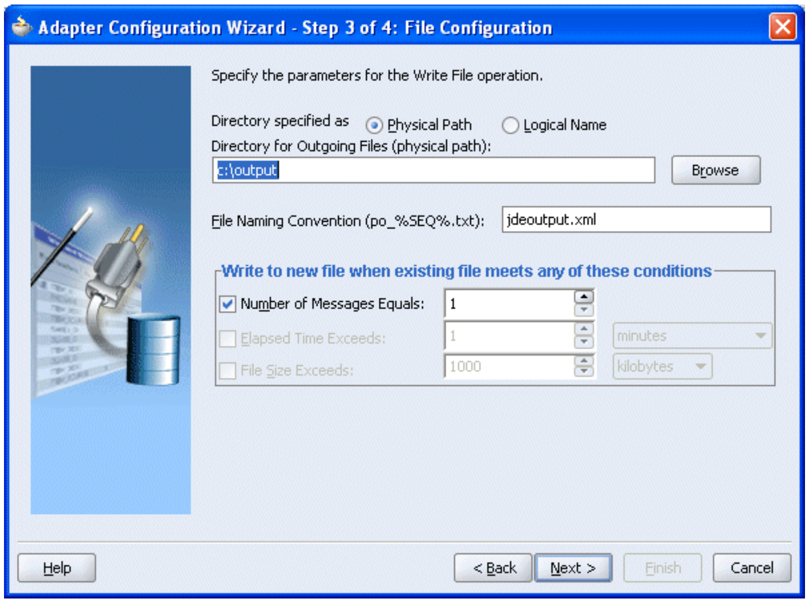
The File Adapter supports three operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, and a Synchronous Read File operation that reads the current contents of a file. Specify the Operation type and Operation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

Operation Type: Read File [✓ Write File] Synchronous Read File

Operation Name: Write

5. Click **Write File** as the Operation Type and click **Next**.

   The Adapter Configuration Wizard - Step 3 of 4: File Configuration window is displayed.


Adapter Configuration Wizard - Step 3 of 4: File Configuration

Specify the parameters for the Write File operation.

Directory specified as [✓ Physical Path] Logical Name

Directory for Outgoing Files (physical path): c:\output

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

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

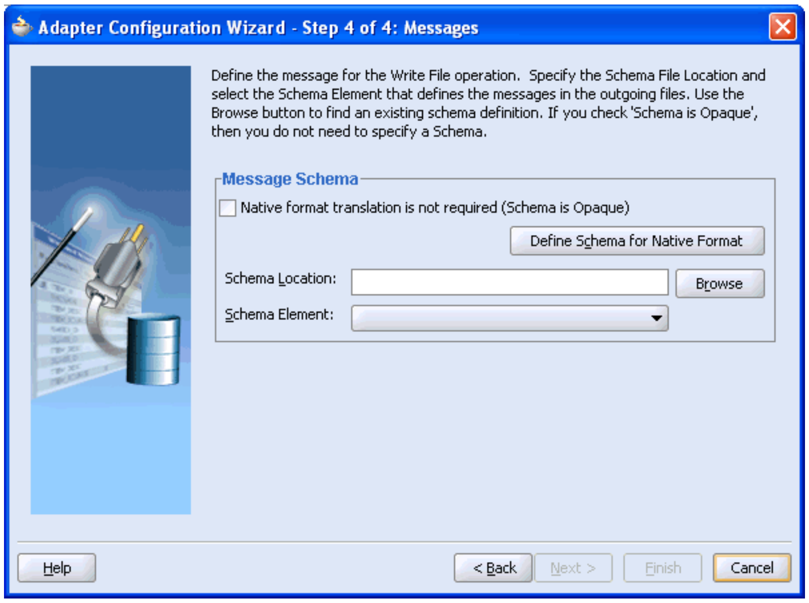
[✓] Number of Messages Equals: 1

[ ] Elapsed Time Exceeds: 1 minutes

[ ] File Size Exceeds: 1000 kilobytes

6. Enter the path of the output directory and name of the output file and click **Next**.

   The Adapter Configuration Wizard - Step 4 of 4: Messages window is displayed.


Adapter Configuration Wizard - Step 4 of 4: 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

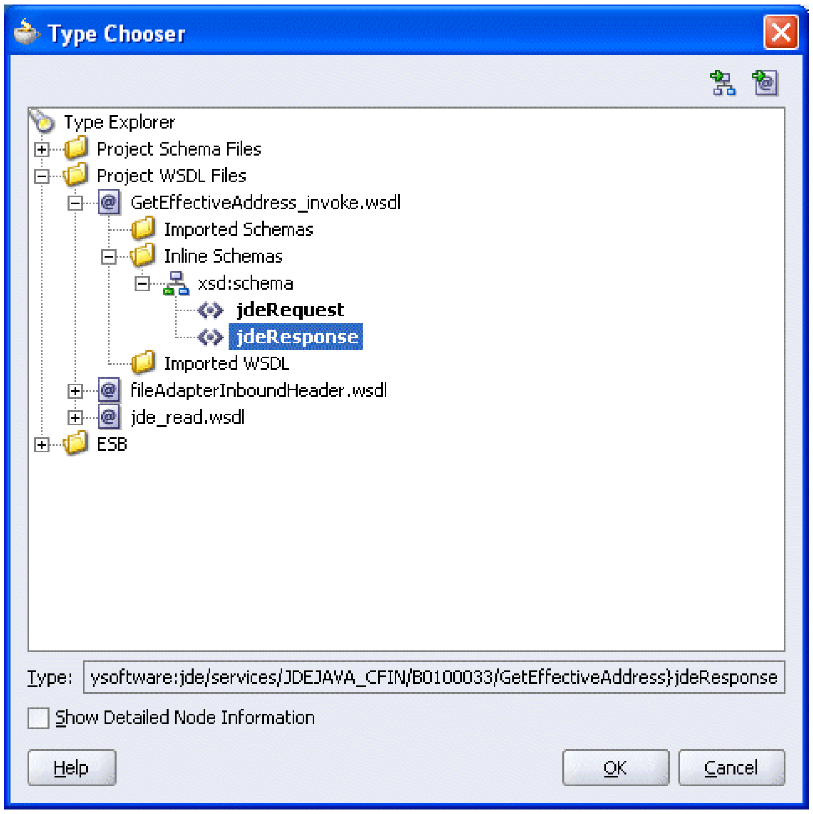
[ ] Native format translation is not required (Schema is Opaque)

Schema Location:

Schema Element:

7. Click **Browse** to select the WSDL.

   The Type Chooser window is displayed.

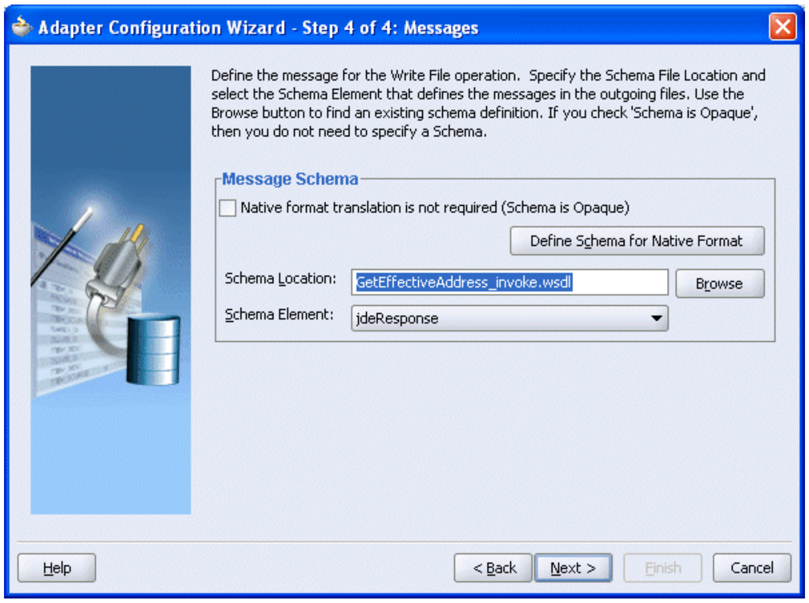


Type: ysoftware:jde/services/JDEJAVA_CFIN/B0100033/GetEffectiveAddress}jdeResponse

Show Detailed Node Information

8. Expand the **Project WSDL Files** folder, select an Inline Schema and click **OK**.

   You are returned to the Adapter Configuration Wizard - Step 4 of 4: Messages window.


Adapter Configuration Wizard - Step 4 of 4: Messages
Message Schema
Define Schema for Native Format
Browse
Help
< Back
Next >
Finish
Cancel

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.

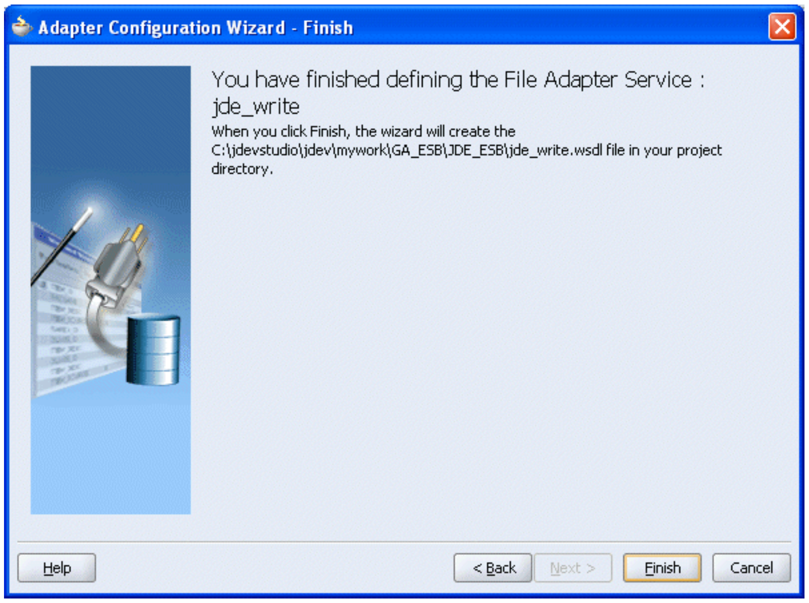
Native format translation is not required (Schema is Opaque)

Schema Location: GetEffectiveAddress_invoke.wsdl

Schema Element: jdeResponse

9. Click **Next**.

   The Adapter Configuration Wizard - Finish window is displayed.

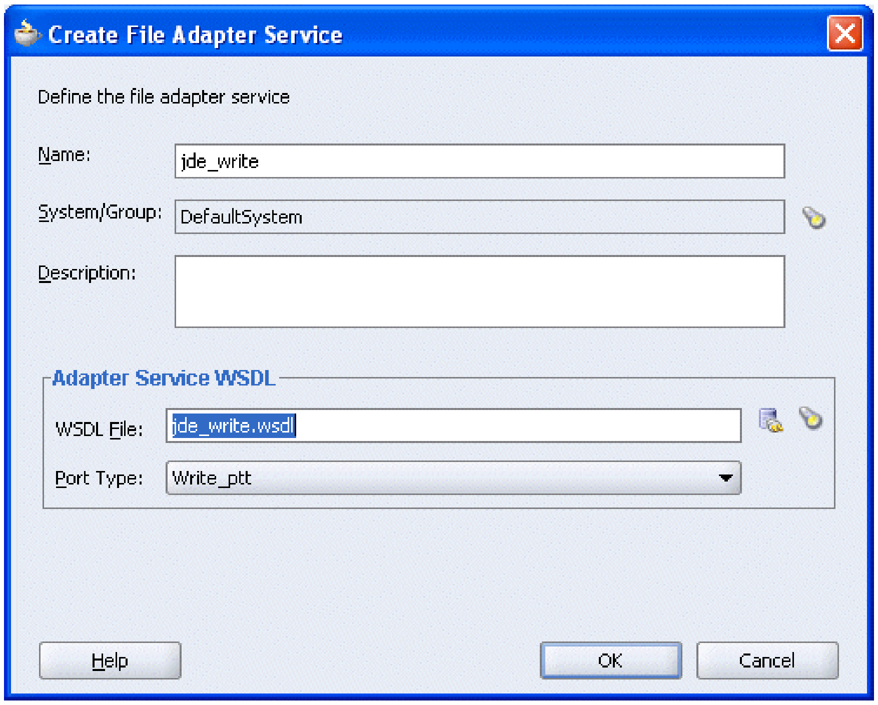


You have finished defining the File Adapter Service : jde_write

When you click Finish, the wizard will create the C:\jdevstudio\jdev\mywork\GA_ESB\JDE_ESB\jde_write.wsdl file in your project directory.

10. Click **Finish**.

    You are returned to the Create File Adapter Service dialog box.



Define the file adapter service

Name: jde_write

System/Group: DefaultSystem

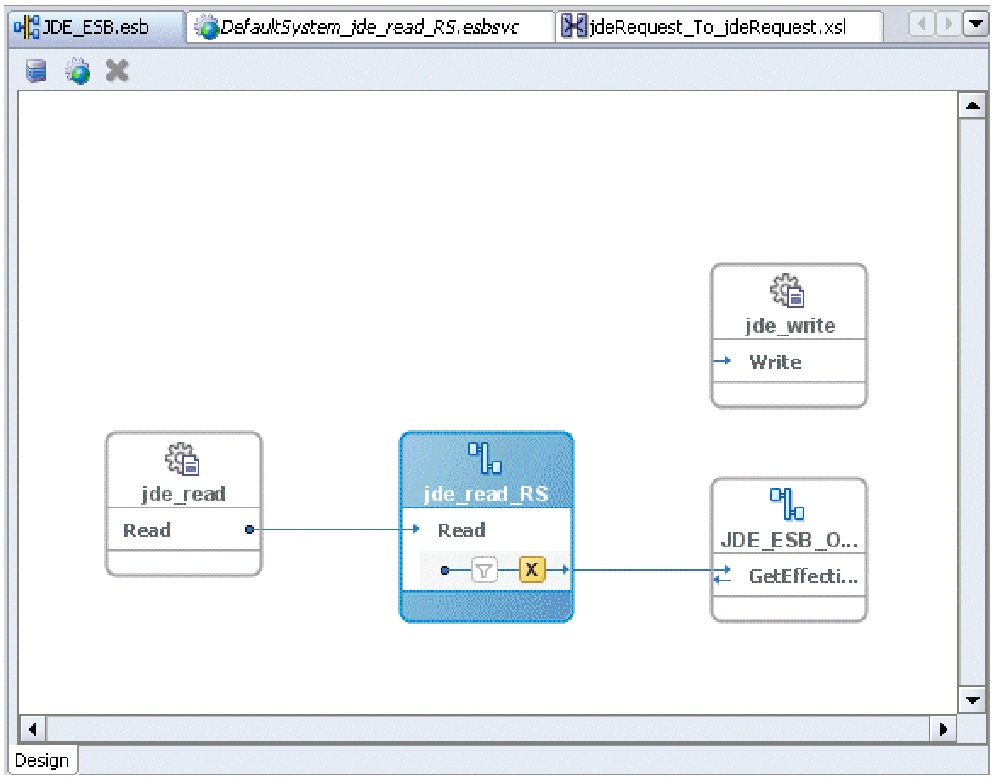
Description:

WSDL File: jde_write.wsdl

Port Type: Write_ptt

11. Click **OK**.

    The Write operation is added to the ESB outbound project view.

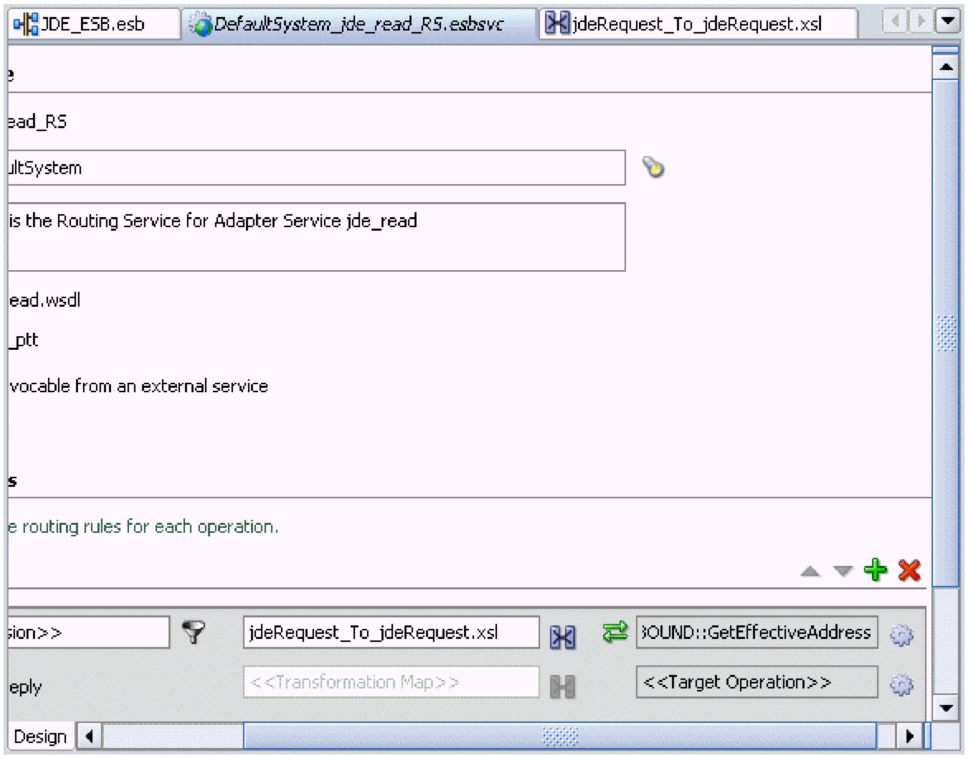


Providing a Routing Service for the Write Operation

1. Double-click the routing service.

   The Routing Service window is displayed.

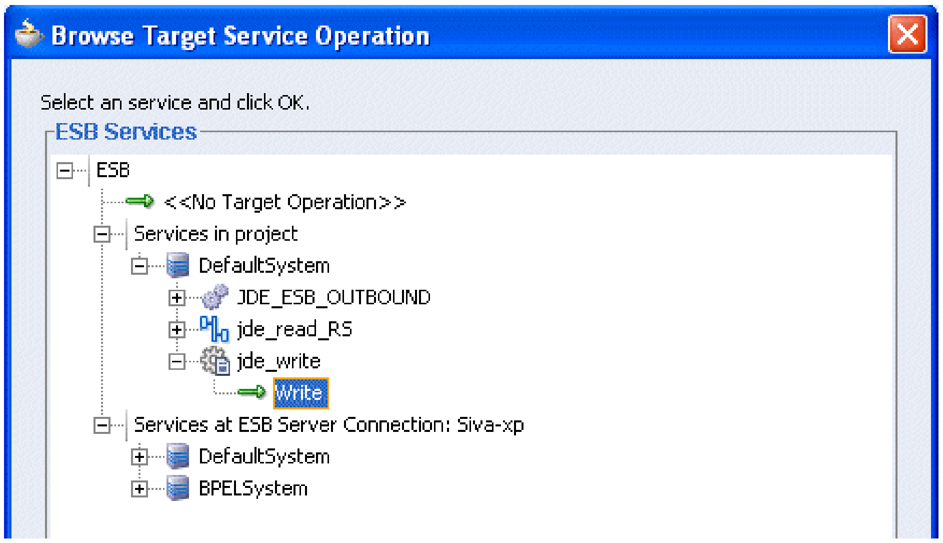
2. Expand the **Routing Rules**.


JDE_ESB.esb
DefaultSystem_jde_read_RS.esbsvc
jdeRequest_To_jdeRequest.xsl
Design

e
ead_RS
ultSystem
is the Routing Service for Adapter Service jde_read
ead.wsdl
_ptt
vocable from an external service
s
e routing rules for each operation.
sion>> jdeRequest_To_jdeRequest.xsl OUND::GetEffectiveAddress
eply <<Transformation Map>> <<Target Operation>>

3. Click on the icon next to the <<Target Operation>> field (**Browse for target service operations**).

   The Browse Target Service Operation window is displayed.



Browse Target Service Operation

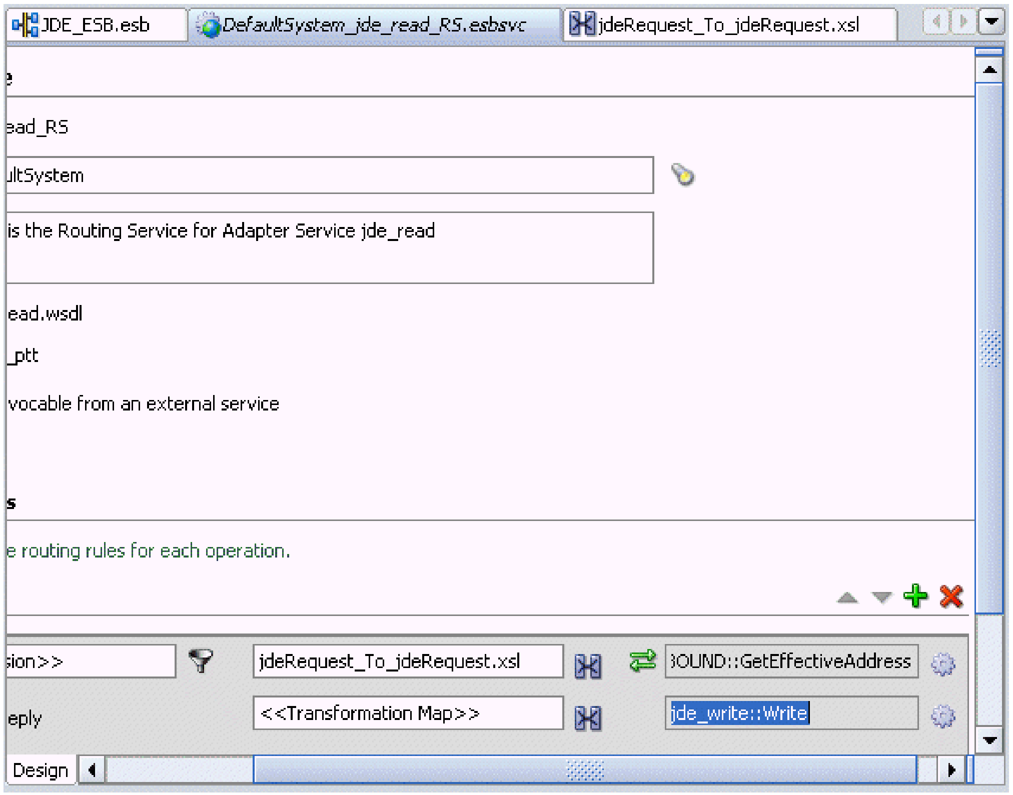
Select an service and click OK.

ESB Services

```
ESB
    <<No Target Operation>>
    Services in project
        DefaultSystem
            JDE_ESB_OUTBOUND
            jde_read_RS
            jde_write
                Write
    Services at ESB Server Connection: Siva-xp
        DefaultSystem
        BPELSystem
```

4. Expand **Services in project**, **Default System**, your adapter service node, for example, **JDE_ESB_Outbound**, and select the service name, for example, **Write**.
5. Click **OK**.

   You are returned to the Routing Rules window.



JDE_ESB.esb
DefaultSystem_jde_read_RS.esbsvc
jdeRequest_To_jdeRequest.xsl
Design

e

ead_RS

ltSystem

is the Routing Service for Adapter Service jde_read

ead.wsdl

_ptt

vocable from an external service

s

e routing rules for each operation.

ion>>

jdeRequest_To_jdeRequest.xsl

3OUND::GetEffectiveAddress

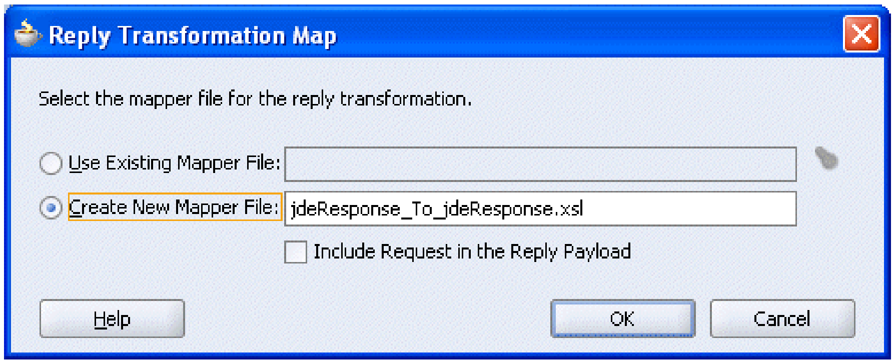
eply

<<Transformation Map>>

jde_write::Write

6. Click on the icon next to the <<Transformation Map>> field (**Select an existing mapper file or create a new one**).

The Reply Transformation Map dialog box is displayed.



Select the mapper file for the reply transformation.

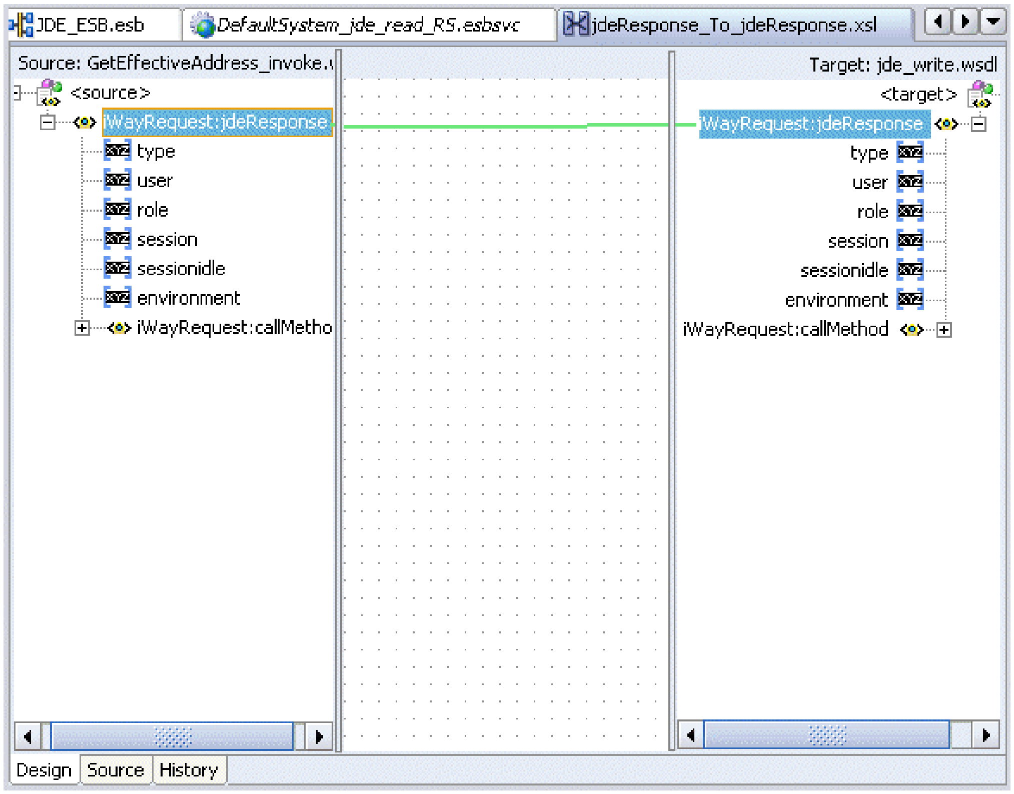
[ ] Use Existing Mapper File:

[✓] Create New Mapper File: jdeResponse_To_jdeResponse.xsl

[ ] Include Request in the Reply Payload

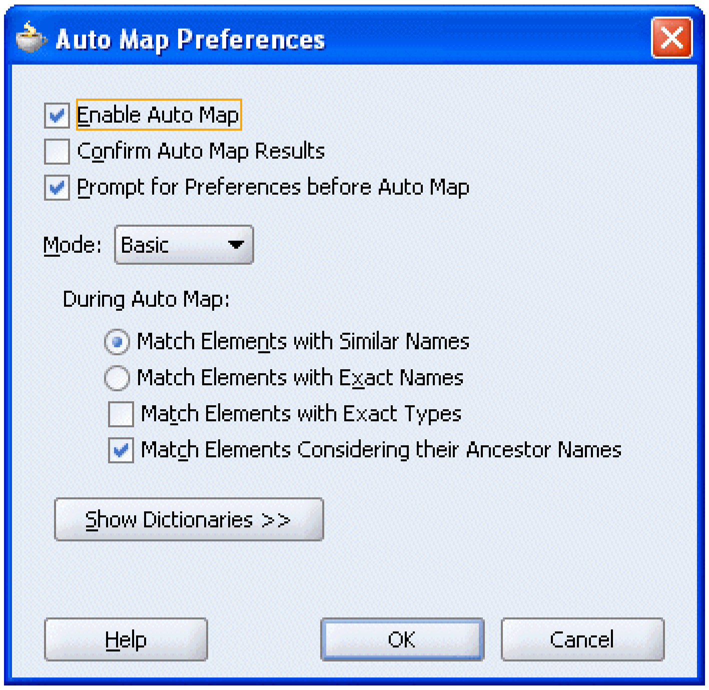
7. Select the **Create New Mapper File** option, specify the file name, and click **OK**.

   The following mapping window is displayed.



8. Select the WSDL file and map it to the Write operation.

   Once you map the WSDL file, the Auto Map Preferences dialog box is displayed.


Auto Map Preferences

[✓] Enable Auto Map
[ ] Confirm Auto Map Results
[✓] Prompt for Preferences before Auto Map

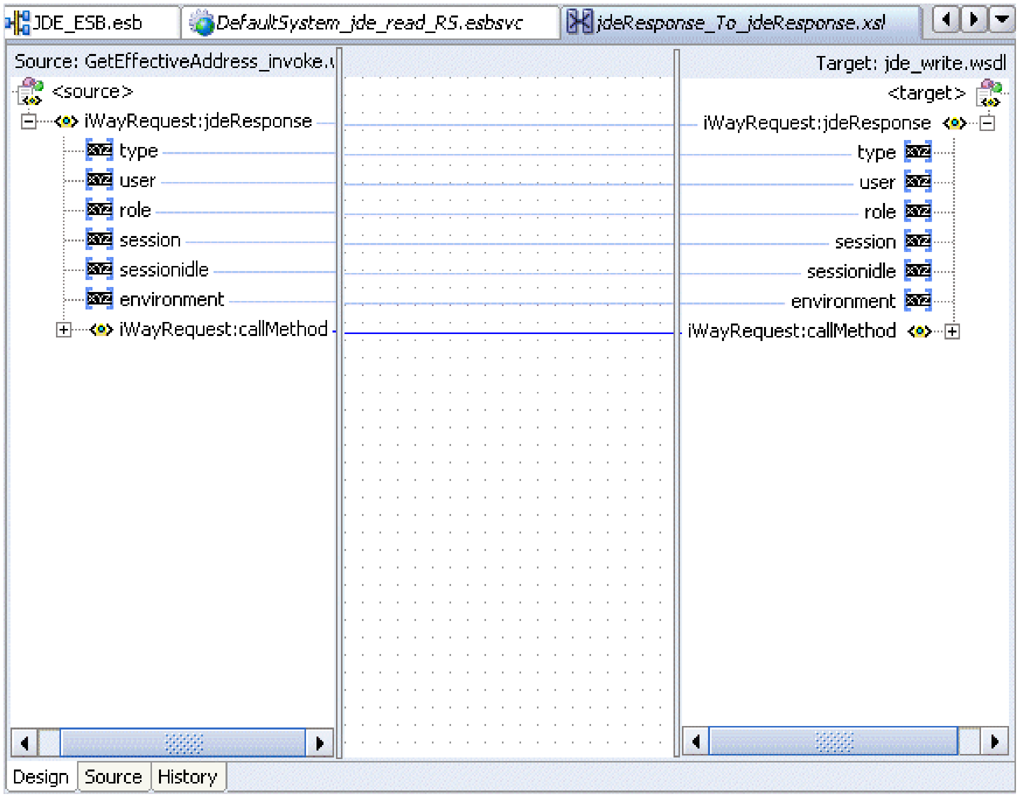
Mode: Basic

During Auto Map:

(●) Match Elements with Similar Names
( ) Match Elements with Exact Names
[ ] Match Elements with Exact Types
[✓] Match Elements Considering their Ancestor Names

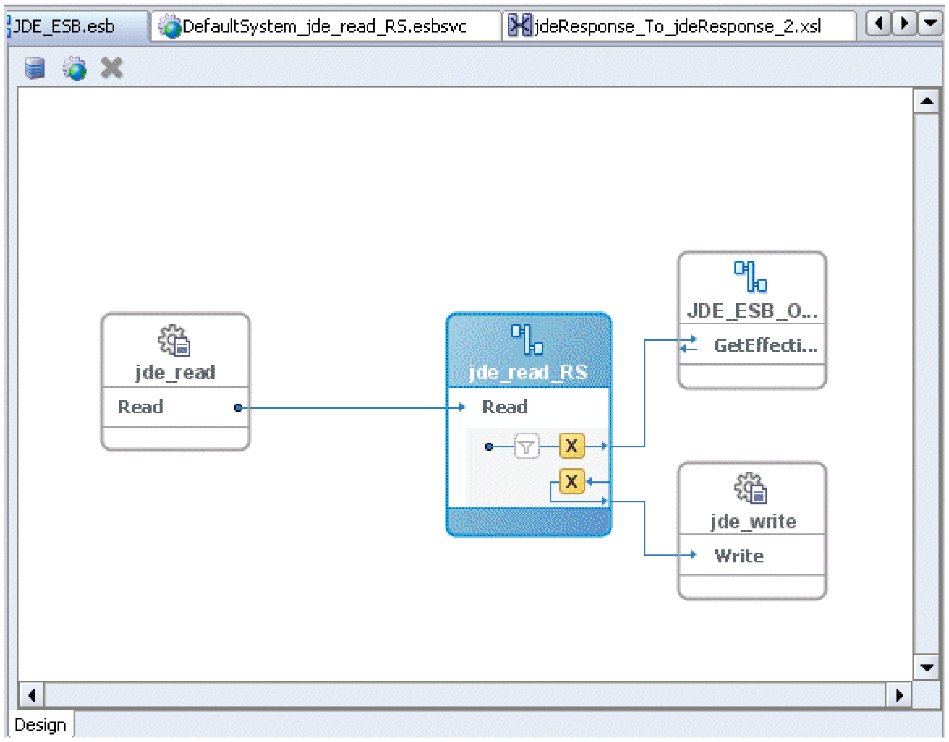
9. Click **OK**.

   The mapping is completed as shown in the following window.



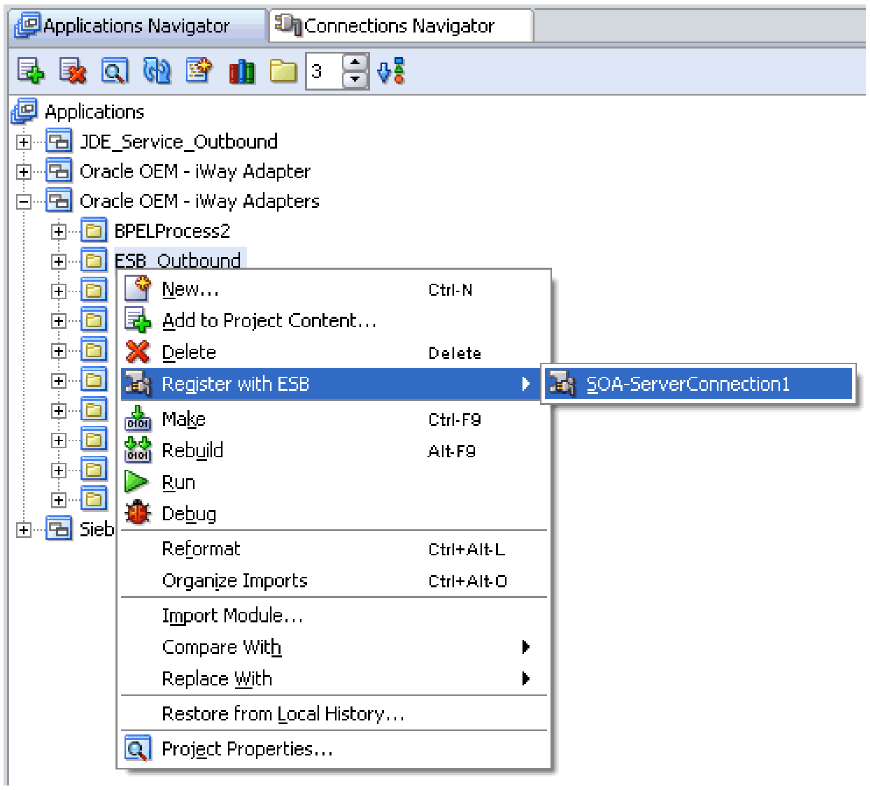
10. Double-click the ESB outbound project file in the left pane, for example, **ESB_ Outbound.esb**.

    Notice that the Routing service is now created for the Write operation.

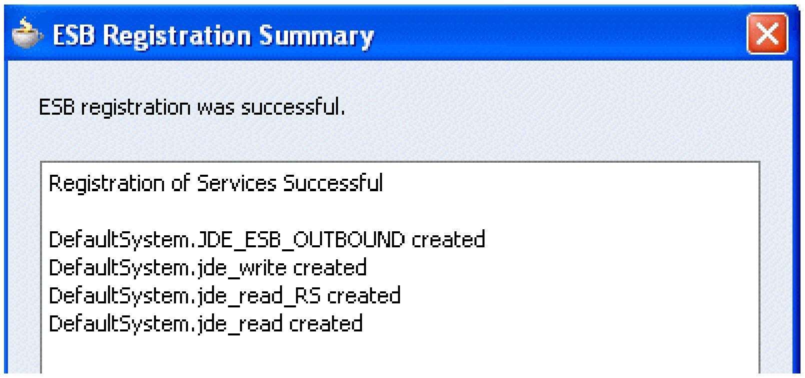


Deploying the Project

1. Right-click the created project, for example, **ESB_Outbound**, select **Register with ESB**, and the server connection, for example, **ServerConnection1**.



After successful deployment, the **Registration of services Successful** message is displayed.

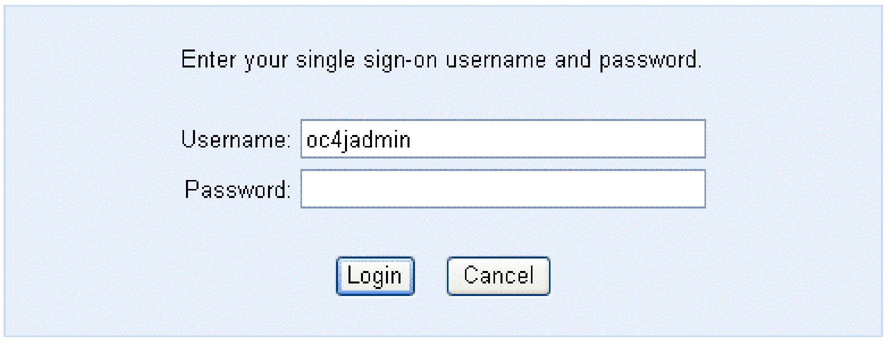


ESB registration was successful.

```
Registration of Services Successful

DefaultSystem.JDE_ESB_OUTBOUND created
DefaultSystem.jde_write created
DefaultSystem.jde_read_RS created
DefaultSystem.jde_read created
```

2. Logon to the ESB Control console to check whether the project has been successfully deployed.

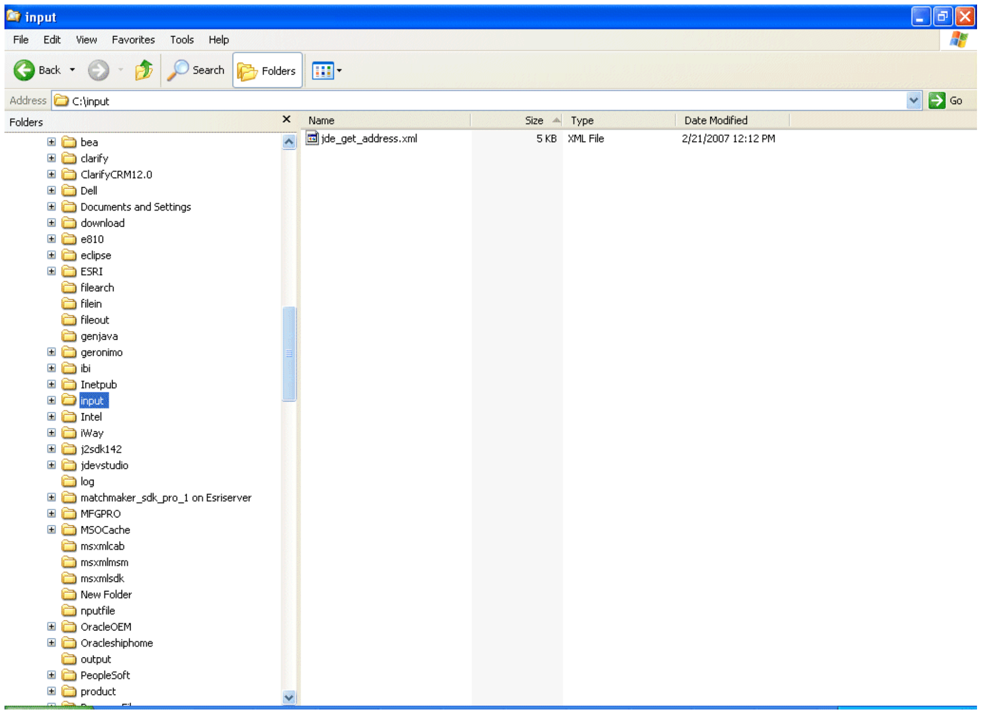

Enter your single sign-on username and password.

Username: oc4jadmin

Password:

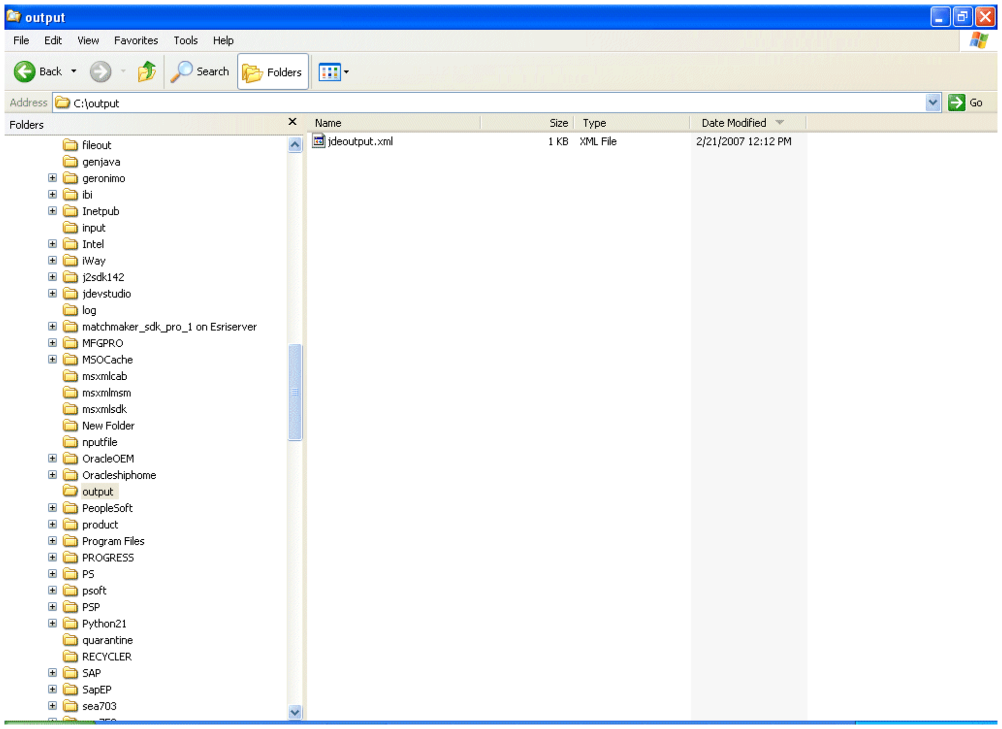
The deployed process is listed under the **Default System** node.

3. Place the XML file in the folder that you specified during the creation of the Read operation.



Name	Size	Type	Date Modified
jde_get_address.xml	5 KB	XML File	2/21/2007 12:12 PM

4. Check whether you are receiving the response in the output folder, which you have specified during the creation of the write operation and also the corresponding instance in the ESB Control console.



Name	Size	Type	Date Modified
jdeoutput.xml	1 KB	XML File	2/21/2007 12:12 PM

5. If the response is not received in the output folder, check the instance and the logs for the corresponding errors in the ESB Control console.

Configuring an ESB Inbound Process

The following example describes how to configure an ESB inbound process to your J.D. Edwards OneWorld system, using an ESB project in Oracle JDeveloper.

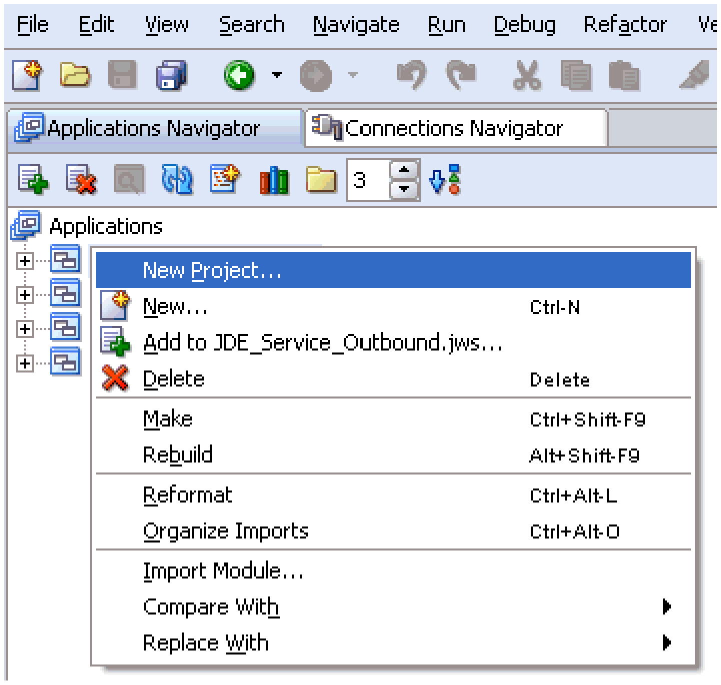
Prerequisites

Before you proceed, you must create an inbound WSDL file for the adapter by using the following steps:

1. Create a target using Application Explorer.
2. Create a channel.
3. Create a WSDL file with the noport option.
4. Restart the Oracle Application Server.

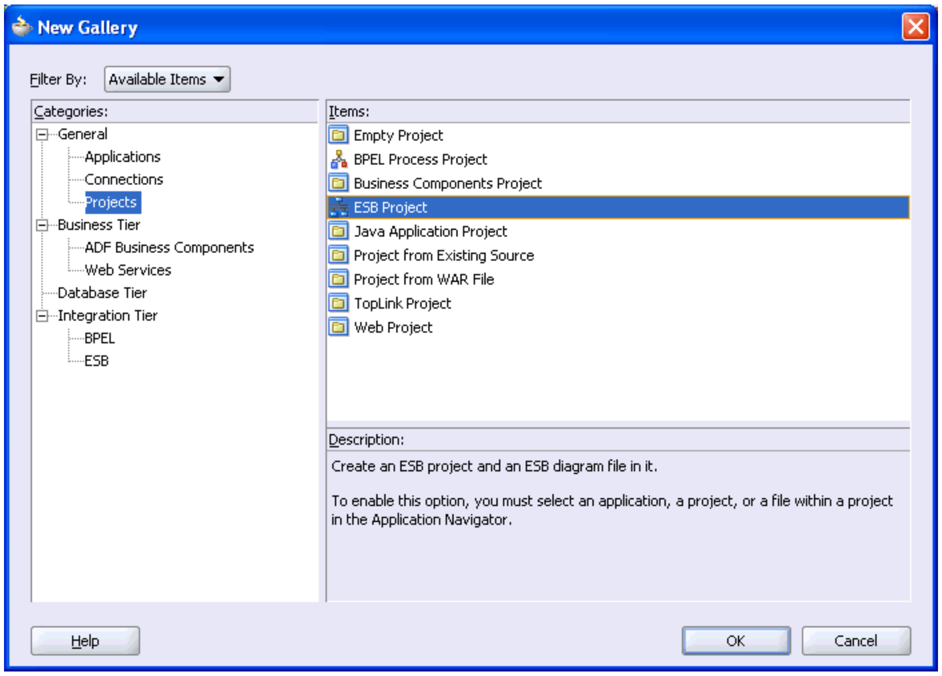
Creating an Inbound ESB Project and Assigning an Inbound WSDL File

1. At the top of the upper left pane, click the **Applications Navigator** tab.



2. Right-click the application node you created and select **New Project**.

   The New Gallery window is displayed.

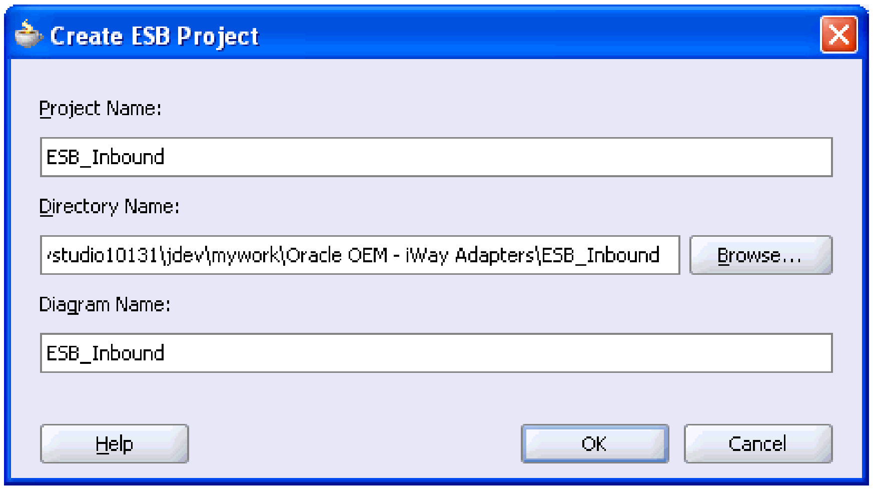


Description:
Create an ESB project and an ESB diagram file in it.

To enable this option, you must select an application, a project, or a file within a project in the Application Navigator.

3. From the Items list, select **ESB Project** and click **OK**.

   The Create ESB Project dialog box is displayed.



Project Name: ESB_Inbound

Directory Name: \studio10131\jdev\mywork\Oracle OEM - iWay Adapters\ESB_Inbound

Diagram Name: ESB_Inbound

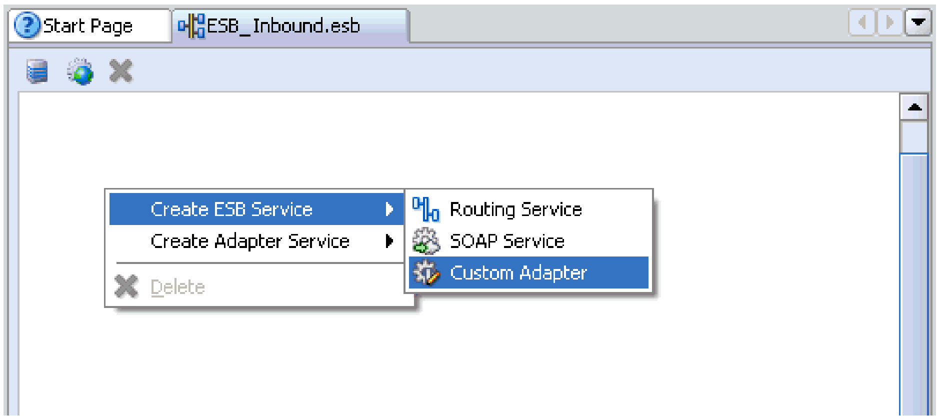
4. Perform the following steps:

   a. Specify a name for the ESB project.

      The Directory Name field and Diagram Name fields are updated automatically.

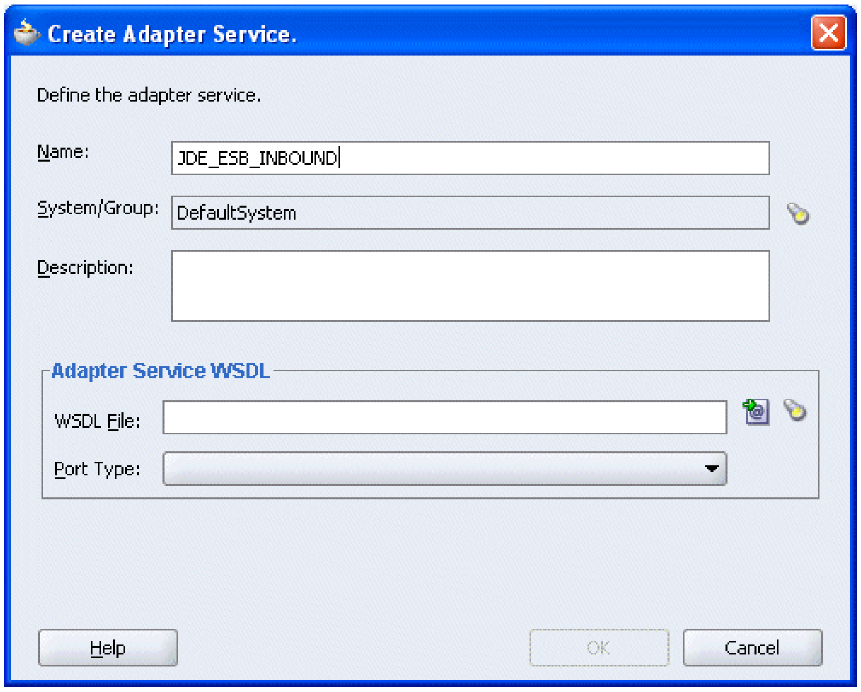
   b. Click **OK**.

   The ESB project is added at the top of the upper left pane.



5. Right-click the ESB project in the middle pane, select **Create ESB Service** followed by **Custom Adapter**.

   The Create Adapter Service dialog box is displayed.



Define the adapter service.

Name: JDE_ESB_INBOUND

System/Group: DefaultSystem

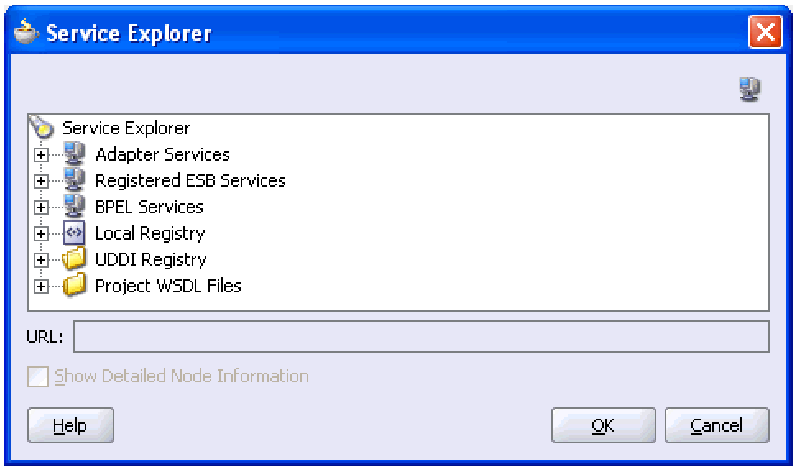
Description:

WSDL File:

Port Type:

6. Enter a name for the adapter service and click the **Service Explorer** icon (second icon from the left preceding the **WSDL File** field).

The Service Explorer dialog box is displayed.



Service Explorer

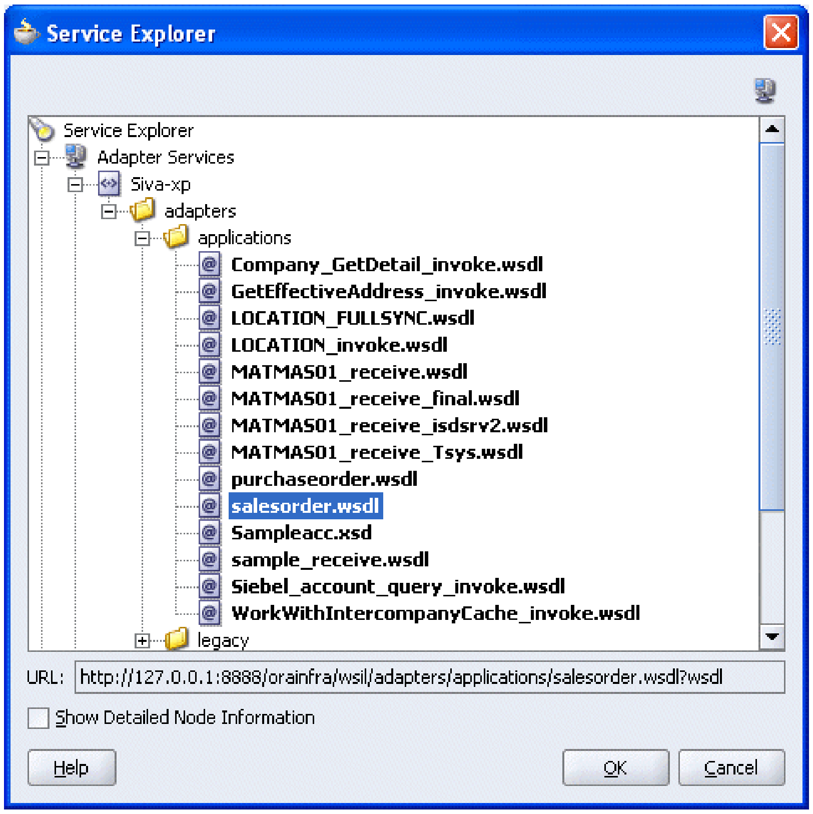
Service Explorer
- Adapter Services
- Registered ESB Services
- BPEL Services
- Local Registry
- UDDI Registry
- Project WSDL Files

URL:

Show Detailed Node Information

7. Expand your new connection under Adapter Services, followed by **adapters**, and then **applications**.

   The WSDL tree displayed in the Service Explorer dialog box lists any WSDL files you have created using Application Explorer. The WSDL tree is generated by a WSDL servlet, which is automatically deployed as part of the BPEL Server installation.



Service Explorer

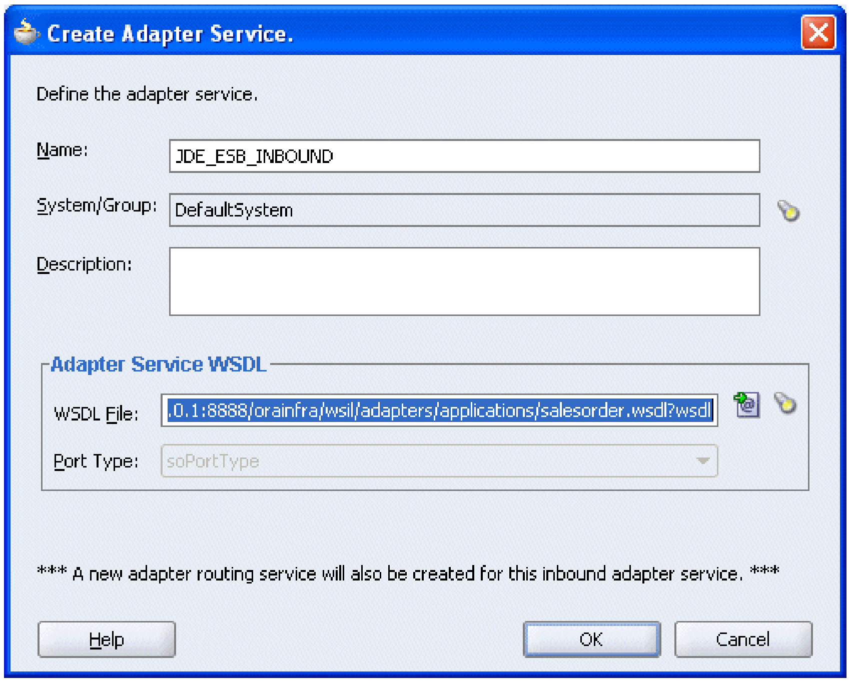
Service Explorer
- Adapter Services
  - Siva-xp
    - adapters
      - applications
        - Company_GetDetail_invoke.wsdl
        - GetEffectiveAddress_invoke.wsdl
        - LOCATION_FULLSYNC.wsdl
        - LOCATION_invoke.wsdl
        - MATMAS01_receive.wsdl
        - MATMAS01_receive_final.wsdl
        - MATMAS01_receive_isdsrv2.wsdl
        - MATMAS01_receive_Tsys.wsdl
        - purchaseorder.wsdl
        - salesorder.wsdl
        - Sampleacc.xsd
        - sample_receive.wsdl
        - Siebel_account_query_invoke.wsdl
        - WorkWithIntercompanyCache_invoke.wsdl
      - legacy

URL: http://127.0.0.1:8888/orainfra/wsil/adapters/applications/salesorder.wsdl?wsdl

Show Detailed Node Information

8. Select an inbound WSDL file that has been created using Application Explorer and click **OK**.

   The **WSDL File** field in the Create Adapter Service dialog box displays the name and location of the selected WSDL file.


Create Adapter Service.

Define the adapter service.

Name: JDE_ESB_INBOUND

System/Group: DefaultSystem

Description:

Adapter Service WSDL

WSDL File: .0.1:8888/orainfra/wsil/adapters/applications/salesorder.wsdl?wsdl

Port Type: soPortType

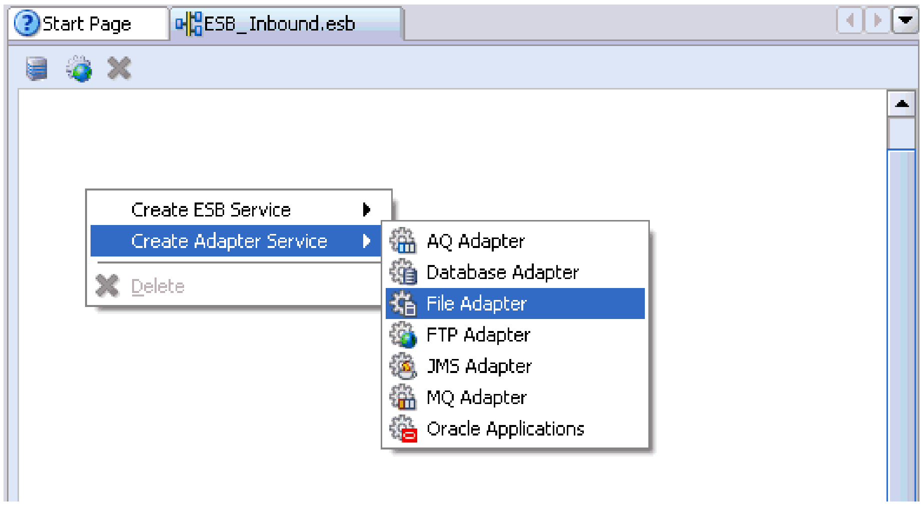
*** A new adapter routing service will also be created for this inbound adapter service. ***

9. Click **OK**.

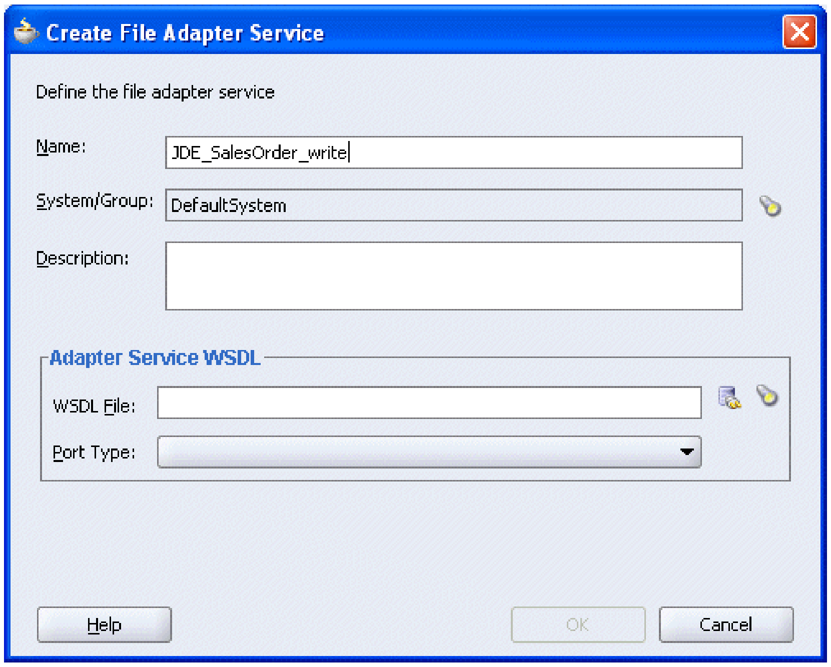
   The new ESB project appears in the visual editor.

Creating a Write Process Operation Using the File Adapter

1. Right-click the ESB project in the middle pane, select **Create Adapter Service** followed by **File Adapter**.



   The Create File Adapter Service dialog box is displayed.


Create File Adapter Service

Define the file adapter service

Name: JDE_SalesOrder_write

System/Group: DefaultSystem

Description:

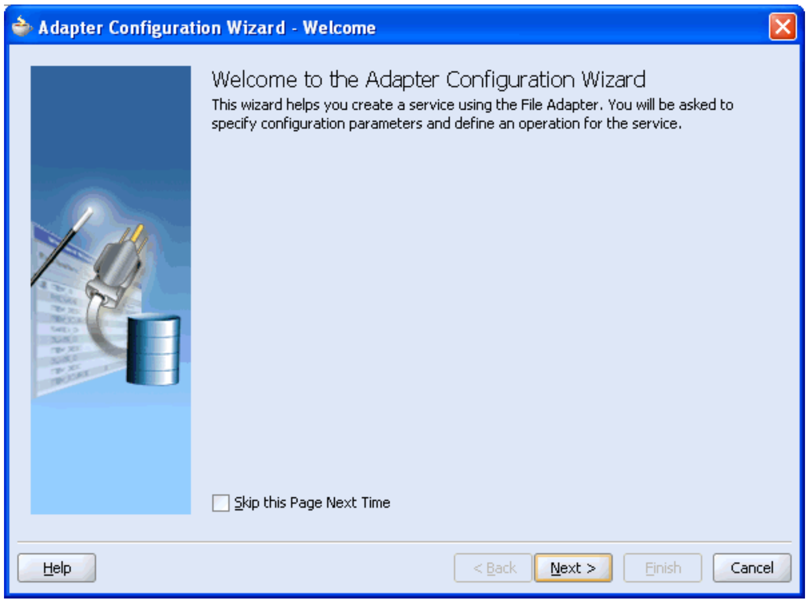
Adapter Service WSDL

WSDL File:

Port Type:

2. Enter a name for the File adapter and click the **Configure adapter service wsdl** icon next to the **WSDL File** field.

   The Adapter Configuration Wizard - Welcome window is displayed.


Adapter Configuration Wizard - Welcome

Welcome to the Adapter Configuration Wizard

This wizard helps you create a service using the File Adapter. You will be asked to specify configuration parameters and define an operation for the service.

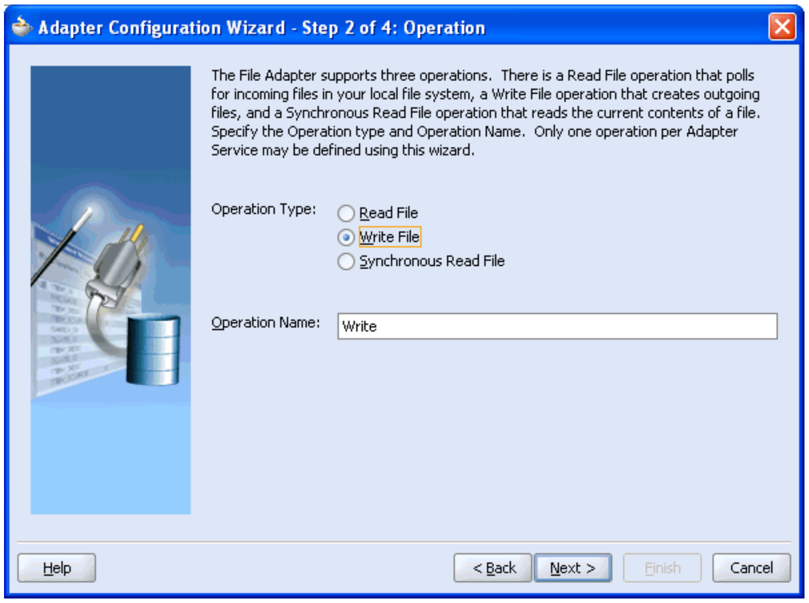
Skip this Page Next Time

3. Click **Next**.

   The Adapter Configuration Wizard - Step 1 of 4: Service Name window is displayed.

4. Click **Next**.

   The Adapter Configuration Wizard - Step 2 of 4: Operation window is displayed.


Adapter Configuration Wizard - Step 2 of 4: Operation

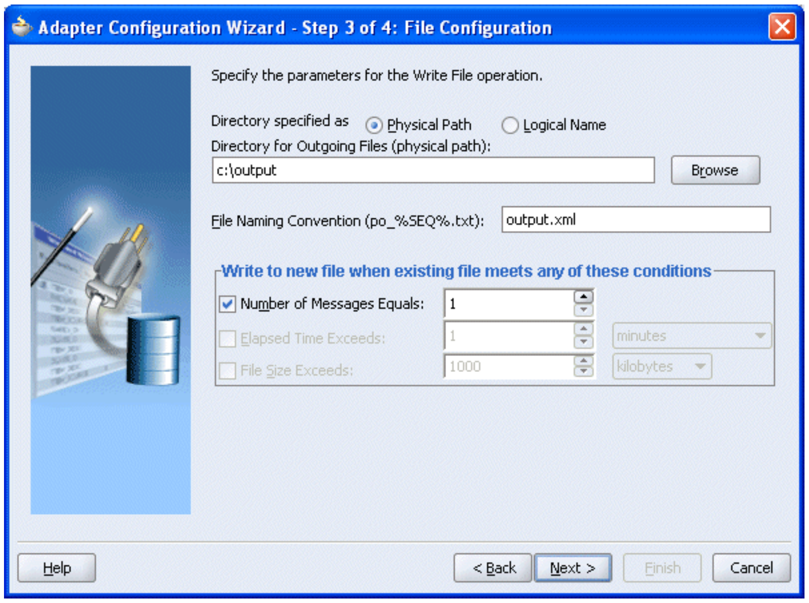
The File Adapter supports three operations. There is a Read File operation that polls for incoming files in your local file system, a Write File operation that creates outgoing files, and a Synchronous Read File operation that reads the current contents of a file. Specify the Operation type and Operation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

Operation Type: Read File [✓ Write File] Synchronous Read File

Operation Name: Write

5. Click **Write File** as the Operation Type and click **Next**.

   The Adapter Configuration Wizard - Step 3 of 4: File Configuration window is displayed.


Adapter Configuration Wizard - Step 3 of 4: File Configuration

Specify the parameters for the Write File operation.

Directory specified as [✓ Physical Path] Logical Name

Directory for Outgoing Files (physical path): c:\output

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

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

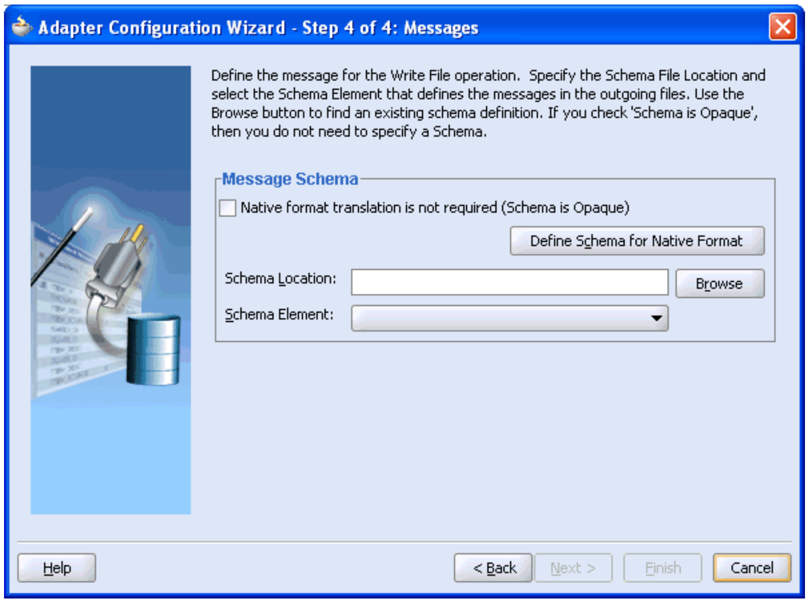
[✓ Number of Messages Equals:] 1

Elapsed Time Exceeds: 1 minutes

File Size Exceeds: 1000 kilobytes

6. Enter the path of the output directory and name of the output file and click **Next**.

   The Adapter Configuration Wizard - Step 4 of 4: Messages window is displayed.


Adapter Configuration Wizard - Step 4 of 4: Messages
Message Schema
Define Schema for Native Format
Browse
Help
< Back
Next >
Finish
Cancel

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.

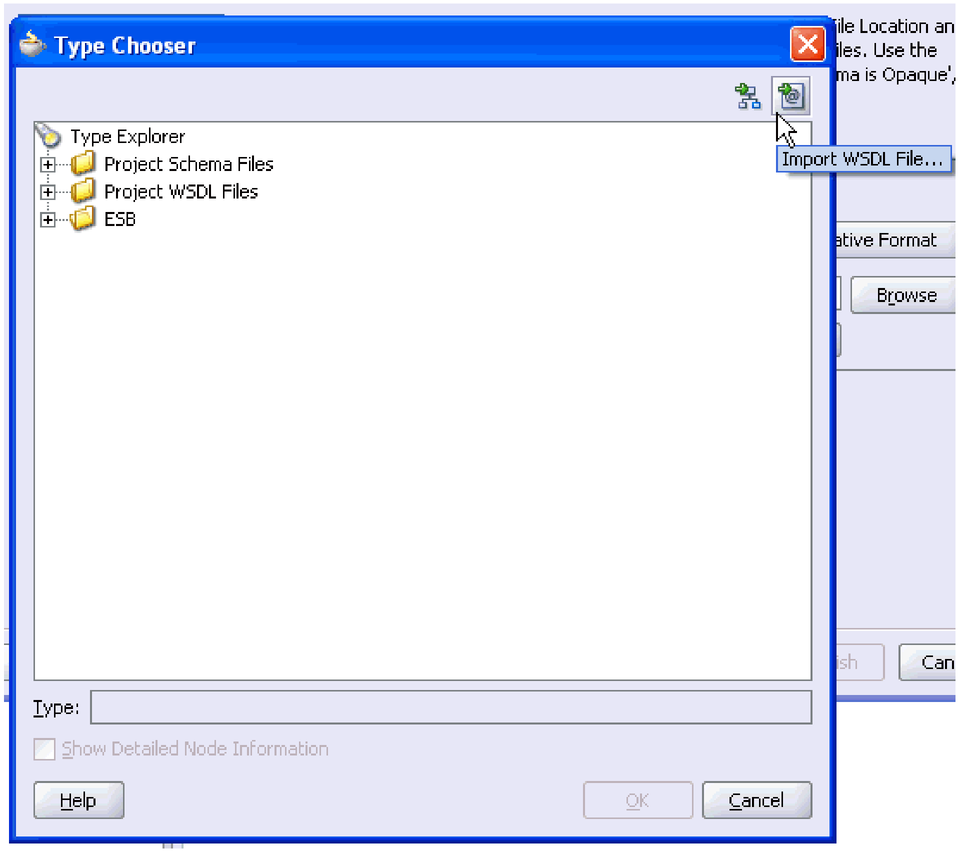
Native format translation is not required (Schema is Opaque)

Schema Location:

Schema Element:

7. Click **Browse** to select the WSDL.

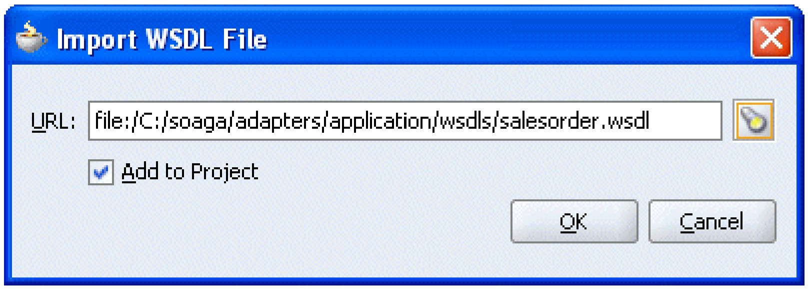
   The Type Chooser window is displayed.



Type:

8. Click the **Import WSDL File** icon on the upper right corner of the dialog box.

   The Import WSDL File dialog box is displayed.

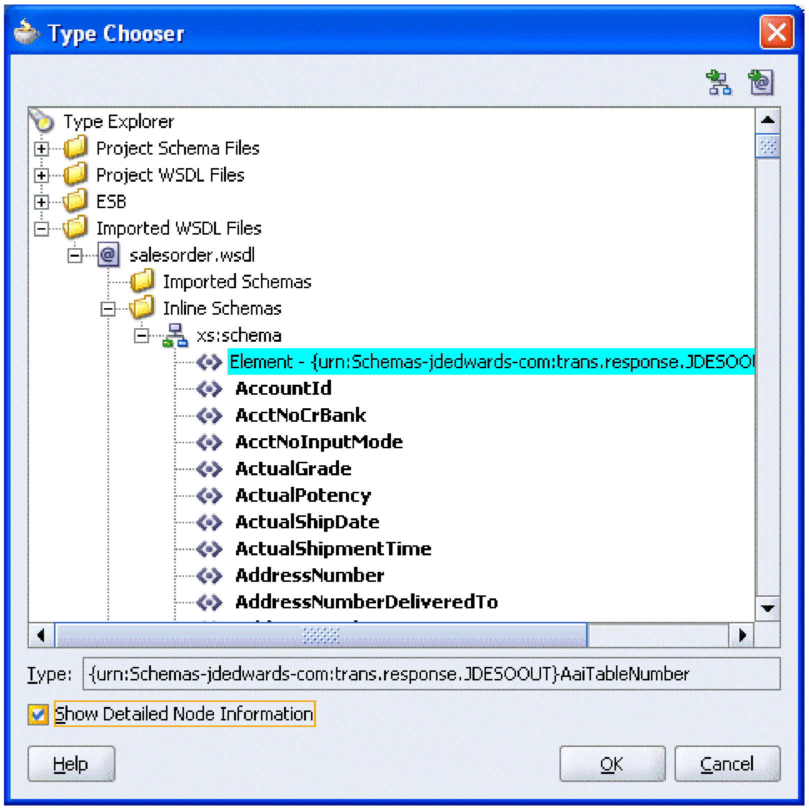


URL: file:/C:/soaga/adapters/application/wsdls/salesorder.wsdl

Add to Project: ✓

9. Select the WSDL file and click **OK**.

   The Imported WSDL Files folder is added.

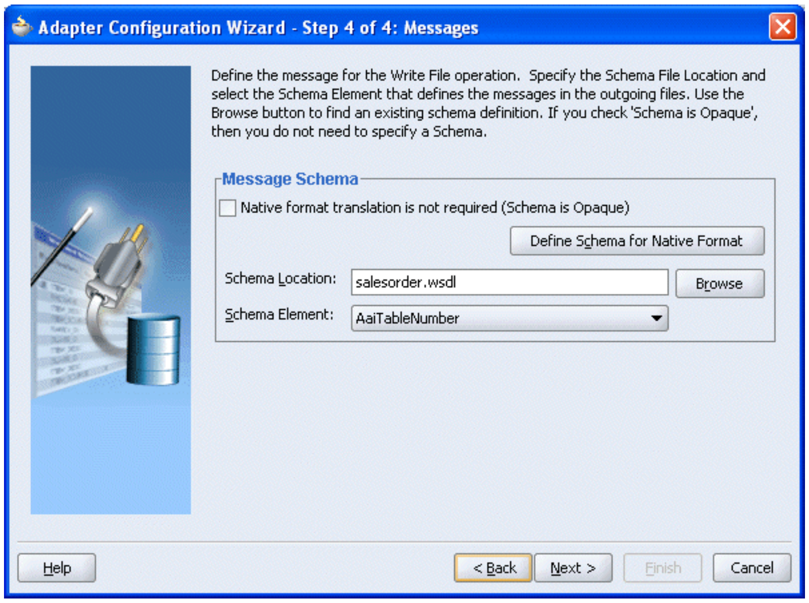


Type: {urn:Schemas-jdedwards-com:trans.response.JDESOOUT}AaiTableNumber

Show Detailed Node Information: ✓

10. Expand the Imported WSDL Files folder, select an Inline Schema, for example, **LOCATION_SYNC**, and click **OK**.

    You are returned to the Adapter Configuration Wizard - Step 4 of 4: Messages window.


Adapter Configuration Wizard - Step 4 of 4: Messages
Message Schema
Define Schema for Native Format
Browse
Help
< Back
Next >
Finish
Cancel

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.

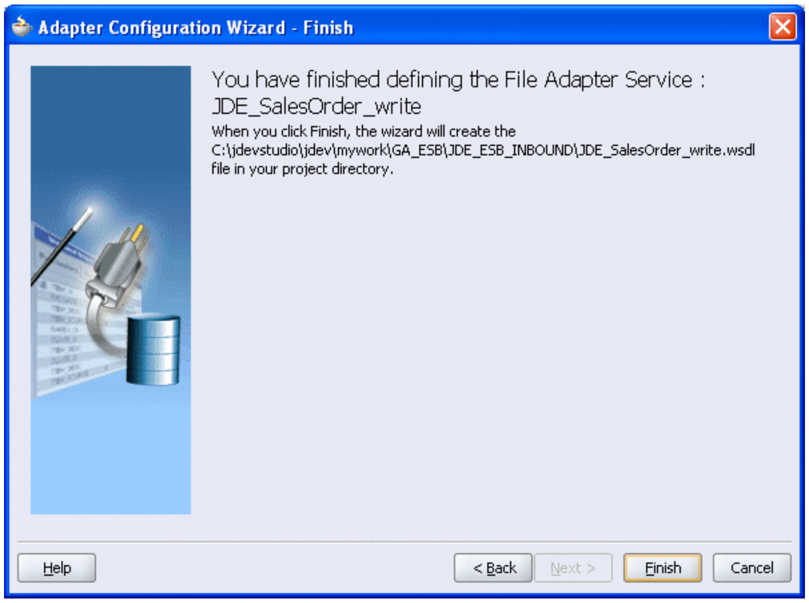
Native format translation is not required (Schema is Opaque)

Schema Location: salesorder.wsdl

Schema Element: AaiTableNumber

11. Click **Next**.

    The Adapter Configuration Wizard - Finish window is displayed.

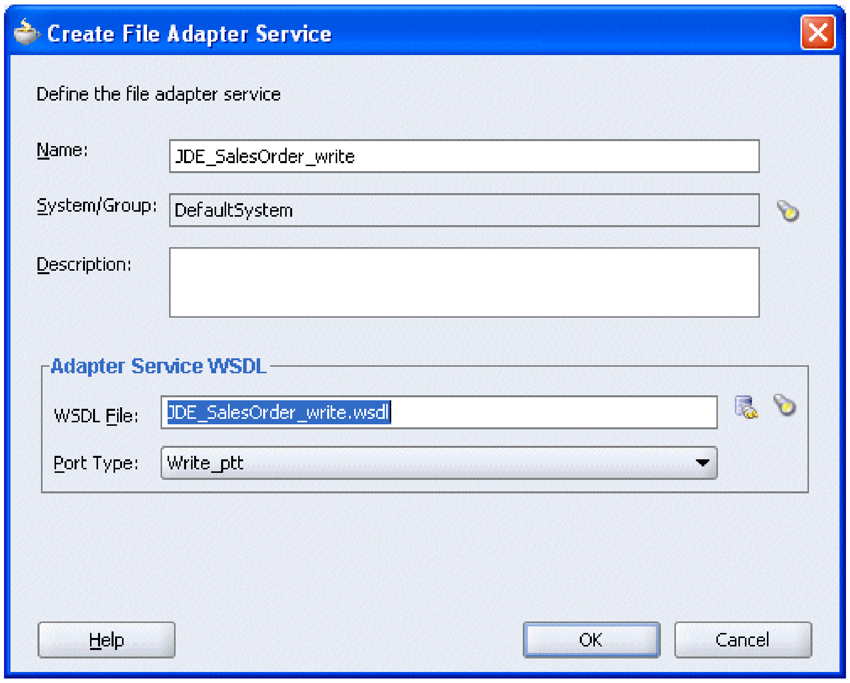


You have finished defining the File Adapter Service : JDE_SalesOrder_write

When you click Finish, the wizard will create the C:\jdevstudio\jdev\mywork\GA_ESB\JDE_ESB_INBOUND\JDE_SalesOrder_write.wsdl file in your project directory.

12. Click **Finish**.

    You are returned to the Create File Adapter Service dialog box.


Create File Adapter Service

Define the file adapter service

Name:	JDE_SalesOrder_write
System/Group:	DefaultSystem
Description:	

Adapter Service WSDL

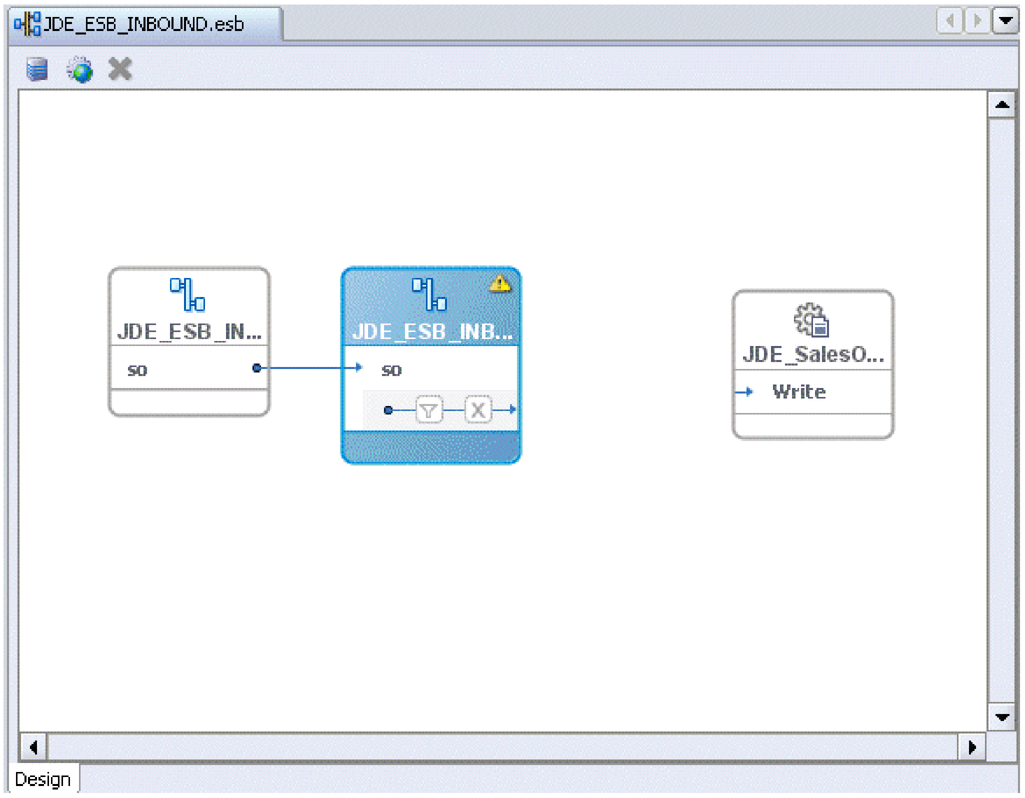
WSDL File:	JDE_SalesOrder_write.wsdl
Port Type:	Write_ptt

13. Click **OK**.

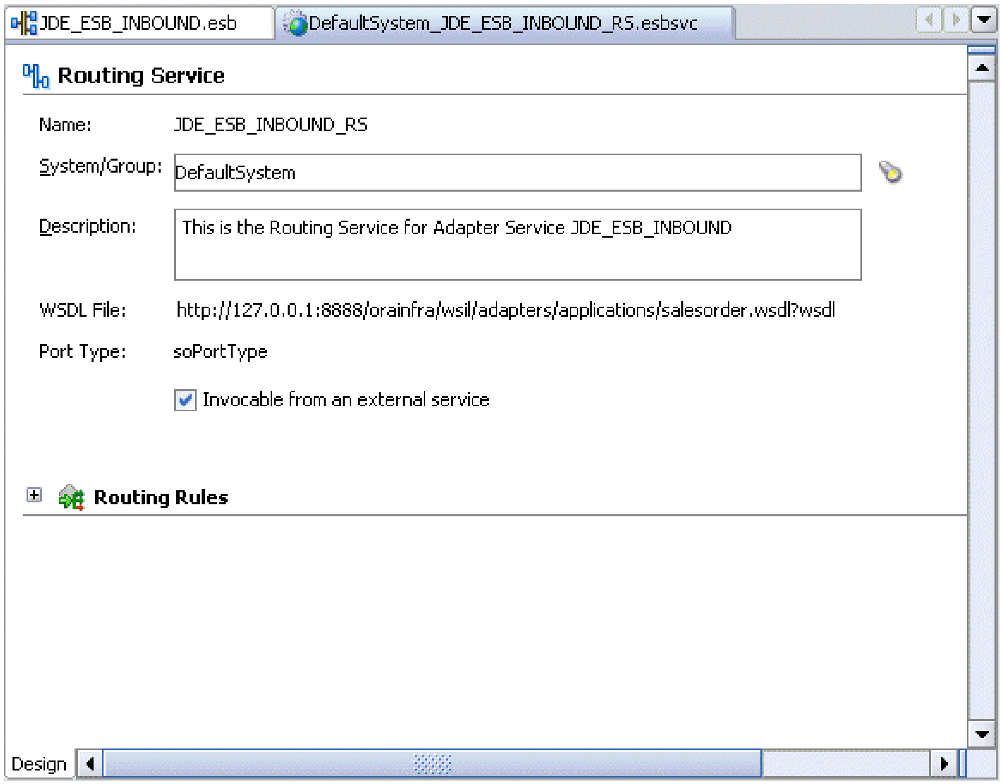
    The Write operation with a routing service is added to the ESB inbound project view.

Providing a Routing Service for the Write Operation

1. Double-click the routing service.



The Routing Service window is displayed.



Name: JDE_ESB_INBOUND_RS

System/Group: DefaultSystem

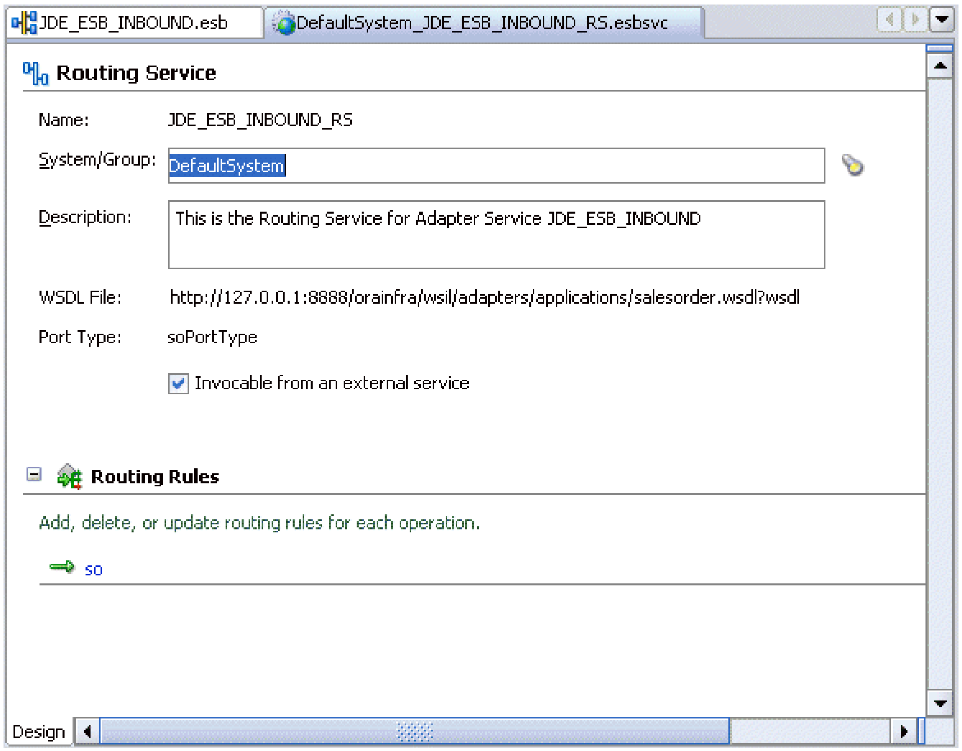
Description: This is the Routing Service for Adapter Service JDE_ESB_INBOUND

WSDL File: http://127.0.0.1:8888/orainfra/wsil/adapters/applications/salesorder.wsdl?wsdl

Port Type: soPortType

☑ Invocable from an external service

2. Expand the **Routing Rules**.



Name: JDE_ESB_INBOUND_RS

System/Group: DefaultSystem

Description: This is the Routing Service for Adapter Service JDE_ESB_INBOUND

WSDL File: http://127.0.0.1:8888/orainfra/wsil/adapters/applications/salesorder.wsdl?wsdl

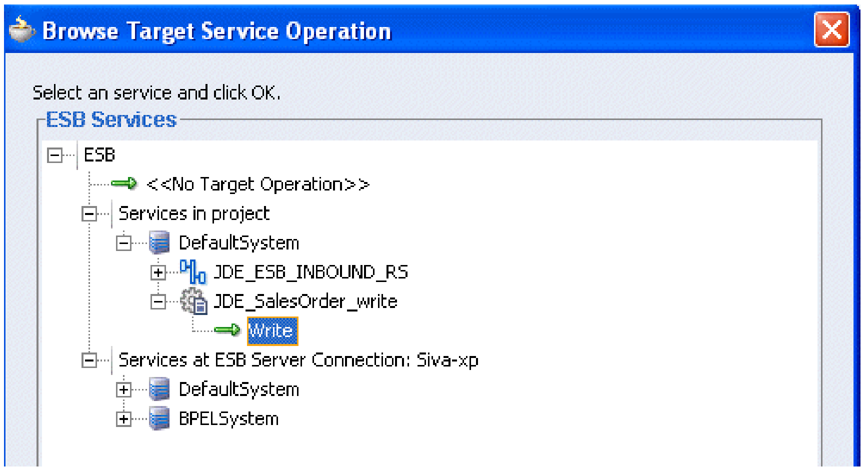
Port Type: soPortType

☑ Invocable from an external service

Add, delete, or update routing rules for each operation.

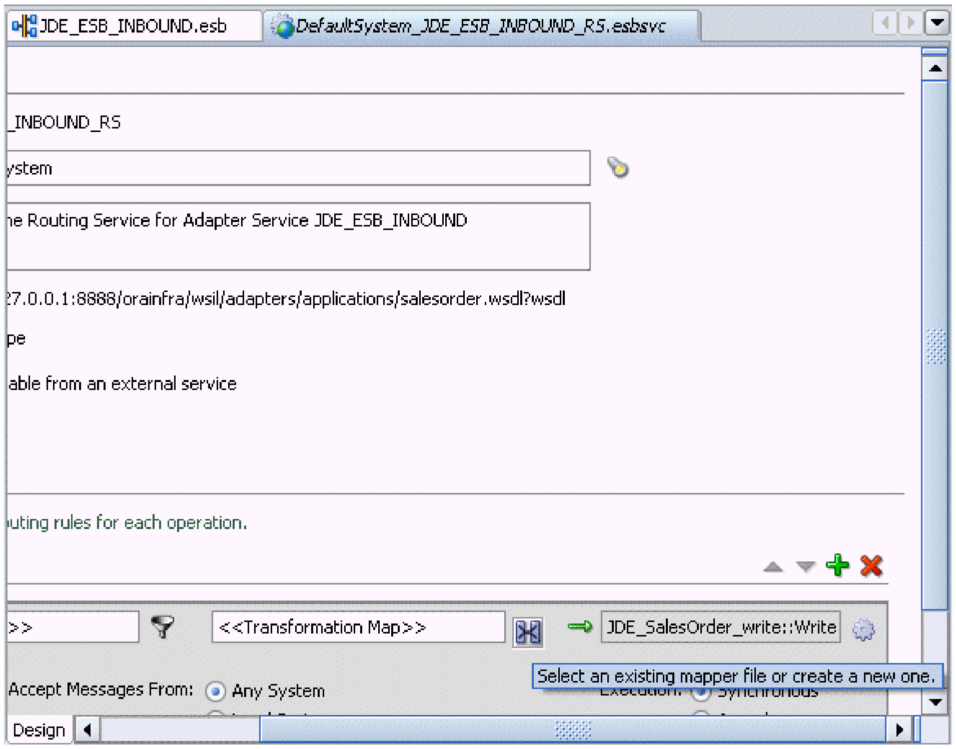
3. Click the green plus sign icon, which represents the option to **Create a new Routing Rule**.

The Browse Target Service Operation window is displayed.



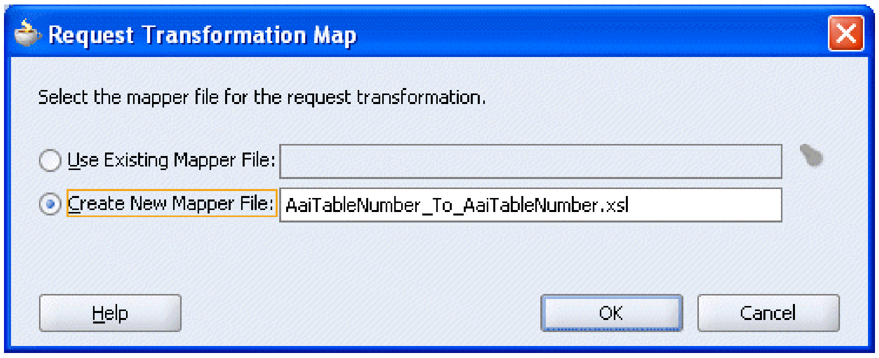
4. Expand **Services in project**, **Default System**, your adapter service node, for example, **JDE_ESB_Inbound_RS**, and select the service name, for example, **Write**.
5. Click **OK**.

   You are returned to the Routing Rules window.


JDE_ESB_INBOUND.esb
DefaultSystem_JDE_ESB_INBOUND_RS.esbsvc
_INBOUND_RS
ystem
ne Routing Service for Adapter Service JDE_ESB_INBOUND
27.0.0.1:8888/orainfra/wsil/adapters/applications/salesorder.wsdl?wsdl
pe
able from an external service
uting rules for each operation.
>>
<<Transformation Map>>
JDE_SalesOrder_write::Write
Select an existing mapper file or create a new one.
Accept Messages From:
Any System
Execution:
Synchronous
Design

6. Click on the icon next to the <<Transformation Map>> field (**Select an existing mapper file or create a new one**).

   The Request Transformation Map dialog box is displayed.



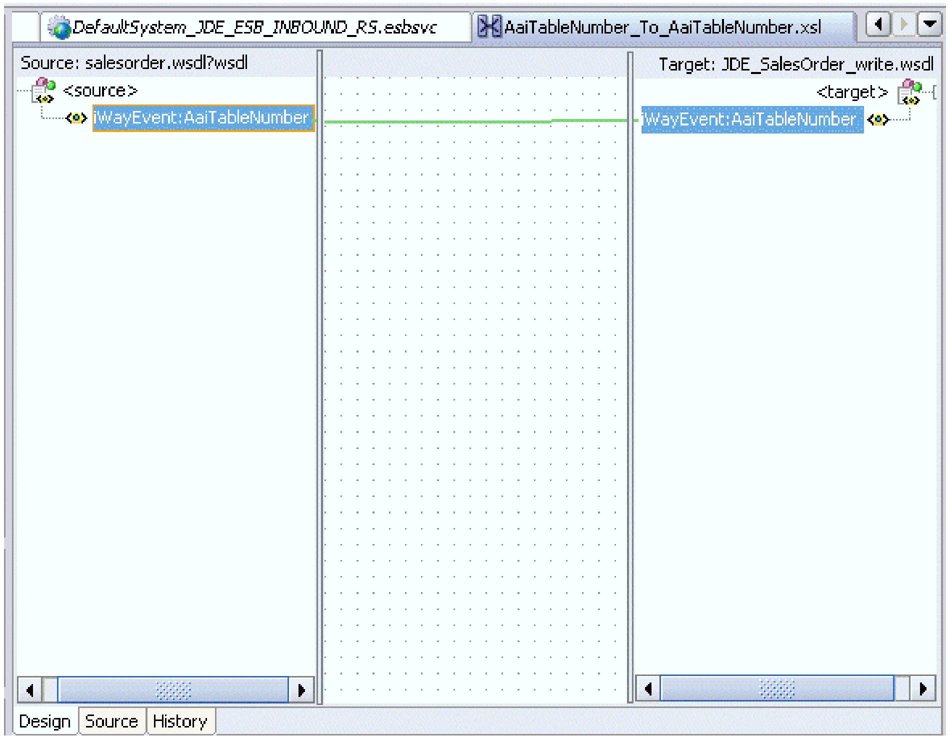
Select the mapper file for the request transformation.

( ) Use Existing Mapper File:

(●) Create New Mapper File: AaiTableNumber_To_AaiTableNumber.xsl

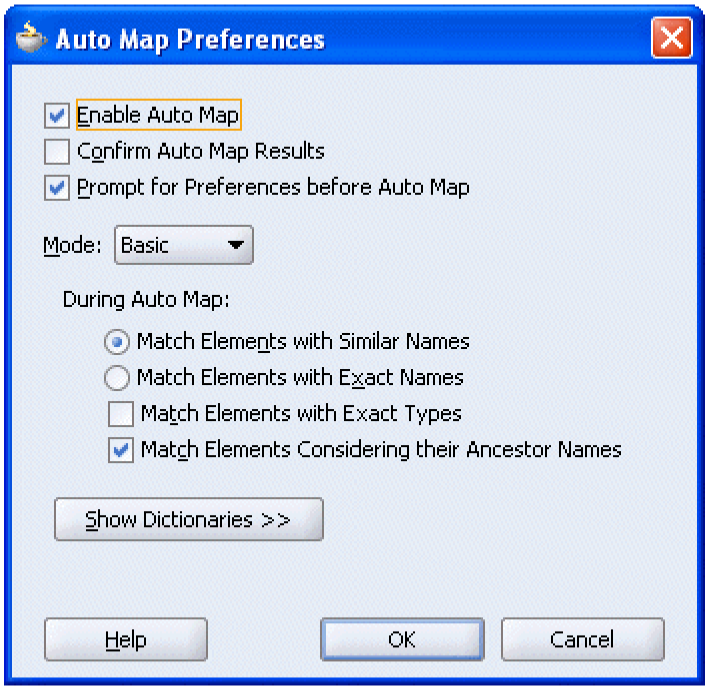
7. Select the **Create New Mapper File** option, specify the file name, and click **OK**.

   The following mapping window is displayed.


DefaultSystem_JDE_ESB_INBOUND_RS.esbsvc
AaiTableNumber_To_AaiTableNumber.xsl
Source: salesorder.wsdl?wsdl
<source>
iWayEvent:AaiTableNumber
Target: JDE_SalesOrder_write.wsdl
<target>
iWayEvent:AaiTableNumber
Design
Source
History

8. Select the WSDL file and map it to the Write operation.

   Once you map the WSDL file, the Auto Map Preferences dialog box is displayed.


Auto Map Preferences

[✓] Enable Auto Map
[ ] Confirm Auto Map Results
[✓] Prompt for Preferences before Auto Map

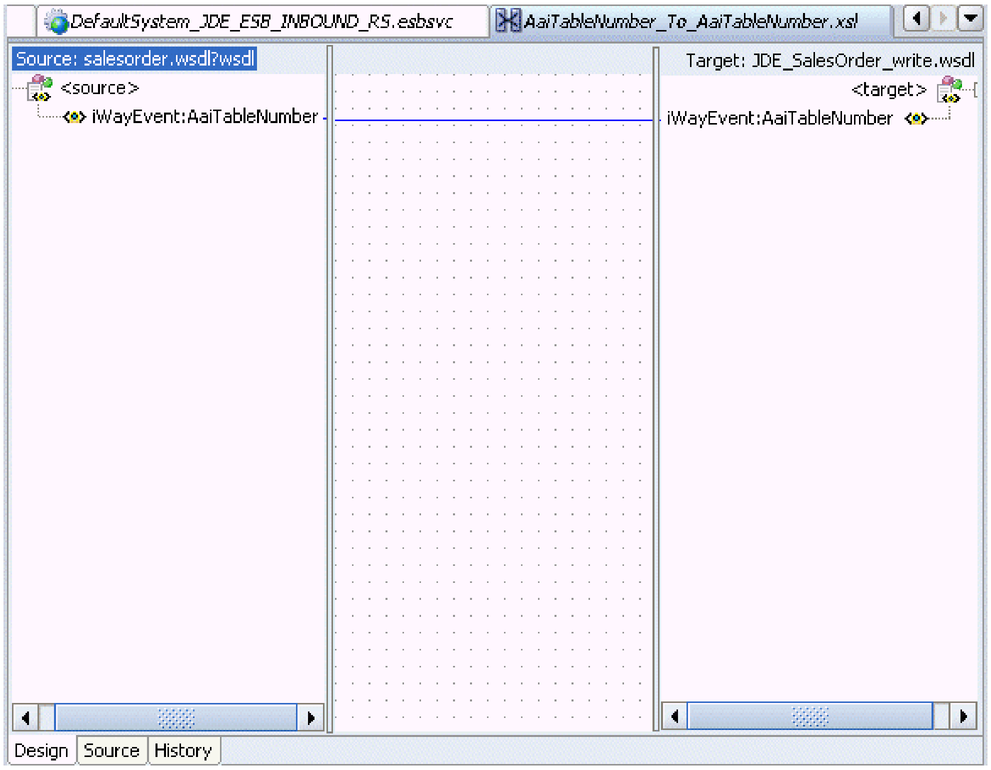
Mode: Basic

During Auto Map:

(●) Match Elements with Similar Names
( ) Match Elements with Exact Names
[ ] Match Elements with Exact Types
[✓] Match Elements Considering their Ancestor Names

9. Click **OK**.

   The mapping is completed as shown in the following window.

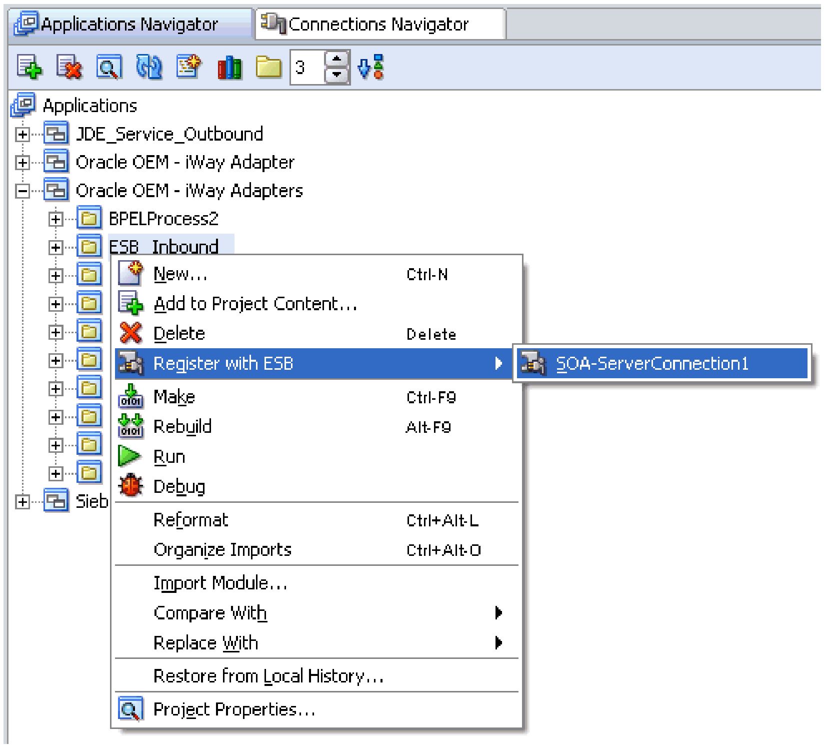

DefaultSystem_JDE_ESB_INBOUND_RS.esbsvc
AaiTableNumber_To_AaiTableNumber.xsl
Source: salesorder.wsdl?wsdl
<source>
iWayEvent:AaiTableNumber
Target: JDE_SalesOrder_write.wsdl
<target>
iWayEvent:AaiTableNumber
Design
Source
History

10. Double-click the ESB inbound project file in the left pane, for example, **ESB_Inbound.esb**.

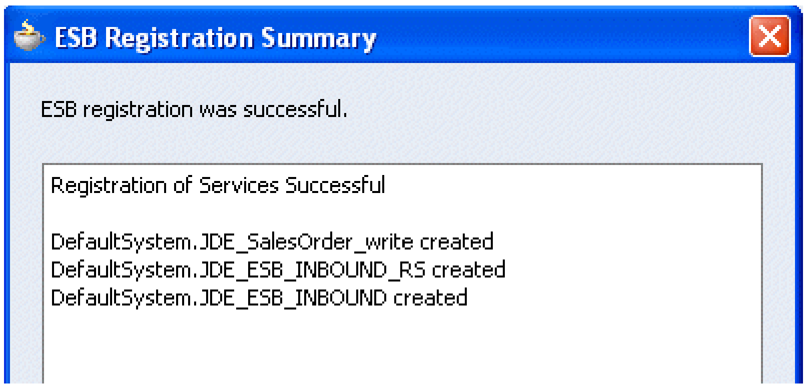
    Notice that the Routing service is now created for the Write operation in the middle pane.

Deploying the Project

1. Right-click the created project, for example, **ESB_Outbound**, select **Register with ESB**, and the server connection, for example, **ServerConnection1**.



After successful deployment, the **Registration of services Successful** message is displayed.

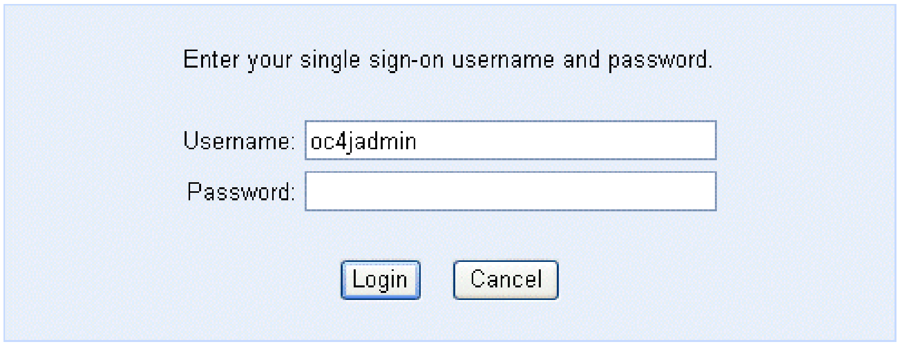


ESB registration was successful.

```
Registration of Services Successful

DefaultSystem.JDE_SalesOrder_write created
DefaultSystem.JDE_ESB_INBOUND_RS created
DefaultSystem.JDE_ESB_INBOUND created
```

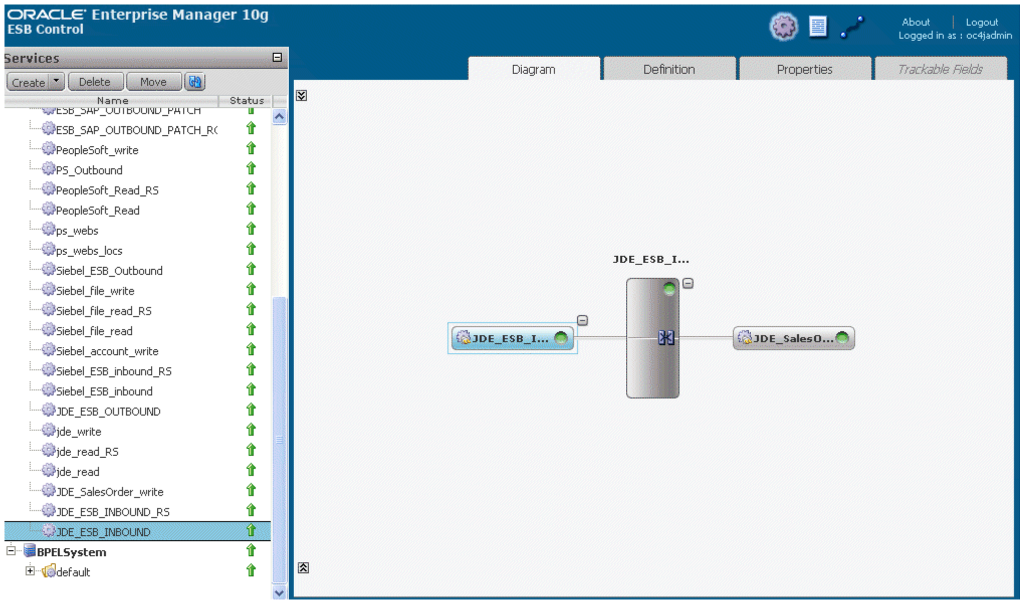
2. Logon to the ESB Control console to check whether the project has been successfully deployed.


Enter your single sign-on username and password.

Username: oc4jadmin

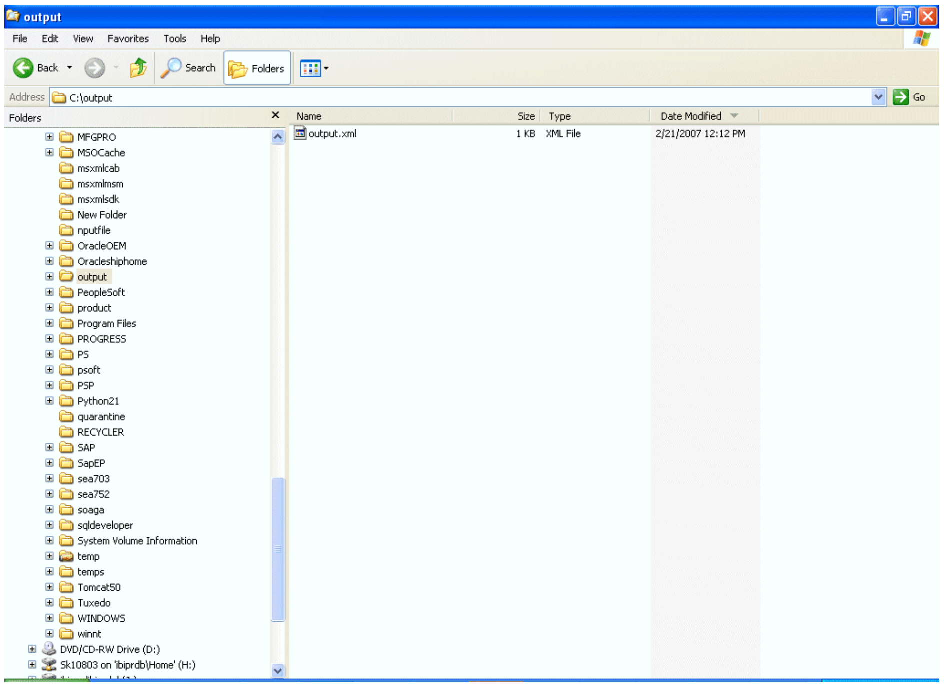
Password:

The deployed process is listed under the **Default System** node.



ESB_SAP_OUTBOUND_PATCH
ESB_SAP_OUTBOUND_PATCH_R(
PeopleSoft_write
PS_Outbound
PeopleSoft_Read_RS
PeopleSoft_Read
ps_webs
ps_webs_locs
Siebel_ESB_Outbound
Siebel_file_write
Siebel_file_read_RS
Siebel_file_read
Siebel_account_write
Siebel_ESB_inbound_RS
Siebel_ESB_inbound
JDE_ESB_OUTBOUND
jde_write
jde_read_RS
jde_read
JDE_SalesOrder_write
JDE_ESB_INBOUND_RS
JDE_ESB_INBOUND
BPELSystem
default

3. Trigger the event.
4. Check whether you are receiving the response in the output folder, which you have specified during the creation of the write operation.



Name	Size	Type	Date Modified
output.xml	1 KB	XML File	2/21/2007 12:12 PM

5. If the response is not received in the output folder, check the instance and the logs for the corresponding errors in the ESB Control console.

7 Troubleshooting and Error Messages

This chapter explains the limitations and workarounds when connecting to J.D. Edwards OneWorld. The following topics are discussed:

- Troubleshooting
- BSE Error Messages

The adapter-specific errors listed in this chapter can arise whether using the adapter with an OracleAS Adapter J2CA or with a OracleAS Adapter Business Services Engine (BSE) configuration.

Troubleshooting

This topic provides troubleshooting information for J.D. Edwards OneWorld, separated into four categories:

- OracleAS Adapter Application Explorer (Application Explorer)
- J.D. Edwards OneWorld
- OracleAS Adapter J2CA
- BSE

Note: Log file information that can be relevant in troubleshooting can be found in the following locations:

- OracleAS Adapter J2CA trace information can be found under the `*OracleAS_home*\opmn\logs` directory.
- BSE trace information can be found under the `*OracleAS_home*\j2ee\home\applications\ws-app-adapter\ibse\ibselogs` directory.
- The log file for Application Explorer can be found under the `*OracleAS_home*\adapters\application\tools` directory.

Application Explorer

To use Application Explorer on **Windows** for debugging or testing purposes, load the batch script `ae.bat`, found under:

```
OracleAS_home\adapters\application\tools
```

On **UNIX**, load the shell script `ae.sh`, found under:

```
OracleAS_home/adapters/application/tools
```

Error	Solution
Cannot connect to OracleAS Adapter for J.D. Edwards OneWorld from Application Explorer: `Problem activating adapter. (Failed to connect to J.D.Edwards, check system availability and/or configuration parameters:...) Check logs for more information.`	Ensure that: ■ J.D. Edwards OneWorld is running. ■ The J.D. Edwards OneWorld user ID and password is correct. ■ The port number is correct.
The following error message appears: java.lang.IllegalStateException: java.lang.Exception: Error Logon to J.D. Edwards OneWorld System	You have provided invalid connection information for J.D. Edwards OneWorld or the wrong JAR file is in the lib directory.
J.D. Edwards OneWorld does not appear in the Application Explorer Adapter node list.	Ensure that the J.D. Edwards OneWorld JAR files, are added to the lib directory.
Logon failure error at runtime.	If the password for connecting to your J.D. Edwards OneWorld system is not specified when creating a target or with the Edit option in Application Explorer, you will be unable to connect to J.D. Edwards OneWorld. The connection password is not saved in `repository.xml`. Update the password using the Edit option in Application Explorer, then restart the application server.
The following exception occurs when you start Application Explorer by activating `ae.bat` (not `iaexplorer.exe`): `java.lang.ClassNotFoundException: org.bouncycastle.jce.provider.BouncyCastleProvider`	This is a benign exception. It does not affect adapter functionality. Download BouncyCastle files from: `ftp://ftp.bouncycastle.org/pub`

Unable to start Application Explorer in a Solaris environment. The following exception is thrown in the console:
`javax.resource.ResourceException: IWAFManagedConnectionFactory: License violation.at com.ibi.afjca.spi.IWAFManagedConnectionFactory.createConnectionFactory(IWAFManagedConnectionFactory.java:98)at com.iwaysoftware.iwae.common.JCATransport.getConnectionFactory(JCATransport.java:133) at com.iwaysoftware.iwae.common.JCATransport.initJCA(JCATransport.java:69)at com.iwaysoftware.iwae.common.JCATransport.<init>(JCATransport.java:62)at com.iwaysoftware.iwae.common.AdapterClient.<init>(AdapterClient.java:85)at com.ibi.bse.ConfigWorker.run(ConfigWorker.java:41)at java.lang.Thread.run(Thread.java:534)`
`Could not create the connection factory.`
	`JAVACMD` is not set on the user system. Before starting Application Explorer, export `JAVACMD` as follows:
	`JAVACMD=/<jdk_home>/bin/java`, where `<jdk_home>` is the directory where JDK is installed on your machine.

J.D. Edwards OneWorld

Error	Cause	Solution
Action code invalid	In the Sales Order request, the Action code appears as "H," an invalid action code.	Use: ■ "I" for inquiry. ■ "C" for change. ■ "D" for delete. ■ "A" to add a new record.
Invalid address number.	The address number does not exist in the Address Book Master file (F0101).	Enter an address number using the Address Book Revisions program (PO1051). Ensure that the number entered is correct.
Record invalid	The record being processed either already exists for an ADD function or does not exist for an INQUIRY, CHANGE, or DELETE function.	If you are attempting to inquire, change, or delete a record you added previously, there could be database problems in your production library. Contact your data processing department.
Item Branch record does not exist.	An Item Branch record (F4102) does not exist for this item in the Branch/Plant specified.	Correct the Branch or enter an Item Branch record for this item in Branch Plant Item Information (P41026).
&1 does not match any of the valid values.	The &1 does not match any of the valid values specified in the Data Dictionary for this field.	Enter a valid value.
Date out of range.	The Last Service Date and the Inspection Date must be within the range of the effective dates of the Service Contract.	Change the date to be greater than or equal to the beginning effective date and less than or equal to the ending effective date of the Service Contract.
Jde.net timeout exception	Net timeout is set to a wrong value	Verify that net timeout is set to 180 at `jde.ini` of [NETWORK QUEUE SETTINGS], for example `JDENETTimeout=180`
Cannot connect to EnterpriseOne Version 8.10	Missing required library files	`Kernel.jar` and `Connector.jar` are required for version B7333. `jdeutil.jar` and `log4j.jar` are required for EnterpriseOne Version 8.10, in addition to `Kernel.jar` and `Connector.jar`.

OracleAS Adapter J2CA

Error	Solution
In Application Explorer, the following error message appears when you attempt to connect to an OracleAS Adapter J2CA configuration: `Could not initialize JCA`	In the Details tab in the right pane, ensure that the directory specified in the Home field points to the correct directory, for example, *`OracleAS_home`*\adapters\application

BPEL Process Manager

Error	Solution
Endpoint activation error on deployment of J.D.Edwards event handling project (inbound) in JDeveloper	Verify that the channel used for this inbound J2CA service is stopped in Application Explorer. If you have started this channel for testing or debugging purposes, you must stop it before starting BPEL PM Server. Endpoint activation is managed by BPEL Process Manager.
The following error message appears in BPEL PM Server Console: `Process "TestJDE" (revision "1.0") compilation failed. <2005-05-18 10:49:53,285> <ERROR><default.collaxa.cube.engine.deployment> <Cube ProcessLoader::create> Failed to read wsdl. Error happened when reading wsdl at "http://127.0.0.1:7777/BPELConsole/wsil/adapters/applications/GetEffectiveAddress_invoke.wsdl?wsdl", because "WSDLException: faultCode=INVALID_WSDL: Invalid XML in document at: http://127.0.0.1:7777/BPELConsole/wsil/adapters/applications/GetEffectiveAddress_invoke.wsdl?wsdl: The element type "P" must be terminated by the matching end-tag "</P>".`	Verify that the specified WSDL file exists at that URL and that the file is valid. Workaround: Change the WSDL location to `localhost:7777`. The default is `127.0.0.1:7777`. Alternative workaround: Add the IP address to the `Dhttp.nonProxyHosts` list found in `obsetenv.bat` (Windows) or `obsetenv.sh` (Unix)
Second message invocation fails at runtime.	Verify that you have all the required patches installed. The required patches are listed and updated on the Oracle Technology Network Web site at `http://www.oracle.com/technology/index.html`
The following exception is thrown in JDeveloper during deployment of the BPEL process: `java.io.FileNotFoundException: \BPELConsole\wsil\adapters\applications\SalesOrder_receive.wsdl?wsdl (The system cannot find the path specified)`	Verify that you have all the required patches installed. The required patches are listed and updated on the Oracle Technology Network Web site at `http://www.oracle.com/technology/index.html`

BSE Error Messages

This topic discusses the different types of errors that can occur when processing Web services through BSE.

General Error Handling in BSE

BSE serves as both a SOAP gateway into the adapter framework and as the engine for some of the adapters. In both design time and runtime, various conditions can cause errors in BSE when Web services that use adapters are running. Some of these conditions and resulting errors are exposed the same way, regardless of the specific adapter; others are exposed differently, based on the adapter being used. This topic explains what you can expect when you encounter some of the more common error conditions on an adapter-specific basis.Usually the SOAP gateway (agent) inside BSE passes a SOAP request message to the adapter required for the Web service. If an error occurs, how it is exposed depends on the adapter and the API or interfaces that the adapter uses. A few scenarios cause the SOAP gateway to generate a SOAP fault. In general, anytime the SOAP agent inside BSE receives an invalid SOAP request, a SOAP fault element is generated in the SOAP response. The SOAP fault element contains fault string and fault code elements. The fault code contains a description of the SOAP agent error. The following SOAP response document results when BSE receives an invalid SOAP request:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

 <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
         <faultcode>SOAP-ENV:Client</faultcode>
         <faultstring>Parameter node is missing</faultstring>
      </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

In this example, BSE did not receive an element in the SOAP request message that is mandatory for the WSDL for this Web service.

Adapter-Specific Error Handling

When an adapter raises an exception during runtime, the SOAP agent in BSE produces a SOAP fault element in the generated SOAP response. The SOAP fault element contains fault code and fault string elements. The fault string contains the native error description from the adapter target system. Since adapters use the target system interfaces and APIs, whether or not an exception is raised depends on how the target systems interface or API treats the error condition. If a SOAP request message is passed to an adapter by the SOAP agent in BSE, and that request is invalid based on the WSDL for that service, the adapter may raise an exception yielding a SOAP fault.

While it is almost impossible to anticipate every error condition that an adapter may encounter, the following is a description of how adapters handle common error conditions and how they are then exposed to the Web services consumer application.

Invalid SOAP Request

If Oracle Application Server Adapter receives a SOAP request message that does not conform to the WSDL for the Web services being executed, then the following SOAP response is generated.

```
<?xml version="1.0" encoding="ISO-8859-1"
 ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
 <SOAP-ENV:Fault>
  <faultcode>SOAP-ENV:Server</faultcode>
  <faultstring>RPC server connection failed: Connection refused:
connect</faultstring>
 </SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Empty Result From Oracle Application Server Adapter Request

If Oracle Application Server Adapter executes a SOAP request using input parameters passed that do not match records in the target system, then the following SOAP response is generated.

Note: The condition for this adapter does not yield a SOAP fault.

```
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
   <SOAP-ENV:Body>
      <m:RunDBQueryResponse xmlns:m="urn:schemas-iwaysoftware-com:iwse"
        xmlns="urn:schemas-iwaysoftware-com:iwse"
        cid="2A3CB42703EB20203F91951B89F3C5AF">
        <RunDBQueryResult run="1" />
     </m:RunDBQueryResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Error Logging In

If Oracle Application Server Adapter executes an invalid SOAP log in request, then the following SOAP response is generated.

```
[2004-07-19T16:28:56:718Z] DEBUG (SOAP1) W.SOAP1.2: POST received
[2004-07-19T16:28:56:718Z] DEBUG (SOAP1) W.SOAP1.2: in XDSOAPHTTPWorker agentName
is [XDSOAPRouter]
[2004-07-19T16:28:56:718Z] DEBUG (SOAP1) W.SOAP1.2: before parse:
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Header>
<m:ibsinfo xmlns:m="urn:schemas-iwaysoftware-...[861]
[2004-07-19T16:28:56:718Z] ERROR (SOAP1) W.SOAP1.2: Attempting string, no encoding
recognized in document
[2004-07-19T16:28:56:734Z] DEEP (SOAP1) W.SOAP1.2: parse complete in 16 msecs
[2004-07-19T16:28:56:859Z] DEEP (SOAP1) W.SOAP1.2: ST_NODICT
[2004-07-19T16:28:56:859Z] DEEP (SOAP1) W.SOAP1.2: ST_FINISH
[2004-07-19T16:28:56:859Z] DEBUG (SOAP1) extractControl - edaDoc: false
[2004-07-19T16:28:56:859Z] DEBUG (SOAP1) now: 2004-07-19T16:28:56Z expires:
2004-07-20T16:28:56Z
[2004-07-19T16:28:56:859Z] DEBUG (SOAP1) W.SOAP1.2: checking for cached agent
[2004-07-19T16:28:56:859Z] DEBUG (SOAP1) W.SOAP1.2: pushagent: adding agent
com.ibi.iwse.XDSOAPRouter
[2004-07-19T16:28:56:875Z] DEBUG (SOAP1) W.SOAP1.2: inside worker the soap Action
is [B0100033.GetEffectiveAddressRequest#test##]
[2004-07-19T16:28:56:875Z] DEBUG (SOAP1) W.SOAP1.2: precedence: 1
[2004-07-19T16:28:56:875Z] DEBUG (SOAP1) W.SOAP1.2: precedence: 1
[2004-07-19T16:28:56:875Z] DEBUG (SOAP1) W.SOAP1.2: numagents: 1
[2004-07-19T16:28:56:890Z] DEBUG (SOAP1) W.SOAP1.2: running agent 1 name
com.ibi.iwse.XDSOAPRouter document 1
[2004-07-19T16:28:56:890Z] INFO  (manager) MGR00X01: Adding active worker:
W.SOAP1.2
[2004-07-19T16:28:56:890Z] DEBUG (SOAP1) W.SOAP1.2: <?xml version="1.0"
encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <SOAP-ENV:Header>
      <m:ibsinfo xmlns:m="urn:schemas-iwaysoftware-com:iwse">
         <m:service>B0100033</m:service>
         <m:method>GetEffectiveAddress</m:method>
         <m:license>test</m:license>
         <m:Username>user</m:Username>
         <m:Password>password</m:Password>
      </m:ibsinfo>
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      <m:GetEffectiveAddress
xmlns:m="urn:iwaysoftware:ibse:jul2003:GetEffectiveAddress">
         <m:jdeRequest type="callmethod">
            <m:callMethod name="GetEffectiveAddress">
               <m:params>
                  <m:param name="mnAddressNumber">12345</m:param>
               </m:params>
               <m:onError/>
            </m:callMethod>
         </m:jdeRequest>
      </m:GetEffectiveAddress>
   </SOAP-ENV:Body>
   <SOAPAction agentName="XDSOAPRouter"
cid="1FF3D44E0B0AFB2A4E9538ED42B71437">B0100033.GetEffectiveAddressRequest#test##<
/SOAPAction>
</SOAP-ENV:Envelope>
[2004-07-19T16:28:56:890Z] DEBUG (SOAP1) W.SOAP1.2: business method:
m:GetEffectiveAddress
[2004-07-19T16:28:56:906Z] DEBUG (SOAP1) W.SOAP1.2: input:
[2004-07-19T16:28:56:906Z] DEBUG (SOAP1) W.SOAP1.2: <?xml version="1.0"
encoding="UTF-8" ?><jdeRequest xmlns:xsd="http://www.w3.org/2001/XMLSchema"
type="callmethod" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><callMethod
name="GetEffectiveAddress"><params><param name="mnAddressNumber">12345</param>
      </params><onError/></callMethod></jdeRequest>
[2004-07-19T16:28:58:234Z] DEBUG (SOAP1) W.SOAP1.2: Agent returned success
[2004-07-19T16:28:58:234Z] INFO  (manager) MGR00X02: Removing active worker:
W.SOAP1.2
[2004-07-19T16:28:58:234Z] DEBUG (SOAP1) W.SOAP1.2: doing docTran, docVal,
listTran for agent(1)
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: sendToAll reply to XDReply:
[protocol=http */null]
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: preemitters from doc: null
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: no preemitters, emitting
contents of doc, usestream=false encoding=UTF-8
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: writeEntity, len: 670 data:
<?xml version="1.0" encoding="UTF-8" ?><SOAP-ENV:Envelope
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><SOAP-ENV:Body><GetEffective
AddressResponse xmlns="urn:iwaysoftware:ibse:jul2003:GetEffectiveAddress:response"
cid="1FF3D44E0B0AFB2A4E9538ED42B71437"><jdeResponse user="USER" type="callmethod"
session="" environment="DV7333"><returnCode code="12">Environment
&apos;DV7333&apos; could not be initialized for user, check user, pwd and
environment attribute
values</returnCode></jdeResponse></GetEffectiveAddressResponse></SOAP-ENV:Body></S
OAP-ENV:Envelope>
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: writeString: HTTP/1.0
```

```
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: writeString: 200
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: writeString: OK
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: writeString: Content-Type:
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: writeString: text/xml
[2004-07-19T16:28:58:250Z] DEBUG (SOAP1) W.SOAP1.2: writeString: Content-Length:
[2004-07-19T16:28:58:265Z] DEBUG (SOAP1) W.SOAP1.2: writeString: 670
[2004-07-19T16:28:58:265Z] INFO  (SOAP1) W.SOAP1.2: W0000X13: Ended message
processing, rc=0
[2004-07-19T16:28:58:265Z] DEEP (SOAP1) W.SOAP1.2: storing used socket
[2004-07-19T16:28:58:265Z] DEBUG (SOAP1) W.SOAP1.2: entering waitforDocument
[2004-07-19T16:29:03:875Z] DEEP (SOAP1) W.SOAP1.2: cleanup: closing sockets(0)
```

Empty Result From Oracle Application Server Adapter Request

If Oracle Application Server Adapter executes a SOAP request using input parameters passed that do not match records in the target system, then the following SOAP response is generated.

Note: The condition for this adapter does not yield a SOAP fault.

```
[2004-07-19T16:27:05:640Z] DEBUG (SOAP1) W.SOAP1.2: POST received
[2004-07-19T16:27:05:640Z] DEBUG (SOAP1) W.SOAP1.2: in XDSOAPHTTPWorker agentName
is [XDSOAPRouter]
[2004-07-19T16:27:05:640Z] DEBUG (SOAP1) W.SOAP1.2: before parse:
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
<m:GetEffectiveAddress xmlns:m="urn:iwaysoftwar...[590]
[2004-07-19T16:27:05:640Z] ERROR (SOAP1) W.SOAP1.2: Attempting string, no encoding
recognized in document
[2004-07-19T16:27:05:640Z] DEEP (SOAP1) W.SOAP1.2: parse complete in 0 msecs
[2004-07-19T16:27:05:781Z] DEEP (SOAP1) W.SOAP1.2: ST_NODICT
[2004-07-19T16:27:05:781Z] DEEP (SOAP1) W.SOAP1.2: ST_FINISH
[2004-07-19T16:27:05:781Z] DEBUG (SOAP1) extractControl - edaDoc: false
[2004-07-19T16:27:05:781Z] DEBUG (SOAP1) now: 2004-07-19T16:27:05Z expires:
2004-07-20T16:27:05Z
[2004-07-19T16:27:05:781Z] DEBUG (SOAP1) W.SOAP1.2: inside isAsync() the soap
Action is ["B0100033.GetEffectiveAddressRequest#test##"]
[2004-07-19T16:27:05:781Z] DEBUG (SOAP1) W.SOAP1.2: inside isAsync() the soap
Action is [B0100033.GetEffectiveAddressRequest#test##]
[2004-07-19T16:27:05:781Z] DEBUG (SOAP1) W.SOAP1.2: checking for cached agent
[2004-07-19T16:27:05:796Z] DEBUG (SOAP1) W.SOAP1.2: pushagent: adding agent
com.ibi.iwse.XDSOAPRouter
[2004-07-19T16:27:05:796Z] DEBUG (SOAP1) W.SOAP1.2: inside worker the soap Action
is [B0100033.GetEffectiveAddressRequest#test##]
[2004-07-19T16:27:05:796Z] DEBUG (SOAP1) W.SOAP1.2: precedence: 1
[2004-07-19T16:27:05:796Z] DEBUG (SOAP1) W.SOAP1.2: precedence: 1
[2004-07-19T16:27:05:796Z] DEBUG (SOAP1) W.SOAP1.2: numagents: 1
[2004-07-19T16:27:05:812Z] DEBUG (SOAP1) W.SOAP1.2: running agent 1 name
com.ibi.iwse.XDSOAPRouter document 1
[2004-07-19T16:27:05:812Z] INFO  (manager) MGR00X01: Adding active worker:
W.SOAP1.2
[2004-07-19T16:27:05:812Z] DEBUG (SOAP1) W.SOAP1.2: <?xml version="1.0"
encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <SOAP-ENV:Body>
      <m:GetEffectiveAddress
xmlns:m="urn:iwaysoftware:ibse:jul2003:GetEffectiveAddress">
         <m:jdeRequest type="callmethod">
            <m:callMethod name="GetEffectiveAddress">
               <m:params>
                  <m:param name="mnAddressNumber">12345</m:param>
               </m:params>
               <m:onError/>
            </m:callMethod>
         </m:jdeRequest>
      </m:GetEffectiveAddress>
   </SOAP-ENV:Body>
   <SOAPAction agentName="XDSOAPRouter"
cid="9F71FEA4C932CD8786F7388D7EF293A1">B0100033.GetEffectiveAddressRequest#test##<
/SOAPAction>
</SOAP-ENV:Envelope>
[2004-07-19T16:27:05:812Z] DEBUG (SOAP1) W.SOAP1.2: business method:
m:GetEffectiveAddress
[2004-07-19T16:27:05:828Z] DEBUG (SOAP1) W.SOAP1.2: input:
[2004-07-19T16:27:05:828Z] DEBUG (SOAP1) W.SOAP1.2: <?xml version="1.0"
encoding="UTF-8" ?><jdeRequest xmlns:xsd="http://www.w3.org/2001/XMLSchema"
type="callmethod" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><callMethod
name="GetEffectiveAddress"><params><param name="mnAddressNumber">12345</param>
</params><onError/></callMethod></jdeRequest>
[2004-07-19T16:27:07:843Z] DEBUG (SOAP1) W.SOAP1.2: Agent returned success
[2004-07-19T16:27:07:843Z] INFO  (manager) MGR00X02: Removing active worker:
W.SOAP1.2
[2004-07-19T16:27:07:843Z] DEBUG (SOAP1) W.SOAP1.2: doing docTran, docVal,
listTran for agent(1)
[2004-07-19T16:27:07:859Z] DEBUG (SOAP1) W.SOAP1.2: sendToAll reply to XDReply:
[protocol=http */null]
[2004-07-19T16:27:07:859Z] DEBUG (SOAP1) W.SOAP1.2: preemitters from doc: null
[2004-07-19T16:27:07:859Z] DEBUG (SOAP1) W.SOAP1.2: no preemitters, emitting
contents of doc, usestream=false encoding=UTF-8
[2004-07-19T16:27:07:859Z] DEBUG (SOAP1) W.SOAP1.2: writeEntity, len: 643 data:
<?xml version="1.0" encoding="UTF-8" ?><SOAP-ENV:Envelope
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><SOAP-ENV:Body><GetEffective
AddressResponse xmlns="urn:iwaysoftware:ibse:jul2003:GetEffectiveAddress:response"
cid="9F71FEA4C932CD8786F7388D7EF293A1"><jdeResponse user="JDE" type="callmethod"
environment="DV7333"><callMethod name="GetEffectiveAddress"><returnCode code="2"/>
<params><param
name="mnAddressNumber">12345</param></params></callMethod></jdeResponse></GetEffec
tiveAddressResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>
[2004-07-19T16:27:07:859Z] DEBUG (SOAP1) W.SOAP1.2: writeString: HTTP/1.0
[2004-07-19T16:27:07:859Z] DEBUG (SOAP1) W.SOAP1.2: writeString: 200
[2004-07-19T16:27:07:859Z] DEBUG (SOAP1) W.SOAP1.2: writeString: OK
[2004-07-19T16:27:07:859Z] DEBUG (SOAP1) W.SOAP1.2: writeString: Content-Type:
[2004-07-19T16:27:07:859Z] DEBUG (SOAP1) W.SOAP1.2: writeString: text/xml
[2004-07-19T16:27:07:875Z] DEBUG (SOAP1) W.SOAP1.2: writeString: Content-Length:
[2004-07-19T16:27:07:875Z] DEBUG (SOAP1) W.SOAP1.2: writeString: 643
[2004-07-19T16:27:07:875Z] INFO  (SOAP1) W.SOAP1.2: W0000X13: Ended message
processing, rc=0
[2004-07-19T16:27:07:875Z] DEEP (SOAP1) W.SOAP1.2: storing used socket
[2004-07-19T16:27:07:875Z] DEBUG (SOAP1) W.SOAP1.2: entering waitforDocument
```

```
[2004-07-19T16:27:12:781Z] DEEP (SOAP1) W.SOAP1.2: cleanup: closing sockets(0)
```

Invalid Call Method

If an invalid call is made to Oracle Application Server Adapter, then the following SOAP response is generated.

```
[2004-07-19T16:24:34:859Z] DEBUG (SOAP1) W.SOAP1.2: POST received
[2004-07-19T16:24:34:859Z] DEBUG (SOAP1) W.SOAP1.2: in XDSOAPHTTPWorker agentName
is [XDSOAPRouter]
[2004-07-19T16:24:34:859Z] DEBUG (SOAP1) W.SOAP1.2: before parse:
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<SOAP-ENV:Body>
<m:GetEffectiveAddress xmlns:m="urn:iwaysoftwar...[581]
[2004-07-19T16:24:34:859Z] ERROR (SOAP1) W.SOAP1.2: Attempting string, no encoding
recognized in document
[2004-07-19T16:24:34:859Z] DEEP (SOAP1) W.SOAP1.2: parse complete in 0 msecs
[2004-07-19T16:24:34:875Z] DEEP (SOAP1) W.SOAP1.2: ST_NODICT
[2004-07-19T16:24:34:875Z] DEEP (SOAP1) W.SOAP1.2: ST_FINISH
[2004-07-19T16:24:34:875Z] DEBUG (SOAP1) extractControl - edaDoc: false
[2004-07-19T16:24:34:875Z] DEBUG (SOAP1) now: 2004-07-19T16:24:34Z expires:
2004-07-20T16:24:34Z
[2004-07-19T16:24:34:875Z] DEBUG (SOAP1) W.SOAP1.2: inside isAsync() the soap
Action is ["B0100033.GetEffectiveAddressRequest#test##"]
[2004-07-19T16:24:34:875Z] DEBUG (SOAP1) W.SOAP1.2: inside isAsync() the soap
Action is [B0100033.GetEffectiveAddressRequest#test##]
[2004-07-19T16:24:34:875Z] DEBUG (SOAP1) W.SOAP1.2: checking for cached agent
[2004-07-19T16:24:34:875Z] DEBUG (SOAP1) W.SOAP1.2: pushagent: adding agent
com.ibi.iwse.XDSOAPRouter
[2004-07-19T16:24:34:875Z] DEBUG (SOAP1) W.SOAP1.2: inside worker the soap Action
is [B0100033.GetEffectiveAddressRequest#test##]
[2004-07-19T16:24:34:890Z] DEBUG (SOAP1) W.SOAP1.2: precedence: 1
[2004-07-19T16:24:34:890Z] DEBUG (SOAP1) W.SOAP1.2: precedence: 1
[2004-07-19T16:24:34:890Z] DEBUG (SOAP1) W.SOAP1.2: numagents: 1
[2004-07-19T16:24:34:890Z] DEBUG (SOAP1) W.SOAP1.2: running agent 1 name
com.ibi.iwse.XDSOAPRouter document 1
[2004-07-19T16:24:35:031Z] INFO  (manager) MGR00X01: Adding active worker:
W.SOAP1.2
[2004-07-19T16:24:35:031Z] DEBUG (SOAP1) W.SOAP1.2: <?xml version="1.0"
encoding="UTF-8" ?>
<SOAP-ENV:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <SOAP-ENV:Body>
      <m:GetEffectiveAddress
xmlns:m="urn:iwaysoftware:ibse:jul2003:GetEffectiveAddress">
         <m:jdeRequest type="callmethod">
            <m:callMethod name="GetAddress">
               <m:params>
                  <m:param name="mnAddressNumber">34518</m:param>
               </m:params>
               <m:onError/>
            </m:callMethod>
         </m:jdeRequest>
      </m:GetEffectiveAddress>
   </SOAP-ENV:Body>
   <SOAPAction agentName="XDSOAPRouter"
```

```
cid="4C0AD8398CB7A5B4DED18057D963AA44">B0100033.GetEffectiveAddressRequest#test##<
/SOAPAction>
</SOAP-ENV:Envelope>
[2004-07-19T16:24:35:031Z] DEBUG (SOAP1) W.SOAP1.2: business method:
m:GetEffectiveAddress
[2004-07-19T16:24:35:031Z] DEBUG (SOAP1) W.SOAP1.2: input:
[2004-07-19T16:24:35:031Z] DEBUG (SOAP1) W.SOAP1.2: <?xml version="1.0"
encoding="UTF-8" ?><jdeRequest xmlns:xsd="http://www.w3.org/2001/XMLSchema"
type="callmethod" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><callMethod
name="GetAddress"><params><param name="mnAddressNumber">34518</param>
      </params><onError/></callMethod></jdeRequest>
[2004-07-19T16:24:36:781Z] DEBUG (SOAP1) W.SOAP1.2: Agent returned success
[2004-07-19T16:24:36:781Z] INFO  (manager) MGR00X02: Removing active worker:
W.SOAP1.2
[2004-07-19T16:24:36:781Z] DEBUG (SOAP1) W.SOAP1.2: doing docTran, docVal,
listTran for agent(1)
[2004-07-19T16:24:36:781Z] DEBUG (SOAP1) W.SOAP1.2: sendToAll reply to XDReply:
[protocol=http */null]
[2004-07-19T16:24:36:781Z] DEBUG (SOAP1) W.SOAP1.2: preemitters from doc: null
[2004-07-19T16:24:36:781Z] DEBUG (SOAP1) W.SOAP1.2: no preemitters, emitting
contents of doc, usestream=false encoding=UTF-8
[2004-07-19T16:24:36:781Z] DEBUG (SOAP1) W.SOAP1.2: writeEntity, len: 595 data:
<?xml version="1.0" encoding="UTF-8" ?><SOAP-ENV:Envelope
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><SOAP-ENV:Body><GetEffective
AddressResponse xmlns="urn:iwaysoftware:ibse:jul2003:GetEffectiveAddress:response"
cid="4C0AD8398CB7A5B4DED18057D963AA44"><jdeResponse user="JDE" type="callmethod"
environment="DV7333"><callMethod name="GetAddress"><returnCode code="99"/><params>
</params></callMethod></jdeResponse></GetEffectiveAddressResponse></SOAP-ENV:Body>
</SOAP-ENV:Envelope>
[2004-07-19T16:24:36:796Z] DEBUG (SOAP1) W.SOAP1.2: writeString: HTTP/1.0
[2004-07-19T16:24:36:796Z] DEBUG (SOAP1) W.SOAP1.2: writeString: 200
[2004-07-19T16:24:36:796Z] DEBUG (SOAP1) W.SOAP1.2: writeString: OK
[2004-07-19T16:24:36:796Z] DEBUG (SOAP1) W.SOAP1.2: writeString: Content-Type:
[2004-07-19T16:24:36:796Z] DEBUG (SOAP1) W.SOAP1.2: writeString: text/xml
[2004-07-19T16:24:36:796Z] DEBUG (SOAP1) W.SOAP1.2: writeString: Content-Length:
[2004-07-19T16:24:36:796Z] DEBUG (SOAP1) W.SOAP1.2: writeString: 595
[2004-07-19T16:24:36:796Z] INFO  (SOAP1) W.SOAP1.2: W0000X13: Ended message
processing, rc=0
[2004-07-19T16:24:36:796Z] DEEP (SOAP1) W.SOAP1.2: storing used socket
[2004-07-19T16:24:36:812Z] DEBUG (SOAP1) W.SOAP1.2: entering waitforDocument
[2004-07-19T16:24:42:671Z] DEEP (SOAP1) W.SOAP1.2: cleanup: closing sockets(0)
```

8 Advanced User Tools

This chapter includes the following topics:

- Web Services Policy-Based Security
- Migrating Repositories

Web Services Policy-Based Security

OracleAS Adapter Application Explorer (Application Explorer) provides a security model called Web services policy-based security. This section describes how the feature works and how to configure it.

Web services provide a layer of abstraction between the back-end business logic and the user or application running the Web service. This enables easy application integration but raises the issue of controlling the use and implementation of critical and sensitive business logic that is run as a Web service.

Application Explorer controls the use of Web services that use adapters, using a feature called policy-based security. This feature enables an administrator to apply "policies" to Business Services (Web services) to deny or permit their execution.

A policy is a set of privileges dealing with the execution of a Business Service (BS) that can be applied to an existing or new BS. When you set specific rights or privileges inside a policy, you do not have to re-create privileges for every BS that has security concerns in common with other Business Services. Instead, you reuse a policy on multiple Business Services.

The goal of the feature is to secure requests at both the transport and the SOAP request level transmitted on the wire. Some of the policies do not deal with security issues directly, but do affect the runtime behavior of the Web services to which they have been applied.

The Business Services administrator creates an "instance" of a policy type, names it, associates individual users or groups (a collection of users), and then applies that policy to one or more Business Services.

You can assign a policy to a Business Service, or to a method within a Business Service. If a policy is only applied to a method, other methods in that Business Service will not be governed by it. However, if a policy is applied to the Business Service, all methods are governed by it. At runtime, the user ID and password that are sent to OracleAS Adapter Business Services Engine (BSE) in the SOAP request message are verified against the list of users for all policies applied to that specific Business Service. The policy type that is supported is Resource Execution, which dictates who can or cannot execute the Business Service.

When a policy is not applied, the default value for a Business Service is to "grant all". For example, anybody can execute the Business Service, until the Resource Execution policy is associated to the Business Service. At that time, only those granted execution permissions, or users not part of the group that has been denied execution permissions, have access to the Business Service.

Configuring Web Services Policy-Based Security

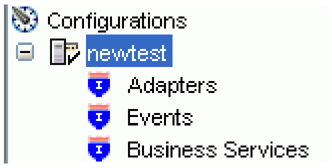
The following procedures describe how to configure Web services policy-based security.

Creating and Associating a User with a Policy

Before you create instances of policies, you must have a minimum of one user or one group to associate to an instance. You can create users and groups using Application Explorer.

1. Start Application Explorer.
2. Right-click the configuration to which you want to connect, for example, **newtest**. See Chapter 2, "Configuring OracleAS Adapter for J.D. Edwards One World" for information on creating a new configuration.
3. Select **Connect**.

   Nodes appear for Adapters, Events, and Business Services (also known as Web services).



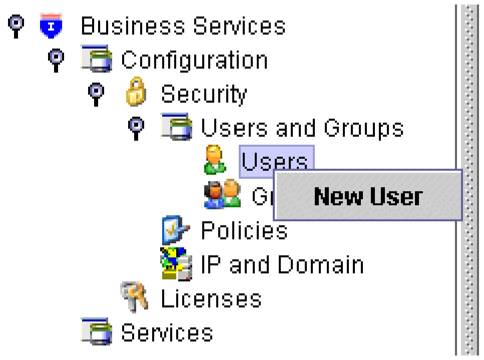
   Perform the following steps:

   a. Expand the **Business Services** node.

   b. Expand the **Configuration** node.

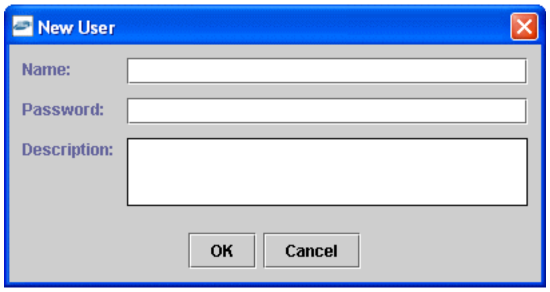
   c. Expand the **Security** node.

   d. Expand the **Users and Groups** node.



4. Right-click **Users** and click **New User**.

The New User dialog box is displayed.



Perform the following steps:

a. In the **Name** field, enter a user ID.

b. In the **Password** field, enter the password associated with the user ID.

c. In the **Description** field, enter a description of the user (optional).

5. Click **OK**.

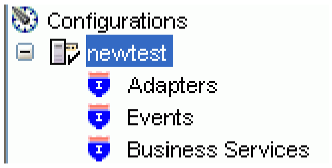
The new user is added under the Users node.

Creating a Group to Use with a Policy

To create a group to use with a policy:

1. Start Application Explorer.
2. Right-click the configuration to which you want to connect, for example, **newtest**. See Chapter 2, "Configuring OracleAS Adapter for J.D. Edwards One World" for information on creating a new configuration.
3. Select **Connect**.

   Nodes appear for Adapters, Events, and Business Services (also known as Web services).



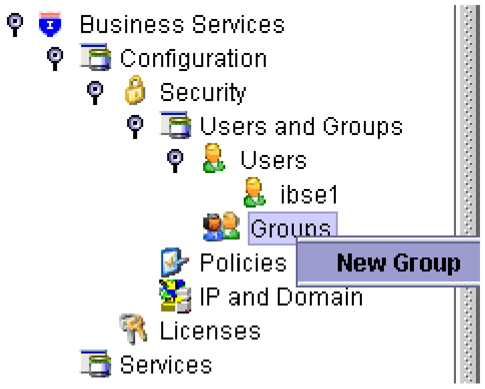
Perform the following steps:

a. Expand the **Business Services** node.

b. Expand the **Configuration** node.

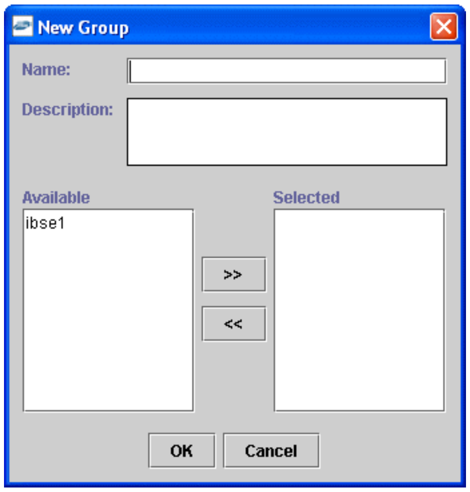
c. Expand the **Security** node.

d. Expand the **Users and Groups** node.



4. Right-click **Groups** and select **New Group**.

   The New Group dialog box is displayed.



   Perform the following steps:

   a. In the **Name** field, enter a name for the group.

   b. In the **Description** field, enter a description for the group (optional).

   c. From the available list of users in the left pane, select one or more users and add them to the **Selected** list by clicking the double right facing arrow.

5. When you have selected at least one user, click **OK**.

The new group is added under the Groups node.

Creating an Execution Policy

An execution policy determines who can execute the Business Services to which the policy is applied.

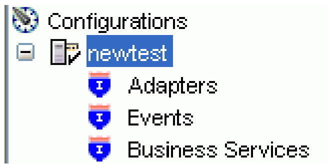
To create an execution policy:

1. Start Application Explorer.

2. Right-click the configuration to which you want to connect, for example, SampleConfig. See Chapter 2, "Configuring OracleAS Adapter for J.D. Edwards One World" for information on creating a new configuration.

3. Select **Connect**.

   Nodes appear for Adapters, Events, and Business Services (also known as Web services).



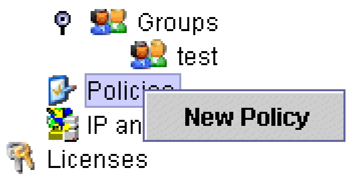
   Perform the following steps:

   a. Expand the **Business Services**node.

   b. Expand the **Configuration** node.

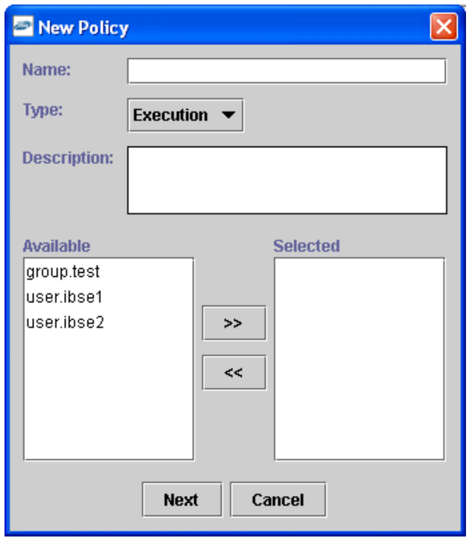
   c. Expand the **Security** node.

   d. Expand the **Policies** node.



4. Right-click **Policies** and select **New Policy**.

   The New Policy dialog box is displayed.



   Perform the following steps:

   a. In the **Name** field, enter a name for the policy.

   b. From the **Type** list, select **Execution**.

   c. In the **Description** field, enter a description for the policy (optional).

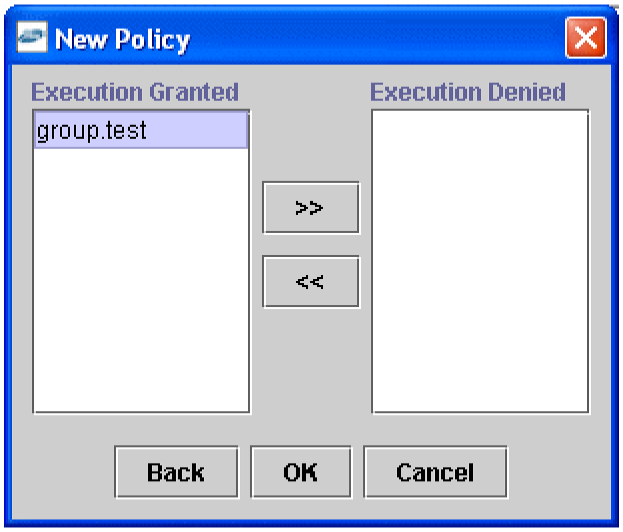
   d. From the available list of users in the left pane, select one or more users and add them to the **Selected** list by clicking the double right facing arrow.

Note: This user ID is verified against the value in the user ID element of the SOAP header sent to BSE in a SOAP request.

---

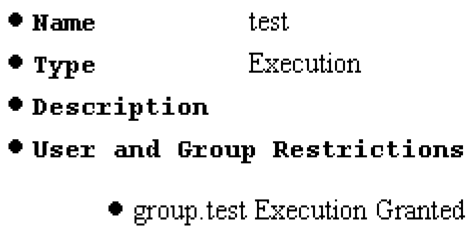
5. When you have selected at least one user selected, click **OK**.
6. Click **Next**.

The New Policy permissions dialog box is displayed.



7. To grant permission to a user or group to execute a Business Service, select the user or group and move them into the **Execution Granted** list by selecting the double left facing arrow.
8. To deny permission to a user or group to execute a Business Service, select the user or group and move them into the **Execution Denied** list by selecting the double right facing arrow.
9. Click **OK**.

   The following pane summarizes your configuration.



- **Name** test
- **Type** Execution
- **Description**
- **User and Group Restrictions**
    - group.test Execution Granted

Using the IP and Domain Restrictions Policy Type

You configure the IP and Domain Restriction policy type slightly differently from other policy types. The IP and Domain Restriction policy type controls connection access to BSE and therefore need not be applied to individual Web services. You need not create a policy, however, you must enable the Security Policy option in Application Explorer.

1. Start Application Explorer.

2. Right-click the configuration to which you want to connect, for example, SampleConfig. See Chapter 2, "Configuring OracleAS Adapter for J.D. Edwards One World" for information on creating a new configuration.
3. Select **Connect**.

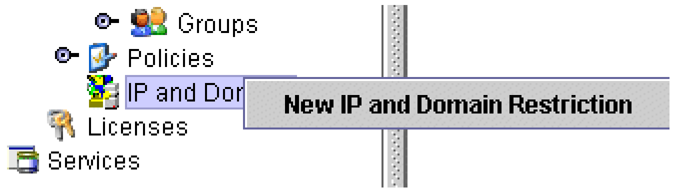
   Nodes appear for Adapters, Events, and Business Services (also known as Web services).

   Perform the following steps:

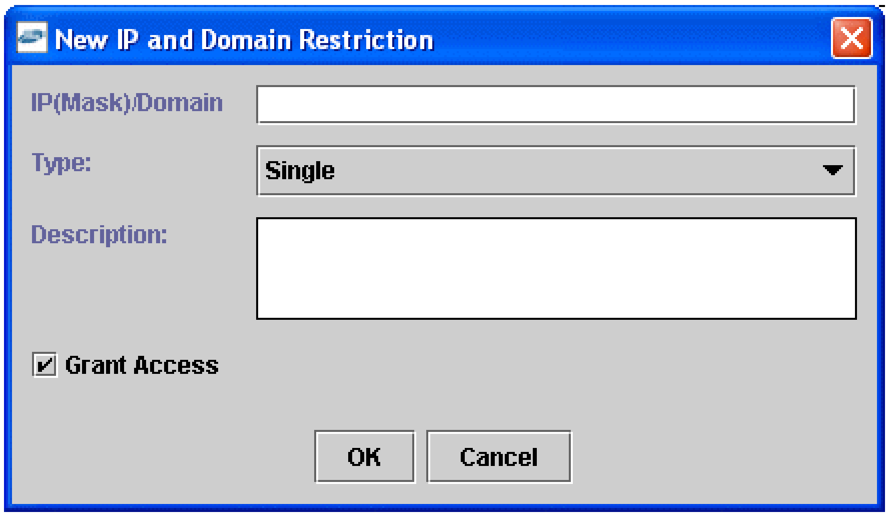
   a. Expand the **Business Services** node.

   b. Expand the **Configuration** node.

   c. Expand the **Security** node.
4. Right-click **IP and Domain** and select **New IP and Domain Restriction**.



   The New IP and Domain Restriction dialog box is displayed.



IP(Mask)/Domain

Type: Single

Description:

☑ Grant Access

   Perform the following steps:

   a. In the **IP(Mask)/Domain** field, enter the IP or domain name using the following guidelines.

      If you select **Single** (Computer) from the **Type** list, you must provide the IP address for that computer. If you only know the DNS name for the computer, click **DNS Lookup** to obtain the IP Address based on the DNS name.

      If you select **Group** (of Computers), you must provide the IP address and subnet mask for the computer group.

      If you select **Domain**, you must provide the domain name.

   b. From the **Type** list, select the type of restriction.

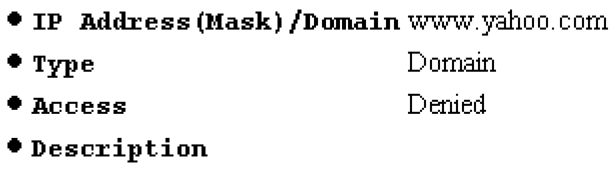
   c. In the **Description** field, enter a description (optional).

d. To grant access, select the **Grant Access** check box.

5. Click **OK**.

The new domain is added under the IP and Domain node.

The following pane summarizes your configuration.



● **IP Address(Mask)/Domain**	www.yahoo.com
● **Type**	Domain
● **Access**	Denied
● **Description**	

Migrating Repositories

During design time, the Oracle repository is used to store metadata created when using Application Explorer to configure adapter connections, browse EIS objects, configure services, and configure listeners to listen for EIS events. The information in the repository is also referenced at runtime. For management purposes, you can migrate BSE and J2CA repositories that are configured for Oracle to new destinations without affecting your existing configuration. For example, you may want to migrate a repository from a test environment to a production environment.

Migrating a BSE Repository

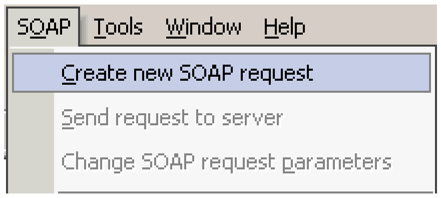
To migrate a BSE repository:

1. Copy the BSE control service URL, for example:

```
http://localhost:7777/ibse/IBSEServlet/admin/iwcontrol.ibs
```

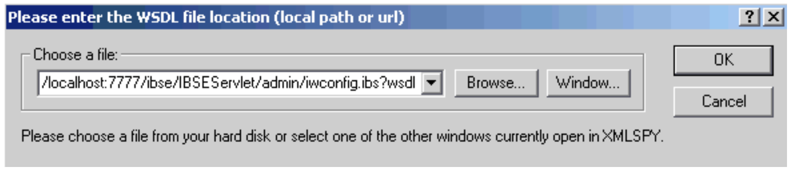
2. Open a third party XML editor, for example, XMLSPY.
3. From the menu bar, click **SOAP**.

   A list of options appears.



4. Select **Create new SOAP request**.

   The WSDL file location dialog box is displayed.



Please enter the WSDL file location (local path or url)

Choose a file: /localhost:7777/ibse/IBSEServlet/admin/iwconfig.ibs?wsdl

Please choose a file from your hard disk or select one of the other windows currently open in XMLSPY.

Perform the following steps:

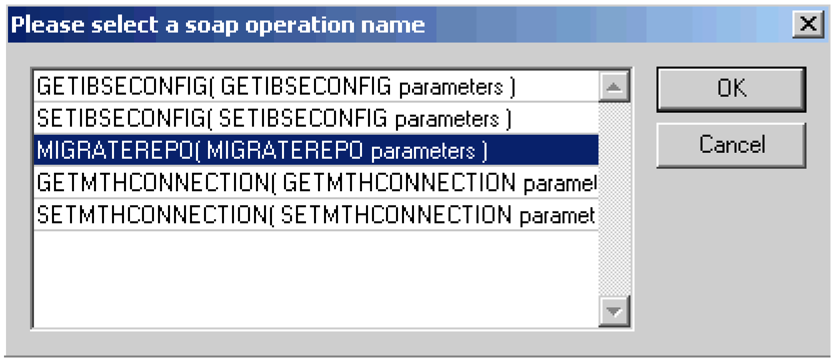
a. In the **Choose a file** field, paste the BSE control service URL.

b. Append **?wsdl** to the URL, for example:

```
http://localhost:7777/ibse/IBSEServlet/admin/iwcontrol.ibs?wsdl
```

5. Click **OK**.

The soap operation name dialog box is displayed and lists the available control methods.



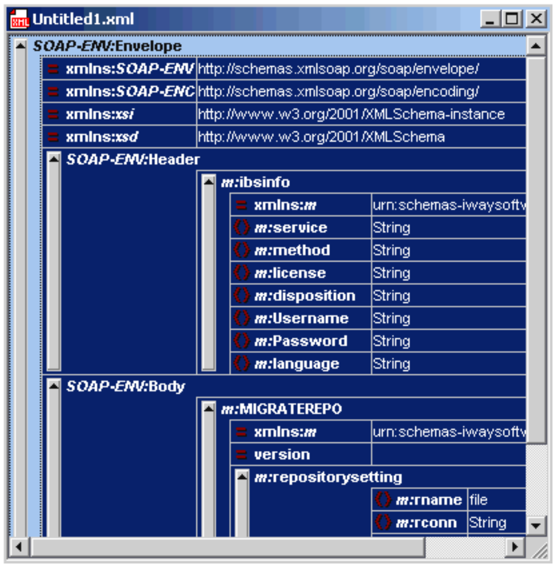
Please select a soap operation name

GETIBSECONFIG(GETIBSECONFIG parameters)
SETIBSECONFIG(SETIBSECONFIG parameters)
MIGRATEREPO(MIGRATEREPO parameters)
GETMTHCONNECTION(GETMTHCONNECTION paramel
SETMTHCONNECTION(SETMTHCONNECTION paramet

6. Select the **MIGRATEREPO(MIGRATEREPO parameters)** control method and click **OK**.

Note: The **MIGRATEREPO(MIGRATEREPO parameters)** control method is available from the BSE administration console. This control method migrates all Web services to the new (empty) repository. You can choose to migrate select Web services only.

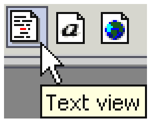
The following window shows the structure of the SOAP envelope.



Untitled1.xml

SOAP-ENV:Envelope
xmlns:SOAP-ENV http://schemas.xmlsoap.org/soap/envelope/
xmlns:SOAP-ENC http://schemas.xmlsoap.org/soap/encoding/
xmlns:xsi http://www.w3.org/2001/XMLSchema-instance
xmlns:xsd http://www.w3.org/2001/XMLSchema
SOAP-ENV:Header
m:ibsinfo
xmlns:m urn:schemas-iwaysoftv
m:service String
m:method String
m:license String
m:disposition String
m:Username String
m:Password String
m:language String
SOAP-ENV:Body
m:MIGRATEREPO
xmlns:m urn:schemas-iwaysoftv
version
m:repositorysetting
m:rname file
m:rconn String

7. Locate the **Text view** icon in the toolbar.



8. To display the structure of the SOAP envelope as text, click the **Text view** icon.

   The <SOAP-ENV:Header> tag is not required and can be deleted from the SOAP envelope.

9. Locate the following section:

```
<m:MIGRATEREPO xmlns:m="urn:schemas-iwaysoftware-com:jul2003:ibse:config"
version="">
<m:repositorysetting>
<m:rname>oracle</m:rname>
<m:rconn>String</m:rconn>
<m:rdriver>String</m:rdriver>
<m:ruser>String</m:ruser>
<m:rpwd>String</m:rpwd>
</m:repositorysetting>
<m:servicename>String</m:servicename>
</m:MIGRATEREPO>
```

   Perform the following steps:

   a. For the `<m:rconn>` tag, replace the String placeholder with a repository URL where you want to migrate your existing BSE repository.

      The Oracle repository URL has the following format:

```
jdbc:oracle:thin:@[host]:[port]:[sid]
```

   b. For the `<m:rdriver>` tag, replace the String placeholder with the location of your Oracle driver.

   c. For the `<m:ruser>` tag, replace the String placeholder with a valid user name to access the Oracle repository.

   d. For the `<m:rpwd>` tag, replace the String placeholder with a valid password to access the Oracle repository.

10. Perform one of the following migration options.

   - If you want to migrate a single Web service from the current BSE repository, enter the Web service name in the `<m:servicename>` tag, for example:

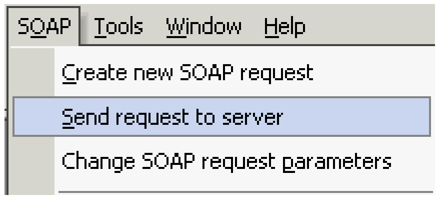
```
<m:servicename>JDEService1</m:servicename>
```

   - If you want to migrate multiple Web services from the current BSE repository, duplicate the `<m:servicename>` tag for each Web service, for example:

```
<m:servicename>JDEService1</m:servicename>
<m:servicename>JDEService2</m:servicename>
```

   - If you want to migrate all Web services from the current BSE repository, remove the `<m:servicename>` tag.

11. From the menu bar, click **SOAP** and select **Send request to server**.



Your BSE repository and any Web services you specified are now migrated to the new Oracle repository URL you specified.

Migrating a J2CA Repository

To migrate a J2CA repository:

1. Navigate to the location of your J2CA configuration directory where the repository schemas and other information is stored, for example:

   ```
   OracleAS_home\adapters\application\config\JCA_CONFIG
   ```

   Where `JCA_CONFIG` is the name of your J2CA configuration.

2. Locate and copy the `repository.xml` file.

3. Place this file in a new J2CA configuration directory to migrate the existing repository.

   Your J2CA repository is migrated to the new J2CA configuration directory.

A

Configuring J.D. Edwards OneWorld for Outbound Transaction Processing

J.D. Edwards OneWorld enables you to specify outbound functionality for Master Business Functions (MBF).

The following topics describe how to enable outbound transaction processing in J.D. Edwards OneWorld and how to modify the `jde.ini` file for XML support.

- Specifying Outbound Functionality for a Business Function
- Modifying the OneWorld jde.ini File

Specifying Outbound Functionality for a Business Function

You can specify outbound functionality for business functions and manage the flow of data. You enable outbound transaction processing using a processing option that controls how a transaction is written.

Outbound Transaction Processing

To process outbound data, you use the:

- Data Export Control table
- Processing Log table

The Data Export Control table manages the flow of the outbound data to third-party applications. The Processing Log table contains all the information about the OneWorld event.

For more information on configuring J.D. Edwards OneWorld for outbound processing, see "Detailed Tasks for OneWorld Operations" in the *J.D. Edwards Interoperability Guide for OneWorld XE*.

Enabling Outbound Transaction Processing

To enable outbound transaction processing:

1. Right-click the application that contains the processing options for the Master Business Functions of the transaction.

   For a list of these options, see Appendix B of the*J.D. Edwards Interoperability Guide for OneWorld XE*.

2. From the shortcut menu, select **Prompt for Values**. Click either the **Outbound** tab or the **Interop** tab.
3. Enter the transaction type.

The OneWorld event listener processes only the after image for the business function.

You are not required to set the before image function.

The Data Export Control Table and the Processing Log Table

The Data Export Control table manages the flow of the outbound data to third-party applications. OneWorld allows the subscription of multiple vendor-specific objects for an interoperability transaction.

The records in the Data Export Control table are used to determine the vendor-specific objects to call from the Outbound Subsystem batch process (R00460) or the Outbound Scheduler batch process (R00461).

The Processing Log table contains all the information about the OneWorld event including the transaction type, order type, and sequence number from the Data Export Control table.

Using the Export Controls

To use the data export controls:

1. On the Work With Data Export Controls pane, click **Add**.
2. Enter values in the Transaction Type and Order Type fields.
3. For each detail row, enter either a batch process name or version or a function name and the library.
4. To launch the vendor-specific object for an add or insert, enter **1**.
5. For the update, delete, and inquiry actions, enter **1**.
6. In the Launch Immediately column, enter **1**.
7. Click **OK**.

Modifying the OneWorld jde.ini File

Because OracleAS Adapter for J.D. Edwards OneWorld uses XML for the transfer of information to and from J.D. Edwards OneWorld, you must configure the OneWorld environment to support XML. You can do this easily by modifying the OneWorld `jde.ini` file.

Modifying a jde.ini File for XML Support

The following is an example of how to modify a `jde.ini` file to implement XML support.

1. Add the following blocks:

```
[JDENET_KERNEL_DEF6]
;krnlName=CALL OBJECT KERNEL
;dispatchDLLName=jdekrnl.dll
;dispatchDLLFunction=_JDEK_DispatchCallObjectMessage@28
;maxNumberOfProcesses=10
;numberOfAutoStartProcesses=0
krnlName=CALL OBJECT KERNEL
dispatchDLLName=XMLCallObj.dll
dispatchDLLFunction=_XMLCallObjectDispatch@28
maxNumberOfProcesses=10
numberOfAutoStartProcesses=0
[JDENET_KERNEL_DEF15]
```

```
krnlName=XML TRANSACTION KERNEL
dispatchDLLName=XMLTransactions.dll
dispatchDLLFunction=_XMLTransactionDispatch@28
maxNumberOfProcesses=1
numberOfAutoStartProcesses=1
```

The parameters containing an underscore (_) and @28 are for Windows operating systems only. For other operating systems, replace the parameters with the values in the following table.

Operating System	Call Object dispatch DLLName	XML Trans dispatch DLLName
AS400	XMLCALLOBJ	XMLTRANS
HP9000B	libxmlcallojb.sl	libxmltransactions .lo
Sun or RS6000	libxmlcallojb.so	libxmltransactions .so

2. Change the following block:

```
[JDENET]
serviceNameListen=6009
serviceNameConnect=6009
maxNetProcesses=5
maxNetConnections=400
maxKernelProcesses=50
maxKernelRanges=15
netTrace=1
ServiceControlRefresh=5
MonitorOption=0 0 0 0 0 0 0 0
```

Change maxKernelRanges to 15.

For more information on establishing your J.D. Edwards OneWorld environment for XML support, see "Setting the jde.ini File for XML" in the *J.D. Edwards Interoperability Guide for OneWorld XE*.

B Sample Files

The Adapter for J.D. Edwards OneWorld supports the jdeRequest and jdeResponse XML structures for executing business functions within OneWorld. Using J.D. Edwards OneWorld XML, you can:

- Aggregate business function calls into a single object.
- Use the J.D. Edwards OneWorld ThinNet API.
- Access both Z files and business functions.

The following topics provide examples of the jdeRequest and jdeResponse XML structures for executing business functions within OneWorld:

- Issuing a Single-Function Request
- Issuing a Multiple-Function Request
- Sample Sales Order Request
- Sample Sales Order Response

Issuing a Single-Function Request

The following example, GetEffectiveAddress, is a single-function call to J.D. Edwards OneWorld, and the result of this request is a standard jdeResponse document. In a single-function request, only one callMethod within the XML object is specified.

Executing a Business Function with a Single-Function Call

The following is a sample GetEffectiveAddress jdeRequest.

```
<jdeRequest type="callmethod" user="JDE" pwd="JDE" environment="DV7333"
session="">
<callMethod name="GetEffectiveAddress" app="BSE" runOnError="no">
<params>
   <param name="mnAddressNumber">1001</param>
   <param name="jdDateBeginningEffective"></param>
   <param name="cEffectiveDateExistence10"></param>
   <param name="szAddressLine1"></param>
   <param name="szAddressLine2"></param>
   <param name="szAddressLine3"></param>
   <param name="szAddressLine4"></param>
   <param name="szZipCodePostal"></param>
   <param name="szCity"></param>
   <param name="szCountyAddress"></param>
   <param name="szState"></param>
   <param name="szCountry"></param>
   <param name="szUserid"></param>
   <param name="szProgramid"></param>
   <param name="jdDateupdated"></param>
   <param name="szWorkstationid"></param>
   <param name="mnTimelastupdated"></param>
   <param name="szNamealpha"></param>
</params>
<onError abort="yes"></onError>
</callMethod>
</jdeRequest>
```

The following is a sample GetEffectiveAddress jdeResponse.

```
<?xml version="1.0"?>

<!DOCTYPE jdeResponse>
<jdeResponse environment="DV7333"
             pwd="JDE"
             session="516.1029417972.68"
             type="callmethod"
             user="JDE">
  <callMethod app="BSE"
              name="GetEffectiveAddress"
              runOnError="no">
    <returnCode code="0"/>
    <params>
      <param name="mnAddressNumber">1001</param>
      <param name="jdDateBeginningEffective"/>
      <param name="cEffectiveDateExistence10"/>
       <param name="szAddressLine1">8055 Tufts Avenue, Suite 1331
</param>
      <param name="szAddressLine2">
</param>
      <param name="szAddressLine3">
</param>
      <param name="szAddressLine4">
</param>
      <param name="szZipCodePostal">80237       </param>
      <param name="szCity">Denver                   </param>
      <param name="szCountyAddress">                         </param>
      <param name="szState">CO</param>
      <param name="szCountry"/>
      <param name="szUserid"/>
      <param name="szProgramid"/>
      <param name="jdDateupdated"/>
      <param name="szWorkstationid"/>
      <param name="mnTimelastupdated">0</param>
      <param name="szNamealpha">J.D. Edwards & Company                  </param>
    </params>
  </callMethod>
</jdeResponse>
```

Issuing a Multiple-Function Request

The following example, GetEffectiveAddress, is a multiple-function call to J.D. Edwards OneWorld, and the result of this request is a standard jdeResponse document with multiple sections. In a multiple-function request, more than one callMethod within the XML object is specified.

Executing a Business Function with a Multiple-Function Call

The following is a sample Purchase Order in the jdeRequest format. The XML contains return parameter specifications, as well as file cleanup logic.

```
<?xml version='1.0' encoding='utf-8' ?>
<jdeRequest pwd='password' type='callmethod' user='user' session=''
environment='DV7333' sessionidle=''>
   <callMethod app='XMLTest' name='GetLocalComputerId'
      runOnError='no'>
   <params>
      <param name='szMachineKey' id='machineKey'></param>
   </params>
   <onError abort='yes'>
   </onError>
   </callMethod>
   <callMethod app='XMLTest' name='F4311InitializeCaching'
      runOnError='no'>
   <params>
      <param name='cUseWorkFiles'>2</param>
   </params>
   </callMethod>
   <callMethod app='XMLTest' name='F4311FSBeginDoc' runOnError='no'
      returnNullData='yes'>
   <params>
      <param name='mnJobNumber' id='jobNumber'></param>
      <param name='szComputerID' idref='machineKey'></param>
      <param name='cHeaderActionCode'>A</param>
      <param name='cProcessEdits'>1</param>
      <param name='cUpdateOrWriteToWorkFile'>2</param>
      <param name='cRecordWrittenToWorkFile'>0</param>
      <param name='szOrderCOmpany' id='orderCompany'>00200</param>
      <param name='szOrderType'>OP</param>
      <param name='szOrderSuffix'>000</param>
      <param name='szBranchPlant'>         M30</param>
      <param name='mnSupplierNumber'
         id='supplierNumber'>4343</param>
      <param name='mnShipToNumber'>0.0</param>
      <param name='jdOrderDate'>2000/03/02</param>
      <param name='cEvaluatedReceiptsFlag'>N</param>
      <param name='cCurrencyMode'>D</param>
      <param name='szTransactionCurrencyCode'>USD</param>
      <param name='mnCurrencyExchangeRate'>0.0</param>
      <param name='szOrderedPlacedBy'>SUBSTITUTE</param>
      <param name='szProgramID'>EP4310</param>
      <param name='szPurchaseOrderPrOptVersion'
         id='Version'>ZJDE0001</param>
      <param name='szUserID'>SUBSTITUTE</param>
      <param name='mnProcessID' id='processID'></param>
      <param name='mnTransactionID' id='transactionID'></param>
   </params>
   <onError abort='yes'>
   <callMethod app='XMLTest' name='F4311ClearWorkFiles'
      runOnError='yes' returnNullData='yes'>
   <params>
      <param name='szComputerID' idref='jobNumber'></param>
      <param name='mnJobNumber' idref='machineKey'></param>
      <param name='cClearHeaderFile'>1</param>
      <param name='cClearDetailFile'>1</param>
      <param name='mnLineNumber'>0</param>
      <param name='cUseWorkFiles'>2</param>
      <param name='mnProcessID' idref='processID'></param>
      <param name='mnTransactionID' idref='transactionID'></param>
   </params>
   </callMethod>
   </onError>
   </callMethod>
   <!-- This is the first EditLine entry -->
   <callMethod app='XMLTest' name='F4311EditLine' runOnError='yes'
      returnNullData='no'>
   <params>
      <param name='mnJobNumber' idref='jobNumber'></param>
      <param name='szComputerID' idref='machineKey'></param>
      <param name='cDetailActionCode'>A</param>
      <param name='cProcessEdits'>1</param>
      <param name='cUpdateOrWriteWorkFile'>2</param>
      <param name='cCurrencyProcessingFlag'>Y</param>
      <param name='szPurchaseOrderPrOptVersion'
      idref='version'></param>
      <param name='szOrderCompany' idref='orderCompany'></param>
      <param name='szOrderType'>OP</param>
      <param name='szOrderSuffix'>000</param>
      <param name='szBranchPlant'>         M30</param>
      <param name='mnSupplierNumber'
         idref='supplierNumber'></param>
      <param name='mnShipToNumber'>0.0</param>
      <param name='jdRequestedDate'>2000/03/02</param>
      <param name='jdTransactionDate'>2000/03/02</param>
      <param name='jdPromisedDate'>2000/03/02</param>
      <param name='jdGLDate'>2000/03/02</param>
      <param name='szUnformattedItemNumber'>1001</param>
      <param name='mnQuantityOrdered'>1</param>
      <param name='szDetailLineBranchPlant'>         M30</param>
      <param name='szLastStatus'>220</param>
      <param name='szNextStatus'>230</param>
      <param name='cEvaluatedReceipts'>N</param>
      <param name='szTransactionCurrencyCode'>USD</param>
      <param name='cSourceRequestingPOGeneration'>0</param>
      <param name='szProgramID'>XMLTest</param>
      <param name='szUserID'>SUBSTITUTE</param>
      <param name='szAgreementNumber'></param>
      <param name='mnAgreementSupplement'>0</param>
      <param name='jdEffectiveDate'></param>
      <param name='szPurchasingCostCenter'></param>
      <param name='szObjectAccount'></param>
      <param name='szSubsidiary'></param>
      <param name='mnProcessID' idref='processID'></param>
      <param name='mnTransactionID' idref='transactionID'></param>
   </params>
   </callMethod>
   <!-- This is the second EditLine entry -->
   <callMethod app='XMLTest' name='F4311EditLine' runOnError='yes'
      returnNullData='no'>
   <params>
      <param name='mnJobNumber' idref='jobNumber'></param>
      <param name='szComputerID' idref='machineKey'></param>
      <param name='cDetailActionCode'>A</param>
      <param name='cProcessEdits'>1</param>
      <param name='cUpdateOrWriteWorkFile'>2</param>
      <param name='cCurrencyProcessingFlag'>Y</param>
      <param name='szPurchaseOrderPrOptVersion'
```

```
         idref='version'></param>
      <param name='szOrderCompany' idref='orderCompany'></param>
      <param name='szOrderType'>OP</param>
      <param name='szOrderSuffix'>000</param>
      <param name='szBranchPlant'>         M30</param>
      <param name='mnSupplierNumber'
         idref='supplierNumber'></param>
      <param name='mnShipToNumber'>0.0</param>
      <param name='jdRequestedDate'>2000/03/02</param>
      <param name='jdTransactionDate'>2000/03/02</param>
      <param name='jdPromisedDate'>2000/03/02</param>
      <param name='jdGLDate'>2000/03/02</param>
      <param name='szUnformattedItemNumber'>2001</param>
      <param name='mnQuantityOrdered'>3</param>
      <param name='szDetailLineBranchPlant'>         M30</param>
      <param name='szLastStatus'>220</param>
      <param name='szNextStatus'>230</param>
      <param name='cEvaluatedReceipts'>N</param>
      <param name='szTransactionCurrencyCode'>USD</param>
      <param name='cSourceRequestingPOGeneration'>0</param>
      <param name='szProgramID'>XMLTest</param>
      <param name='szUserID'>SUBSTITUTE</param>
      <param name='szAgreementNumber'></param>
      <param name='mnAgreementSupplement'>0</param>
      <param name='jdEffectiveDate'></param>
      <param name='szPurchasingCostCenter'></param>
      <param name='szObjectAccount'></param>
      <param name='szSubsidiary'></param>
      <param name='mnProcessID' idref='processID'></param>
      <param name='mnTransactionID' idref='transactionID'></param>
   </params>
   </callMethod>
   <callMethod app='XMLTest' name='F4311EditDoc' runOnError='no'
      returnNullData='no'>
   <params>
      <param name='szOrderSuffix'>000</param>
      <param name='szComputerID' idref='machineKey'></param>
      <param name='mnJobnumber' idref='jobNumber'></param>
      <param name='mnAddressNumber' idref='supplierNumber'></param>
      <param name='szOrderType'>OP</param>
      <param name='szOrderCompany' idref='orderCompany'></param>
      <param name='szVersionProcOption' idref='version'></param>
      <param name='cActionCode'>A</param>
      <param name='mnProcessID' idref='processID'></param>
      <param name='mnTransactionID' idref='transactionID'></param>
   </params>
   </callMethod>
   <callMethod app='XMLTest' name='F4311EndDoc' runOnError='no'
      returnNullData='no'>
   <params>
      <param name='szComputerID' idref='machineKey'></param>
      <param name='mnJobNumber' idref='jobNumber'></param>
      <param name='szCallingApplicationName'>XMLTest</param>
      <param name='szVersion' idref='version'></param>
      <param name='szUserID'>SUBSTITUTE</param>
      <param name='mnOrderNumberAssigned'
         id='orderNumberAssigned'></param>
      <param name='cUseWorkFiles'>2</param>
      <param name='cConsolidateLines'>0</param>
      <param name='mnProcessID' idref='processID'></param>
      <param name='mnTransactionID' idref='transactionID'></param>
   </params>
   </callMethod>
   <returnParams runOnError='yes' returnNullData='no'>
      <param name='JobNumber' idref='machineKey'></param>
      <param name='ComputerID' idref='jobNumber'></param>
      <param name='OrderNumberAssigned'
         idref='orderNumberAssigned'></param>
   </returnParams>
   <!-- This is a default error catch for the entire document-->
   <onError abort='yes'>
   <callMethod app='XMLTest' name='F4311ClearWorkFiles'
      runOnError='yes' returnNullData='no'>
   <params>
      <param name='szComputerID' idref='jobNumber'></param>
      <param name='mnJobNumber' idref='machineKey'></param>
      <param name='cClearHeaderFile'>1</param>
      <param name='cClearDetailFile'>1</param>
      <param name='mnLineNumber'>0</param>
      <param name='cUseWorkFiles'>2</param>
      <param name='mnProcessID' idref='processID'></param>
      <param name='mnTransactionID' idref='transactionID'></param>
   </params>
   </callMethod>
   </onError>
</jdeRequest>
```

The Purchase Order response document contains individual return codes for each callMethod executed. In addition, this method returns the order number assigned for the Purchase Order.

```
<?xml version="1.0" encoding="utf-8" ?>

<jdeResponse environment="DV7333" user="JDE" type="callmethod" sessionidle=""
session="2612.1026498135.5" pwd="JDE">
   <callMethod name="GetLocalComputerId" runOnError="no" app="XMLTest">
   <returnCode code="0"/>
   <params>
      <param name="szMachineKey" id="machineKey">XEENT</param>
   </params>
   </callMethod>
   <callMethod name="F4311InitializeCaching" runOnError="no" app="XMLTest">
   <returnCode code="0"/>
   <params>
      <param name="cUseWorkFiles">2</param>
   </params>
   </callMethod>
   <callMethod name="F4311FSBeginDoc" returnNullData="yes" runOnError="no"
app="XMLTest">
   <returnCode code="0"/>
   <params>
      <param name="mnJobNumber" id="jobNumber">3</param>
      <param name="szComputerID" idref="machineKey">XEENT</param>
      <param name="cHeaderActionCode">1</param>
      <param name="cProcessEdits">1</param>
      <param name="cUpdateOrWriteToWorkFile">2</param>
      <param name="cRecordWrittenToWorkFile">1</param>
      <param name="cCurrencyProcessingFlag">Z</param>
      <param name="szOrderCOmpany" id="orderCompany">00200</param>
      <param name="mnOrderNumber">0</param>
      <param name="szOrderType">OP</param>
      <param name="szOrderSuffix">000</param>
      <param name="szBranchPlant">        M30</param>
      <param name="szOriginalOrderCompany"/>
      <param name="szOriginalOrderNumber"/>
      <param name="szOriginalOrderType"/>
      <param name="szRelatedOrderCompany"/>
      <param name="szRelatedOrderNumber"/>
      <param name="szRelatedOrderType"/>
      <param name="mnSupplierNumber" id="supplierNumber">17000</param>
       <param name="mnShipToNumber">6074</param>
       <param name="jdRequestedDate">2002/07/12</param>
       <param name="jdOrderDate">2000/03/02</param>
       <param name="jdPromisedDate">2002/07/12</param>
       <param name="jdCancelDate"/>
       <param name="szReference01"/>
       <param name="szReference02"/>
        <param name="szDeliveryInstructions01">
   </param>
         <param name="szDeliveryInstructions02">
   </param>
       <param name="szPrintMessage"/>
       <param name="szSupplierPriceGroup"/>
       <param name="szPaymentTerms"/>
       <param name="szTaxExplanationCode"/>
       <param name="szTaxRateArea"/>
       <param name="szTaxCertificate">                    </param>
       <param name="cAssociatedText"/>
       <param name="szHoldCode"/>
       <param name="szFreightHandlingCode"/>
       <param name="mnBuyerNumber">0</param>
       <param name="mnCarrierNumber">0</param>
       <param name="cEvaluatedReceiptsFlag">N</param>
       <param name="cSendMethod"/>
       <param name="szLandedCostRule">   </param>
       <param name="szApprovalRouteCode"/>
       <param name="mnChangeOrderNumber">0</param>
       <param name="cCurrencyMode">D</param>
       <param name="szTransactionCurrencyCode">USD</param>
       <param name="mnCurrencyExchangeRate">0</param>
       <param name="szOrderedPlacedBy">SUBSTITUTE</param>
       <param name="szOrderTakenBy"/>
       <param name="szProgramID">EP4310</param>
       <param name="szApprovalRoutePO"/>
       <param name="szPurchaseOrderPrOptVersion" id="Version">ZJDE0001</param>
       <param name="szBaseCurrencyCode">USD</param>
       <param name="szUserID">SUBSTITUTE</param>
       <param name="cAddNewLineToExistingOrder"/>
       <param name="idInternalVariables">0</param>
       <param name="cSourceOfData"/>
       <param name="mnSODOrderNumber">0</param>
       <param name="szSODOrderType"/>
       <param name="szSODOrderCompany"/>
       <param name="szSODOrderSuffix"/>
       <param name="mnRetainage">0</param>
       <param name="szDescription"/>
      <param name="szRemark"/>
      <param name="jdEffectiveDate"/>
      <param name="jdPhysicalCompletionDate"/>
      <param name="mnTriangulationRateFromCurrenc">0</param>
```

```
         <param name="mnTriangulationRateToCurrency">0</param>
         <param name="cCurrencyConversionMethod"/>
         <param name="szPriceAdjustmentScheduleN"/>
         <param name="cAIADocument"/>
         <param name="mnProcessID" id="processID">2612</param>
         <param name="mnTransactionID" id="transactionID">4</param>
      </params>
      </callMethod>
      <callMethod name="F4311EditLine" returnNullData="no" runOnError="yes"
   app="XMLTest">
      <returnCode code="0"/>
      <params>
         <param name="mnJobNumber" idref="jobNumber">3</param>
         <param name="szComputerID" idref="machineKey">XEENT</param>
         <param name="mnOrderLineNumber">1</param>
         <param name="cDetailActionCode">1</param>
         <param name="cProcessEdits">1</param>
         <param name="cUpdateOrWriteWorkFile">2</param>
         <param name="cRecordWrittenToWorkFile">1</param>
         <param name="cCurrencyProcessingFlag">Y</param>
         <param name="szPurchaseOrderPrOptVersion"
            idref="version">ZJDE0001</param>
         <param name="szOrderCompany"
            idref="orderCompany">00200</param>
         <param name="szOrderType">OP</param>
         <param name="szOrderSuffix">000</param>
         <param name="szBranchPlant">         M30</param>
         <param name="mnSupplierNumber" idref="supplierNumber">17000</param>
         <param name="mnShipToNumber">6074</param>
         <param name="jdRequestedDate">2000/03/02</param>
         <param name="jdTransactionDate">2000/03/02</param>
         <param name="jdPromisedDate">2000/03/02</param>
         <param name="jdGLDate">2000/03/02</param>
           <param name="szUnformattedItemNumber">1001
      </param>
         <param name="mnQuantityOrdered">1</param>
         <param name="mnUnitPrice">32,1000</param>
         <param name="mnExtendedPrice">32,1</param>
         <param name="szLineType">S</param>
         <param name="szDescription1">Bike Rack - Trunk Mount</param>
        <param name="szDescription2">                              </param>
        <param name="szDetailLineBranchPlant">         M30</param>
        <param name="szLocation">  .   .            </param>
        <param name="szLotNumber">                             </param>
        <param name="szTransactionUoM">EA</param>
        <param name="szPurchasingUoM">EA</param>
        <param name="szLastStatus">220</param>
        <param name="szNextStatus">230</param>
        <param name="mnDiscountFactor">1</param>
        <param name="szInventoryPriceRule">        </param>
        <param name="szPrintMessage"> </param>
        <param name="cTaxable">Y</param>
        <param name="szGLClassCode">IN30</param>
        <param name="mnBuyerNumber">8444</param>
        <param name="szPurchasingCategoryCode1"> </param>
        <param name="szPurchasingCategoryCode2"> </param>
        <param name="szPurchasingCategoryCode3"> </param>
        <param name="szPurchasingCategoryCode4">240</param>
        <param name="szLandedCostRule"> </param>
        <param name="mnWeight">80</param>
```

```
  <param name="szWeightUoM">OZ</param>
  <param name="mnVolume">2,25</param>
  <param name="szVolumeUoM">FC</param>
  <param name="cEvaluatedReceipts">N</param>
  <param name="cInventoryInterface">Y</param>
  <param name="szTransactionCurrencyCode">USD</param>
  <param name="szBaseCurrencyCode">USD</param>
  <param name="cSourceRequestingPOGeneration">0</param>
  <param name="szProgramID">XMLTest</param>
  <param name="szUserID">SUBSTITUTE</param>
  <param name="szAgreementNumber"/>
  <param name="mnAgreementSupplement">0</param>
  <param name="jdEffectiveDate"/>
  <param name="szPurchasingCostCenter"/>
  <param name="szObjectAccount"/>
  <param name="szSubsidiary"/>
  <param name="cStockingType">P</param>
  <param name="mnProcessID" idref="processID">2612</param>
  <param name="mnTransactionID" idref="transactionID">4</param>
  <param name="mnIdentifierShortItem">60003</param>
</params>
</callMethod>
<callMethod name="F4311EditLine" returnNullData="no"
      runOnError="yes" app="XMLTest">
<returnCode code="0"/>
<params>
  <param name="mnJobNumber" idref="jobNumber">3</param>
  <param name="szComputerID" idref="machineKey">XEENT</param>
  <param name="mnOrderLineNumber">2</param>
  <param name="cDetailActionCode">1</param>
  <param name="cProcessEdits">1</param>
  <param name="cUpdateOrWriteWorkFile">2</param>
  <param name="cRecordWrittenToWorkFile">1</param>
  <param name="cCurrencyProcessingFlag">Y</param>
  <param name="szPurchaseOrderPrOptVersion"
      idref="version">ZJDE0001</param>
  <param name="szOrderCompany"
      idref="orderCompany">00200</param>
  <param name="szOrderType">OP</param>
  <param name="szOrderSuffix">000</param>
  <param name="szBranchPlant">         M30</param>
  <param name="mnSupplierNumber"
   idref="supplierNumber">17000</param>
  <param name="mnShipToNumber">6074</param>
  <param name="jdRequestedDate">2000/03/02</param>
  <param name="jdTransactionDate">2000/03/02</param>
  <param name="jdPromisedDate">2000/03/02</param>
  <param name="jdGLDate">2000/03/02</param>
  <param name="szUnformattedItemNumber">2001
  </param>
  <param name="mnQuantityOrdered">3</param>
  <param name="mnUnitPrice">164,0817</param>
  <param name="mnExtendedPrice">492,2451</param>
  <param name="szLineType">S</param>
  <param name="szDescription1">Cro-Moly Frame, Red           </param>
  <param name="szDescription2">                              </param>
  <param name="szDetailLineBranchPlant">         M30</param>
  <param name="szLocation">  .   .            </param>
  <param name="szLotNumber">                              </param>
  <param name="szTransactionUoM">EA</param>
```

```
          <param name="szPurchasingUoM">EA</param>
          <param name="szLastStatus">220</param>
          <param name="szNextStatus">230</param>
          <param name="mnDiscountFactor">1</param>
          <param name="szInventoryPriceRule">        </param>
          <param name="szPrintMessage"> </param>
          <param name="cTaxable">Y</param>
          <param name="szGLClassCode">IN30</param>
          <param name="szPurchasingCategoryCode1"> </param>
          <param name="szPurchasingCategoryCode2"> </param>
          <param name="szPurchasingCategoryCode3"> </param>
          <param name="szPurchasingCategoryCode4">200</param>
          <param name="szLandedCostRule"> </param>
          <param name="mnWeight">3</param>
          <param name="szWeightUoM">OZ</param>
          <param name="szVolumeUoM">FC</param>
          <param name="cEvaluatedReceipts">N</param>
          <param name="cInventoryInterface">Y</param>
          <param name="szTransactionCurrencyCode">USD</param>
          <param name="szBaseCurrencyCode">USD</param>
          <param name="cSourceRequestingPOGeneration">0</param>
          <param name="szProgramID">XMLTest</param>
          <param name="szUserID">SUBSTITUTE</param>
          <param name="szAgreementNumber"/>
          <param name="mnAgreementSupplement">0</param>
          <param name="jdEffectiveDate"/>
          <param name="szPurchasingCostCenter"/>
          <param name="szObjectAccount"/>
          <param name="szSubsidiary"/>
          <param name="cStockingType">M</param>
          <param name="mnProcessID" idref="processID">2612</param>
          <param name="mnTransactionID" idref="transactionID">4</param>
          <param name="mnIdentifierShortItem">60062</param>
        </params>
        </callMethod>
        <callMethod name="F4311EditDoc" returnNullData="no"
              runOnError="no" app="XMLTest">
        <returnCode code="0"/>
        <params>
          <param name="szOrderSuffix">000</param>
          <param name="szComputerID" idref="machineKey">XEENT</param>
          <param name="mnJobnumber" idref="jobNumber">3</param>
          <param name="mnAddressNumber"
              idref="supplierNumber">17000</param>
          <param name="szOrderType">OP</param>
          <param name="szOrderCompany"
              idref="orderCompany">00200</param>
          <param name="szVersionProcOption"
              idref="version">ZJDE0001</param>
          <param name="cActionCode">A</param>
          <param name="mnProcessID" idref="processID">2612</param>
          <param name="mnTransactionID" idref="transactionID">4</param>
     </params>
        </callMethod>
        <callMethod name="F4311EndDoc" returnNullData="no"
              runOnError="no" app="XMLTest">
        <returnCode code="0"/>
        <params>
          <param name="szComputerID" idref="machineKey">XEENT</param>
          <param name="mnJobNumber" idref="jobNumber">3</param>
```

```
      <param name="szCallingApplicationName">XMLTest</param>
      <param name="szVersion" idref="version">ZJDE0001</param>
      <param name="szUserID">SUBSTITUTE</param>
      <param name="mnOrderNumberAssigned"
          id="orderNumberAssigned">4884</param>
      <param name="cUseWorkFiles">2</param>
      <param name="cConsolidateLines">0</param>
      <param name="mnProcessID" idref="processID">2612</param>
      <param name="mnTransactionID" idref="transactionID">4</param>
    </params>
    </callMethod>
    <returnParams>
      <param name="JobNumber" idref="machineKey">XEENT</param>
      <param name="ComputerID" idref="jobNumber">3</param>
      <param name="OrderNumberAssigned" idref="orderNumberAssigned">4884</param>
</returnParams>
</jdeResponse>
```

Sample Sales Order Request

The following is a sample Sales Order request.

Executing a Sales Order Request

The following is an example of a Sales Order request.

```
<?xml version='1.0' encoding='utf-8' ?>
<jdeRequest type='callmethod' user='JDE' pwd='JDE' environment='DV7333'>
   <callMethod name='GetLocalComputerId' app='XMLInterop'
         runOnError='no'>
   <params>
     <param name='szMachineKey' id='2'></param>
   </params>
   <onError abort='yes'>
   </onError>
   </callMethod>
   <callMethod name='F4211FSBeginDoc' app='XMLInterop'
         runOnError='no'>
   <params>
     <param name='mnCMJobNumber' id='1'></param>
     <param name='cCMDocAction'>A</param>
     <param name='cCMProcessEdits'>1</param>
     <param name='szCMComputerID' idref='2'></param>
     <param name='cCMUpdateWriteToWF'>2</param>
     <param name='szCMProgramID'>XMLInterop</param>
     <param name='szCMVersion'>ZJDE0001</param>
     <param name='szOrderType'>SO</param>
     <param name='szBusinessUnit'>         M30</param>
     <param name='mnAddressNumber'>4242</param>
     <param name='jdOrderDate'>2000/03/29</param>
     <param name='szReference'>10261</param>
     <param name='cApplyFreightYN'>Y</param>
     <param name='szCurrencyCode'></param>
     <param name='cWKSourceOfData'></param>
     <param name='cWKProcMode'></param>
     <param name='mnWKSuppressProcess'>0</param>
   </params>
   <onError abort='yes'>
   <callMethod name='F4211ClearWorkFile' app='XMLInterop'
      runOnError='yes'>
   <params>
<param name='mnJobNo' idref='1'></param>
<param name='szComputerID' idref='2'></param>
<param name='mnFromLineNo'>0</param>
<param name='mnThruLineNo'>0</param>
<param name='cClearHeaderWF'>2</param>
<param name='cClearDetailWF'>2</param>
<param name='szProgramID'>XMLInterop</param>
<param name='szCMVersion'>ZJDE0001</param>
   </params>
   </callMethod>
   </onError>
   </callMethod>
   <callMethod name='F4211FSEditLine' app='XMLInterop'
      runOnError='yes'>
   <params>
     <param name='mnCMJobNo' idref='1'></param>
     <param name='cCMLineAction'>A</param>
     <param name='cCMProcessEdits'>1</param>
     <param name='cCMWriteToWFFlag'>2</param>
     <param name='szCMComputerID' idref='2'></param>
<!-- param name='mnLineNo'>10261</param -->
     <param name='szItemNo'>1001</param>
     <param name='mnQtyOrdered'>1</param>
     <param name='cSalesTaxableYN'>N</param>
     <param name='szTransactionUOM'>EA</param>
     <param name='szCMProgramID'>XMLInterop</param>
     <param name='szCMVersion'>ZJDE0001</param>
     <param name='cWKSourceOfData'></param>
   </params>
   <onError abort='no'>
   </onError>
   </callMethod>
   <callMethod name='F4211FSEditLine' app='XMLInterop'
      runOnError='yes'>
   <params>
     <param name='mnCMJobNo' idref='1'></param>
     <param name='cCMLineAction'>A</param>
     <param name='cCMProcessEdits'>1</param>
     <param name='cCMWriteToWFFlag'>2</param>
     <param name='szCMComputerID' idref='2'></param>
<!-- param name='mnLineNo'>10262</param -->
     <param name='szItemNo'>1001</param>
     <param name='mnQtyOrdered'>10</param>
     <param name='cSalesTaxableYN'>N</param>
     <param name='szTransactionUOM'>EA</param>
     <param name='szCMProgramID'>XMLInterop</param>
     <param name='szCMVersion'>ZJDE0001</param>
     <param name='cWKSourceOfData'></param>
   </params>
   <onError abort='no'>
   </onError>
   </callMethod>
   <callMethod name='F4211FSEndDoc' app='XMLInterop'
      runOnError='no'>
   <params>
     <param name='mnCMJobNo' idref='1'></param>
     <param name='szCMComputerID' idref='2'></param>
     <param name='szCMProgramID'>XMLInterop</param>
     <param name='szCMVersion'>ZJDE0001</param>
     <param name='cCMUseWorkFiles'>2</param>
   </params>
   <onError abort='no'>
   <callMethod name='F4211ClearWorkFile' app='XMLInterop'
      runOnError='yes'>
<params>
     <param name='mnJobNo' idref='1'></param>
     <param name='szComputerID' idref='2'></param>
     <param name='mnFromLineNo'>0</param>
     <param name='mnThruLineNo'>0</param>
     <param name='cClearHeaderWF'>2</param>
     <param name='cClearDetailWF'>2</param>
     <param name='szProgramID'>XMLInterop</param>
     <param name='szCMVersion'>ZJDE0001</param>
   </params>
   </callMethod>
   </onError>
   </callMethod>
   <returnParams failureDestination='ERROR.Q'
      successDestination='SUCCESS.Q' runOnError='yes'>
   </returnParams>
   <onError abort='yes'>
   <callMethod name='F4211ClearWorkFile' app='XMLInterop'
      runOnError='yes'>
   <params>
     <param name='mnJobNo' idref='1'></param>
     <param name='szComputerID' idref='2'></param>
     <param name='mnFromLineNo'>0</param>
     <param name='mnThruLineNo'>0</param>
     <param name='cClearHeaderWF'>2</param>
     <param name='cClearDetailWF'>2</param>
     <param name='szProgramID'>XMLInterop</param>
     <param name='szCMVersion'>ZJDE0001</param>
   </params>
   </callMethod>
   </onError>
   </jdeRequest>
```

Sample Sales Order Response

This is the corresponding response document for the Sales Order request. There are error messages returned in the document. The error messages can be used within a workflow. For example:

```
<error code="2597">Warning: WARNING: Duplicate Customer Order Number
</error>
<error code="4136">Warning: Pick date is less than todays date</error>
```

Using the Sales Order Response

The following is the jdeResponse document.

```
<?xml version="1.0" encoding="utf-8" ?>
<jdeResponse environment="DV7333" user="JDE" type="callmethod" pwd="JDE">

  <callMethod name="GetLocalComputerId" runOnError="no"
      app="XMLInterop">
   <returnCode code="0"/>
   <params>
      <param name="szMachineKey" id="2">XEENT</param>
   </params>
```

```
</callMethod><callMethod name="F4211FSBeginDoc" runOnError="no"
   app="XMLInterop">
<returnCode code="1"/>
<params>
  <param name="mnCMJobNumber" id="1">3</param>
  <param name="cCMDocAction">A</param>
  <param name="cCMProcessEdits">1</param>
  <param name="szCMComputerID" idref="2">XEENT</param>
  <param name="cCMErrorConditions">1</param>
  <param name="cCMUpdateWriteToWF">2</param>
  <param name="szCMProgramID">XMLInterop</param>
  <param name="szCMVersion">ZJDE0001</param>
  <param name="szOrderCo">00200</param>
  <param name="szOrderType">SO</param>
  <param name="szBusinessUnit">              M30</param>
  <param name="mnAddressNumber">4242</param>
  <param name="mnShipToNo">4242</param>
  <param name="jdRequestedDate">2000/03/29</param>
  <param name="jdOrderDate">2000/03/29</param>
  <param name="jdPromisedDate">2000/03/29</param>
  <param name="szReference">10261</param>
  <param name="szDeliveryInstructions1">              </param>
  <param name="szDeliveryInstructions2">               </param>
  <param name="szPrintMesg">          </param>
  <param name="szPaymentTerm">   </param>
  <param name="cPaymentInstrument"> </param>
  <param name="mnTradeDiscount">,000</param>
  <param name="szTaxExplanationCode">S </param>
  <param name="szTaxArea">DEN       </param>
  <param name="szCertificate">                    </param>
  <param name="szHoldOrdersCode">  </param>
  <param name="cPricePickListYN">Y</param>
  <param name="szRouteCode">   </param>
  <param name="szStopCode">   </param>
  <param name="szZoneNumber">   </param>
  <param name="szFreightHandlingCode">   </param>
  <param name="cApplyFreightYN">Y</param>
  <param name="mnCommissionCode1">6001</param>
  <param name="mnCommissionRate1">5,000</param>
  <param name="mnCommissionRate2">,000</param>
  <param name="szWeightDisplayUOM">  </param>
  <param name="szVolumeDisplayUOM">  </param>
  <param name="cMode">D</param>
  <param name="szCurrencyCode">USD</param>
  <param name="jdDateUpdated">2002/07/12</param>
  <param name="szWKBaseCurrency">USD</param>
  <param name="cWKAdvancedPricingYN">N</param>
  <param name="szWKCreditMesg">  </param>
  <param name="szWKTempCreditMesg">  </param>
  <param name="cWKSourceOfData"/>
  <param name="cWKProcMode"/>
  <param name="mnWKSuppressProcess">0</param>
  <param name="szPricingGroup">PREFER  </param>
  <param name="mnProcessID">2252</param>
  <param name="mnTransactionID">4</param>
</params><errors><error code="2597">Warning: WARNING: Duplicate
   Customer Order Number</error><error code="4136">Warning: Pick
  date is less than todays date</error></errors>
</callMethod><callMethod name="F4211FSEditLine" runOnError="yes"
   app="XMLInterop">
```

```
<returnCode code="1"/><params>
     <param name="mnCMJobNo" idref="1">3</param>
     <param name="cCMLineAction">A</param>
     <param name="cCMProcessEdits">1</param>
     <param name="cCMWriteToWFFlag">2</param>
     <param name="cCMRecdWrittenToWF">1</param>
     <param name="szCMComputerID" idref="2">XEENT</param>
     <param name="cCMErrorConditions">1</param>
     <param name="szOrderCo">00200</param>
      <param name="szOrderType">SO</param>     <param name="szBusinessUnit">
      M30</param>
     <param name="mnShipToNo">4242</param>
     <param name="jdRequestedDate">2000/03/29</param>
     <param name="jdPromisedDate">2000/03/29</param>
     <param name="jdPromisedDlvryDate">2000/03/29</param>
     <param name="szItemNo">1001                     </param>
     <param name="szLocation">  .   .   </param>
     <param name="szDescription1">Bike Rack Trunk Mount </param>
     <param name="szDescription2">                       </param>
     <param name="szLineType">S</param>
     <param name="szLastStatus">900</param>
     <param name="szNextStatus">540</param>
     <param name="mnQtyOrdered">1</param>
     <param name="mnQtyBackordered">1</param>
     <param name="mnUnitPrice">44,99</param>
     <param name="mnUnitCost">32,1000</param>
     <param name="szPrintMesg">          </param>
     <param name="cPaymentInstrument"> </param>
     <param name="cSalesTaxableYN">N</param>
     <param name="cAssociatedText"> </param>
     <param name="szTransactionUOM">EA</param>
     <param name="szPricingUOM">EA</param>
     <param name="mnItemWeight">80</param>
     <param name="szWeightUOM">OZ</param>
     <param name="mnForeignUnitPrice">44,99</param>
     <param name="mnForeignUnitCost">32,1000</param>
     <param name="mnDiscountFactor">1</param>
     <param name="mnCMLineNo">1</param>
     <param name="szCMProgramID">XMLInterop</param>
     <param name="szCMVersion">ZJDE0001</param>
     <param name="mnSupplierNo">4343</param>
     <param name="mnWKOrderTotal">44,99</param>
     <param name="mnWKForeignOrderTotal">44,99</param>
     <param name="mnWKTotalCost">32,1</param>
     <param name="mnWKForeignTotalCost">32,1</param>
     <param name="cWKSourceOfData"/>
     <param name="cWKCheckAvailability">1</param>
     <param name="mnLastLineNoAssigned">1</param>
     <param name="cStockingType">P</param>
     <param name="cParentItmMethdOfPriceCalcn">1</param>
     <param name="mnShortItemNo">60003</param>
     <param name="szSalesOrderFlags">0</param>
     <param name="jdPriceEffectiveDate">2000/03/29</param>
     <param name="jdPromisedShip">2000/03/29</param>
     <param name="mnQuantityAvailable">-34</param>
     <param name="mnItemVolume_ITVL">2,25</param>
     <param name="szVolumeUOM_VLUM">FC</param>
     <param name="szRevenueBusinessUnit">  M30</param>
     <param name="mnProcessID">2252</param>
     <param name="mnTransactionID">4</param>
```

```
</params><errors><error code="030B">Warning: Order Quantity
   Exceeds what&apos;s Available</error></errors>
</callMethod><callMethod name="F4211FSEditLine" runOnError="yes"
   app="XMLInterop"><returnCode code="1"/><params>
  <param name="mnCMJobNo" idref="1">3</param>
  <param name="cCMLineAction">A</param>
  <param name="cCMProcessEdits">1</param>
  <param name="cCMWriteToWFFlag">2</param>
  <param name="cCMRecdWrittenToWF">1</param>
  <param name="szCMComputerID" idref="2">XEENT</param>
  <param name="cCMErrorConditions">1</param>
  <param name="szOrderCo">00200</param>
  <param name="szOrderType">SO</param>
  <param name="szBusinessUnit">         M30</param>
  <param name="mnShipToNo">4242</param>
  <param name="jdRequestedDate">2000/03/29</param>
  <param name="jdPromisedDate">2000/03/29</param>
  <param name="jdPromisedDlvryDate">2000/03/29</param>
  <param name="szItemNo">1001                     </param>
  <param name="szLocation">  .   .   </param>
  <param name="szDescription1">Bike Rack-Trunk Mount   </param>
  <param name="szDescription2">                        </param>
  <param name="szLineType">S</param>
  <param name="szLastStatus">900</param>
  <param name="szNextStatus">540</param>
  <param name="mnQtyOrdered">10</param>
  <param name="mnQtyBackordered">10</param>
  <param name="mnUnitPrice">44,99</param>
  <param name="mnUnitCost">32,1000</param>
  <param name="szPrintMesg">          </param>
  <param name="cPaymentInstrument"> </param>
  <param name="cSalesTaxableYN">N</param>
  <param name="cAssociatedText"> </param>
  <param name="szTransactionUOM">EA</param>
  <param name="szPricingUOM">EA</param>
  <param name="mnItemWeight">800</param>
  <param name="szWeightUOM">OZ</param>
  <param name="mnForeignUnitPrice">44,99</param>
  <param name="mnForeignUnitCost">32,1000</param>
  <param name="mnDiscountFactor">1</param>
  <param name="mnCMLineNo">2</param>
  <param name="szCMProgramID">XMLInterop</param>
  <param name="szCMVersion">ZJDE0001</param>
  <param name="mnSupplierNo">4343</param>
  <param name="mnWKOrderTotal">494,89</param>
  <param name="mnWKForeignOrderTotal">494,89</param>
  <param name="mnWKTotalCost">321</param>
  <param name="mnWKForeignTotalCost">321</param>
  <param name="cWKSourceOfData"/>
  <param name="cWKCheckAvailability">1</param>
  <param name="mnLastLineNoAssigned">2</param>
  <param name="cStockingType">P</param>
  <param name="cParentItmMethdOfPriceCalcn">1</param>
  <param name="mnShortItemNo">60003</param>
  <param name="szSalesOrderFlags">                 0   </param>
  <param name="jdPriceEffectiveDate">2000/03/29</param>
  <param name="jdPromisedShip">2000/03/29</param>
  <param name="mnQuantityAvailable">-44</param>
  <param name="mnItemVolume_ITVL">22,5</param>
  <param name="szVolumeUOM_VLUM">FC</param>
  <param name="szRevenueBusinessUnit">         M30</param>
  <param name="mnProcessID">2252</param>
  <param name="mnTransactionID">4</param>
</params><errors><error code="030B">Warning: Order Quantity
   Exceeds what&apos;s Available</error></errors>
</callMethod><callMethod name="F4211FSEndDoc" runOnError="no"
   app="XMLInterop"><returnCode code="0"/>
<params>
  <param name="mnCMJobNo" idref="1">3</param>
  <param name="mnSalesOrderNo">2623</param>
  <param name="szCMComputerID" idref="2">XEENT</param>
  <param name="cCMErrorCondition">0</param>
  <param name="szOrderType">SO</param>
  <param name="szKeyCompany">00200</param>
  <param name="mnOrderTotal">494,89</param>
  <param name="szWorkstationID">XEENT</param>
  <param name="szCMProgramID">XMLInterop</param>
  <param name="szCMVersion">ZJDE0001</param>
  <param name="mnTimeOfDay">174220</param>
  <param name="cCMUseWorkFiles">2</param>
  <param name="cCMProcessEdits">1</param>
  <param name="mnProcessID">2252</param>
  <param name="mnTransactionID">4</param>
</params>   </callMethod><returnParams failureDestination="ERROR.Q"
   successDestination="SUCCESS.Q">
</returnParams></jdeResponse>
```

Glossary

adapter

Provides universal connectivity by enabling an electronic interface to be accommodated (without loss of function) to another electronic interface.

agent

Supports service protocols in listeners and documents.

business service

Also known as a Web service. A Web service is a self-contained, modularized function that can be published and accessed across a network using open standards. It is the implementation of an interface by a component and is an executable entity.

channel

Represents configured connections to particular instances of back-end systems. A channel binds one or more event ports to a particular listener managed by an adapter.

listener

A component that accepts requests from client applications.

port

Associates a particular business object exposed by the adapter with a particular disposition. A disposition is a URL that defines the protocol and location of the event data. The port defines the end point of the event consumption.

Index